An Empirical Study of Meta- and Hyper-Heuristic Search for Multi-Objective Release Planning

Yuanyuan Zhang, CREST, University College London, UK
Mark Harman, CREST, University College London, UK
Gabriela Ochoa, University of Stirling, UK
Guenther Ruhe, University of Calgary, Canada
Sjaak Brinkkemper, Utrecht University, The Netherlands

A variety of meta-heuristic search algorithms have been introduced for optimising software release planning. However, there has been no comprehensive empirical study of different search algorithms across multiple different real world datasets. In this paper we present an empirical study of global, local and hybrid meta- and hyper-heuristic search based algorithms on 10 real world datasets. We find that the hyper-heuristics are particularly effective. For example, the hyper-heuristic genetic algorithm significantly outperformed the other six approaches (and with high effect size) for solution quality 85% of the time, and was also faster than all others 70% of the time. Furthermore, correlation analysis reveals that it scales well as the number of requirements increases.

Categories and Subject Descriptors: Software Engineering [**Requirements/Specifications**]

General Terms: Algorithms, Experimentation, Measurement

Additional Key Words and Phrases: Strategic Release Planning, Meta-Heuristics, Hyper-Heuristics

ACM Reference Format:
Yuanyuan Zhang, Mark Harman, Gabriela Ochoa, Guenther Ruhe and Sjaak Brinkkemper, 2017. An Empirical Study of Meta- and Hyper-Heuristic Search for Multi-Objective Release Planning. *ACM Trans. Softw. Eng. Methodol.* V, N, Article A (January YYYY), 32 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Release planning is the problem of determining the sets of requirements that should be included in a set of upcoming releases of a software system. In order to plan the software release cycle, a number of different conflicting objectives need to be taken into account. For example, the estimated cost of implementing a requirement has to be balanced against the perceived value to the customer of that requirement. There may be multiple stakeholders, and their different interpretations of cost and value may lead to complex solution spaces for project managers.

In order to help decision-makers navigate these complex solution spaces, meta-heuristic search has been widely studied as a candidate solution technique [Bagnall et al. 2001; Ruhe and Greer 2003; Zhang et al. 2007]. This work has placed release planning within the general area of Search Based Software Engineering (SBSE) [Harman and Jones 2001]. Table I summarises the literature on search based release planning, listing the meta-heuristic algorithms proposed and the datasets on which they have been evaluated.

The different cost and value objectives for each stakeholder are typically measured along incomparable dimensions. To avoid the familiar problem of 'comparing apples with oranges', much of the previous work on multi-objective release planning has used Pareto optimal search. The result of such a search is a Pareto front. Each element on this front is a candidate solution to the release planning problem. All solutions on the Pareto front are non-denominated: no other solution on the front is better according to all objectives. The Pareto front thus represents a set of 'best compromises' between the objectives that can be found by the search based algorithm.

The overview in Table I reveals that previous work has evaluated meta-heuristic algorithms on very few real world datasets. Much of the previous work presents results for only a single

real world dataset. In the absence of real world datasets, many authors have relied upon synthetically generated data. While studies on synthetically generated data can answer experimental research questions [Harman et al. 2012a], they cannot address the essential empirical question that will be asked by any release planner: "how well can I expect these techniques to behave on real world data?"

As a result, the state-of-the-art is currently poorly understood: though a variety of different algorithms has been proposed, there has been no empirical study across multiple different algorithms and multiple different datasets. We wish to address this issue by providing a thorough empirical study of optimised release planning. We believe that this may help to understand the different strengths and weaknesses of algorithms for release planning and their performance on real world datasets. We hope that our study will also provide results against which future work can compare[1].

We report the results of an empirical study using 10 real world datasets. We investigate multi-objective release planning with respect to these datasets, which we optimise using Hill Climbing (HC), Genetic Algorithms (GA) and Simulated Annealing (SA).

As a sanity check, recommended for SBSE work [Arcuri and Briand 2011; Harman et al. 2012c], we also report results for purely random search. Random search provides a baseline against which to benchmark more 'intelligent' search techniques. Our study also includes hyper-heuristic versions of HC, GA and SA. Hyper-heuristics [Burke et al. 2013] are a more recent trend in search methodologies, not previously been used in any SBSE research. The findings we report here indicate that they are promising for release planning problems.

Overall, our study thus involves 7 different algorithms. We assess solutions found using these algorithms according to 4 different measures of solution quality, over each of the 10 real world datasets. We include standard, widely used, measures of multi-objective solution quality: convergence, hypervolume and two different assessments of each algorithm's contribution to the Pareto front. We also measure diversity and speed. For algorithms that produce good quality solutions, these are important additional algorithmic properties for decision-makers, because they need quick answers that enable them to base their decisions on the full diversity of candidate solutions.

The primary contributions of the paper are as follows:

1. **Comprehensive study**: We provide a comprehensive study of the performance of global, local and hybrid meta-heuristic algorithms for release planning problems on 10 real-world datasets. The results facilitate detailed algorithm comparison and reveal that dataset specifics can lead to important differences in study findings.
2. **Introduce hyper-heuristic search**: We introduce and evaluate hyper-heuristic algorithms for release planning. We present evidence that they provide good solution quality diversity and speed.
3. **Scalability assessment**: We investigate the scalability of meta- and hyper-heuristic algorithms on real world datasets for the first time. The results provide evidence that hyper-heuristics have attractive scalability and that random search is surprisingly *un*scalable for release planning.

The rest of the paper is organised as follows: Section 2 sets our work in the context of related work. Section 3 introduces our three hyper-heuristic release planning algorithms. Section 4 explains our experimental methodology. Section 5 presents the results and discusses the findings. Section 6 analyses threats to validity. Section 7 concludes the paper.

[1]Note to referee: if the paper is accepted then we will make all our implementations and data publicly available on the web to support replication and further study.

Table I. Previous meta-heuristic algorithm studies

Author(s) [Paper]	Algorithm(s)	M/S Objective	Dataset(s)
Bagnall *et al.* [Bagnall et al. 2001]	Hill Climbing Algorithm (HC), Simulated Annealing (SA), Greedy Algorithm	Single	Synthetic
Feather & Menzies [Feather and Menzies 2002], Feather *et al.* [Feather et al. 2004], Feather *et al.* [Feather et al. 2006]	SA	Single	NASA
Ruhe & Greer [Ruhe and Greer 2003], Ruhe & Ngo-The [Ruhe and Ngo-The 2004], Amandeep *et al.* [Amandeep et al. 2004], Ruhe [Ruhe 2010]	Genetic Algorithm (GA)	Single & Multiple	Synthetic
Zhang *et al.* [Zhang et al. 2007]	Non-Dominated Sorting Genetic Algorithm-II (NSGA-II), Pareto GA	Multiple	Synthetic
Durillo *et al.* [Durillo et al. 2009], Durillo *et al.* [Durillo et al. 2011]	Multi-Objective Cellular genetic algorithm (MOCell), Pareto Archived Evolution Strategy (PAES), NSGA-II	Multiple	Synthetic, Motorola
Colares *et al.* [Colares et al. 2009]	NSGA-II	Multiple	Synthetic
Finkelstein *et al.* [Finkelstein et al. 2008], Zhang *et al.* [Zhang et al. 2011]	NSGA-II, Two-Archive Algorithm	Multiple	Motorola
Zhang & Harman [Zhang and Harman 2010], Zhang *et al.* [Zhang et al. 2013]	NSGA-II, Archive-based NSGA-II	Multiple	Synthetic, RALIC
Sagrado & Águila [Del Sagrado and Del Águila 2009], Sagrado *et al.* [Del Sagrado et al. 2010]	Ant Colony Optimization (ACO), SA, GA	Single	Synthetic
Jiang *et al.* [Jiang et al. 2010a], Xuan *et al.* [Xuan et al. 2012]	Backbone Multilevel Algorithm	Single	Eclipse, Mozilla, Gnome
Zhang *et al.* [Zhang et al. 2010]	NSGA-II	Multiple	Ericsson
Jinag *et al.* [Jiang et al. 2010b]	Hybrid ACO, ACO, SA	Single	Synthetic
Kumari *et al.* [Kumari et al. 2012]	Quantum-inspired Elitist Multi-objective Evolutionary Algorithm (QEMEA)	Multiple	Synthetic
Souza *et al.* [de Souza et al. 2011], Ferreira & Souza [do Nascimento Ferreira and de Souza 2012]	ACO	Single	Synthetic
Brasil *et al.* [Brasil et al. 2012]	NSGA-II, MOCell	Multiple	Synthetic
Cai *et al.* [Cai et al. 2012], Cai & Wei [Cai and Wei 2013]	Domination and Decomposition based Multi-Objective Evolutionary Optimization (MOEA/DD), Strength Pareto Evolutionary Algorithm-II (SPEA2), NSGA-II	Multiple	Synthetic
Tonella *et al.* [Tonella et al. 2013]	Interactive GA (IGA), Incomplete Analytic Hierarchy Process (IAHP)	Single	ACube
Paixão & Souza [Paixão and de Souza 2013a], Paixão & Souza [Paixão and de Souza 2013b], Paixão & Souza [Paixão and de Souza 2015]	GA, SA	Single	Synthetic, Eclipse, Mozilla
Li *et al.* [Li et al. 2014]	NSGA-II	Multiple	Synthetic, Motorola
Chaves-González & Pérez-Toledano [Chaves-González and Pérez-Toledano 2015]	Multi-objective Differential Evolution (MODE)	Multiple	Synthetic
Sagrado *et al.* [del Sagrado et al. 2015], Águila & Sagrado [del Águila and Sagrado 2016]	Greedy Randomized Adaptive Search Procedure (GRASP), NSGA-II, Multi-objective ACO	Multiple	Synthetic
Li *et al.* [Li et al. 2017]	Cellular Algorithm hybridised with Differential Evolution (CellDE), Multi-objective Particle Swarm Optimization (SMPSO), Alternating Variable Method (AVM), GA, (1+1) EA, NSGA-II, PAES, SPEA2	Single & Multiple	Synthetic, a CPS of energy domain
Karim & Ruhe *et al.* [Karim and Ruhe 2014]	NSGA-II	Multiple	Ms Word, Theme-based RP datasets
Pitangueira *et al.* [Pitangueira et al. 2017]	NSGA-II	Multiple	
Araújo *et al.* [Araújo et al. 2017]	IGA	Single	

2. THE CONTEXT OF OUR STUDY

Bagnall *et al.* [Bagnall et al. 2001] first suggested the term *Next Release Problem* and described various meta-heuristic optimisation algorithms for solving it. Feather and Menzies [Feather and Menzies 2002] were the first to use a real world dataset, but this dataset is no longer publicly available. Ruhe *et al.* [Amandeep et al. 2004; Ruhe and Greer 2003; Ruhe and Ngo-The 2004; Saliu and Ruhe 2005] introduced the software release planning process together with exact optimisation algorithms [Al-Emran et al. 2010; AlBourae et al. 2006; Ruhe 2010; Ruhe and Saliu 2005] and meta-heuristics, such as genetic algorithms. Van den Akker *et al.* [Li et al. 2010; Van den Akker et al. 2005a,b, 2008] and Veerapen *et al.* [Veerapen et al. 2015] also studied exact approaches to single objective constrained requirements selection problems.

Zhang *et al.* [Zhang et al. 2007] introduced the Multi-Objective Next Release Problem formulation as a Pareto optimal problem with a set of objectives. However, Feather *et al.* [Feather et al. 2006, 2004] had previously used Simulated Annealing to construct a form of Pareto front for visualisation of choices. Also, at the same time, Saliu and Ruhe [Saliu and Ruhe 2007] introduced a multi-objective search-based optimisation to balance the tension between user-level and system-level requirements. Subsequently, Finkelstein *et al.* [Finkelstein et al. 2009] used multi-objective formulations to characterise different notions of fairness in the satisfaction of multiple stakeholders with different views on the priorities for requirement choices.

Table I lists the previous meta-heuristic algorithm studies for selecting requirements in next release problem. The multi-objective formulation are used in many studies, thereby adopting different multi-objective meta-heuristic algorithms to find good enough solutions. Svahnberg *et al.* [Svahnberg et al. 2010] conduct a systematic review on release planning models and find that there are a number of release planning models proposed by using similar techniques. Several hard and soft constraints are also considered as requirement selection factors. Recently, uncertainty analysis [Li et al. 2014] and risk handling [Pitangueira et al. 2017, 2016] in release planning are studied to identify the sensitive requirements and evaluate the robustness of the release plan in the presence of uncertainty.

The multi-objective formulation subsumes previous single objective formulations: any single objective formulation that has a single objective and no constraints is clearly a special case of a multi-objective formulation for n objectives where $n = 1$. Furthermore, a constrained single objective formulation, in which there is a single optimisation objective and a set of constraints to be satisfied, can be transformed into a multi-objective formulation in which the constraints become additional objectives to be met.

The technical details of the various approaches used for constrained single objective formulations and their multi-objective counterparts are, of course, different. However, in this paper we want to study the most general setting in which requirements optimisation choices might be cast. Therefore, we adopt the multi-objective paradigm.

In each formulation of the Next Release Problem or the Release Planning problem in the literature, there is a slightly different formulation. The Next Release Problem (NRP) considers only a single release, while Release Planning (RP) considers a series of releases. NRP is thus a special case of the RP. Since RP is the more general case, this is the formulation we shall study in this paper.

In the RP process, software requirements prioritisaton activity is interrelated to requirements selection. Achimugu *et al.* [Achimugu et al. 2014] and Pitangueira *et al.* [Pitangueira et al. 2015] describe the existing prioritisation techniques and their limitations. Some of these strategies arrange the requirements in a hierarchy; some cluster the requirements into several groups by different priority levels using a verbal ranking; some rely on relative values by pairwise comparison using a numerical ranking; some use discrete values, the others use the continuous scale. One of the advantages of a numerical ranking is that it can be sorted. These

priority-based approaches usually assign a rank order or level to each requirement from the 'best' to the 'worst'.

Compared with these priority-based methods, we can provide more than one (usually many) optimal alternative solutions within a certain criterion. As such, the requirements engineer has the opportunity to observe the impact of including or excluding certain requirements, and can use this to choose the best from the different alternatives, without affecting the quality of solutions.

2.1. Representation of Release Planning

Requirements management in RP is formulated as a combinatorial optimisation problem. Approaches to the NRP represent the solution as a bitset of requirements for the next release. The RP formulation, being more general, is typically represented as a sequence of integers that denote release sequence numbers.

It is assumed that for an existing software system, there is a set of stakeholders, $C = \{c_1, \ldots, c_j, \ldots, c_m\}$ whose requirements are to be considered in the development of the releases of the software. Each stakeholder may have a degree of importance for the project that can be reflected by a weight factor. More important stakeholder has a higher level of influence on the project, thereby deserving greater weight. There are a number of techniques to prioritise lists of stakeholder [Lim 2010], which can be used to produce the weight set. The set of relative weights associated with each stakeholder c_j $(1 \leq j \leq m)$ is denoted by a weight set: $Weight\_Sta = \{ws_1, \ldots, ws_j, \ldots, ws_m\}$ where $ws_j \in [0, 1]$ and $\sum_{j=1}^{m} ws_j = 1$.

The set of requirements is denoted by: $\Re = \{r_1, \ldots, r_i, \ldots, r_n\}$. The study is based on the assumption that the requirements represented in the paper are at a similar level of abstraction. They are not stated in too much detail nor on too high a level of abstraction. If the level differs among the requirements, there might be difficulty in selecting the correct subset of requirements for RP. The resources needed to implement a particular requirement can be denoted as cost terms and considered to be the associated cost to fulfil the requirement. The cost vector for the set of requirements r_i $(1 \leq i \leq n)$ is denoted by: $Cost = \{cost_1, \ldots, cost_i, \ldots, cost_n\}$. Usually, not all requirements are equally satisfied to a given stakeholder. The level of satisfaction can be denoted as a value to the stakeholders' organisations. Each stakeholder c_j $(1 \leq j \leq m)$ assigns a value to requirement r_i $(1 \leq i \leq n)$ denoted by: $v(r_i, c_j)$ where $v(r_i, c_j) > 0$ if stakeholder c_j desires implementation of the requirement r_i and 0 otherwise. The satisfaction $v(r_i, c_j)$ for the stakeholder could be represented as different terms in practice, such as revenue or the degree of importance.

There are S $(S > 1)$ possible releases considered in the model, which allow us to look further rather than just the next release. In each release s $(1 \leq s \leq S)$, a release plan vector $Plan = \{p_1, \ldots, p_i, \ldots, p_n\} \in \{0, \ldots, S\}$ determines the requirements that are to be implemented in release s. In this vector, $p_i = s$ if requirement r_i is delivered in release s and 0 if requirement r_i is not selected in the first s releases. The smaller the number of release s is, the earlier the requirement r_i is selected. That is, the RP representation we used is an integer sequence in which each index position denotes a requirement number. The value stored at this index denotes the assignment of a release number into which the corresponding requirement will be deployed. For example, a release plan has the three-release formulation ($S = 3$) with 4 requirements ($n = 4$) $Plan = \{2, 0, 3, 1\}$. That is, requirement r_1 is selected in the second release; requirement r_2 is not selected in any release; and so on.

The set of relative weights associated with each release s $(1 \leq s \leq S)$ is denoted by a weight set: $Weight\_Rel = \{wr_1, \ldots, wr_s, \ldots, wr_S\}$. The weight set $Weight\_Rel$ represents the relative importance levels among consecutive (time periods) releases and denotes how much more important to include a requirement r_i in (a former time period) release s than in (a latter time period) release S. In this paper we use the three-release formulation $S = 3$, with

$Weight\_Rel = \{5, 3, 1\}$ for the first, second and third releases respectively. When we consider the impact of release weight, the first release is the most important with the highest weight and the third release is the least important.

The decision vector $\overrightarrow{x_i} = \{x_1, \ldots, x_n\} \in \{0, \ldots, S\}$ considers both release plan vector $Plan$ and the impact of release weight ***Weight_Rel*** on a release plan. Such as, a release plan has the three-release formulation ($S = 3$) with 4 requirements ($n = 4$) $Plan = \{2, 0, 3, 1\}$ and the weight of the releases is $Weight\_Rel = \{5, 3, 1\}$. In this case, the decision vector of the release plan is $\overrightarrow{x_i} = \{3, 0, 1, 5\}$. The decision vector will be used in the fitness functions defined in Section 2.2.

2.2. Fitness Functions

In any approach to SBSE it is necessary to choose the objectives, which define the fitness function(s) used to guide the search [Harman et al. 2012b,c].

The RP problem seeks a weighted assignment that optimises for a set of objectives. Each paper has its own formulation of the objectives to be met in arriving at solutions to the NRP/RP problem considered. However, there are also some generalisations that can be undertaken for different objectives, because all approaches to NRP/RP involve a set of objectives to be maximised and a set of objectives to be minimised.

The overall value of a given requirement r_i $(1 \leq i \leq n)$, such as $Revenue_i$ can be calculated by:

$$Revenue_i = \sum_{j=1}^{m} ws_j \cdot v(r_i, c_j) \tag{1}$$

The fitness functions are defined as follows:

$$\text{Maximize } f_1(\overrightarrow{x}) = \sum_{i=1}^{n} Revenue_i \cdot x_i \tag{2}$$

$$\text{Minimize } f_2(\overrightarrow{x}) = \sum_{i=1}^{n} Cost_i \quad (if \;\; x_i > 0) \tag{3}$$

The objectives of $Revenue$ and $Cost$ in the fitness functions can also be replaced by other objectives. The specific choice of objectives to be maximised or minimised are parameters to the search based optimisation algorithm used to search for requirement sets. The algorithms use these objectives as fitness functions that guide the search. We can compare different algorithms across different datasets, because the algorithm itself does not change, merely the fitness functions used to guide the search.

3. HYPER-HEURISTIC SEARCH

Hyper-heuristics are "automated methodologies for selecting or generating heuristics to solve computational search problems" [Burke et al. 2010]. The main motivation is to reduce the need for a human expert in the process of designing optimisation algorithms, and thus raise the level of generality in which these methodologies operate. Hyper-heuristics [Burke et al. 2013] are search methodologies, more recently introduced to the optimisation literature than the meta-heuristics that have been enthusiastically adopted by the SBSE community [Harman et al. 2012b; Räihä 2010]. Hyper-heuristics are modern search methodologies successfully applied in Operational Research domains such as timetabling, scheduling and routing [Burke et al. 2013]. In the optimisation literature, hyper-heuristics have been widely applied to sin-

gle objective problem formulations, but there have been very few attempts at tackling multi-objective problems . In the wider literature on engineering and design in general (rather than software engineering in particular), hyper-heuristics have been widely applied. However, even in this wider context, the design and engineering problems attacked using hyper-heuristics have tended to be single objective problems, with only a very few previous attempts to apply hyper-heuristics to multi-objective problems [Burke et al. 2005; McClymont and Keedwell 2011]. This paper is thus the first paper in the requirements engineering literature to explore the use of hyper-heuristics and one of the few papers in the optimisation literature to use multi-objective hyper-heuristics.

Hyper-heuristics differ from meta-heuristics because they search the space of heuristics or meta-heuristics rather than the space of solutions. Some well-known meta-heuristics, such as Genetic Algorithms (GAs) [Holland 1975] and Genetic Programming (GP) [Poli et al. 2008], use strategy or procedure to guide and explore the solution space, so that they are able to come to an acceptable and reasonable solution to a problem. Instead, hyper-heuristics encapsulate problem specific information using a pool of low-level heuristics (known as search operators). The search space of hyper-heuristics is the permutation of the designed search operators. Using hyper-heuristics, we seek the sequence of operators and to find the good-enough heuristics in a given situation rather than providing a solution to the problem directly. Search operators can be of different kinds (e.g. mutation, recombination, etc). For combinatorial optimisation problems such as NRP/RP, several variation operators, involving swapping, inserting or deleting components according to different criteria are often used [Ochoa et al. 2012]. It is not easy to know before hand which of these operators will be the best suited for the problem at hand, and indeed, it is generally observed that having a combination of operators works better than using a single one.

3.1. Heuristic search operators

We designed and implemented 10 search operators for hyper-heuristic release planning, ranging from simple randomised neighbourhoods to greedy and more informed and smarter procedures. These are explained in Figure II. The first two (Random and Swap) represent standard mutation operators (i.e. they perform a small change on the solution, by swapping or changing solution components), while the remaining 8 follow the so-called 'ruin-recreate' (also known as the 'destruction-construction') principle, which has proved successful in real-world optimisation problems [Pisinger and Ropke 2007]. Ruin-recreate operators partly decompose (ruin) the solution and subsequently recreate it, incorporating problem-specific reconstruction heuristics to rebuild the solutions from their decomposed fragments.

The search space of hyperheuristics is the permutation of the designed search operators. When considering a pool of operators, a mechanism needs to be devised in order to select and apply them during the search process. Simple ways are to select operators uniformly at random, or to follow a fixed sequence. More interesting, adaptive selection mechanisms can be applied, which gather information and learn from the search process, in the form of reward statistics from previous steps, to inform the operator choice. *Adaptive Operator Selection* (AOS), is a recently coined term given to such approaches [Fialho et al. 2008, 2010], which is described below.

3.2. Adaptive Operator Selection

An adaptive operator selection scheme consists of two components, called the 'credit assignment' mechanism and the 'selection' mechanism [Fialho et al. 2010]. Credit assignment involves the attribution of credit (or reward) to the hyper-heuristic's operators, determined by their performance during the search process.

Table II. The description of 10 search operators

Operator	Description
Random	With uniform probability, select a requirement and change its release number to another release version (uniformly selected)
Swap	Swap the release numbers of two requirements in the sequence
Delete_Add	With uniform probability, exclude a requirement from the current release, and add it to another release (also selected with uniform probability)
Delete_Add_Best	With uniform probability, select a requirement, r, and an objective o, replacing r with the best performing requirement according to objective o
Delete_Worst_Add	With uniform probability, select a requirement, r, and an objective o, replacing r with the worst performing requirement according to objective o
Delete_Worst_Add_Best	With uniform probability, select an objective o, and a release number, n, replacing the worst performing requirement according to objective o at release n with the best performing requirement (according to o)
Delay_Ahead	With uniform probability, select two requirements r_1 (for a release other than the first) and r_2 (for a release other than the last). Move the release date of r_1 to a later release number (the number selected with uniform probability from those that follow its current release position). Move the release number of r_2 forward to an earlier number (selected with uniform probability from those that precede it). That is, r_1 is 'delayed' and r_2 is 'advanced'
Delay_Ahead_Best	With uniform probability, select a requirement, r, and an objective o. Delay r and advance the best requirement according to objective o
Delay_Worst_Ahead	With uniform probability, select a requirement, r, and an objective o. Delay the release of the worst requirement (according to objective o) and advance the release of r
Delay_Worst_Ahead_Best	With uniform probability, select an objective o. Delay the release of the worst requirement and advance the release of the best ('worst' and 'best' according to objective o)

Our hyper-heuristic release planners use the scheme proposed by Fialho *et al.* [Fialho et al. 2008], known as 'extreme value credit assignment', which is based on the principle that infrequent, yet large, improvements in the objective score are likely to be more effective than frequent, small improvements.

Therefore, it rewards operators that have had a recent large positive impact on the objective score, whilst consistent operators that only yield small improvements receive less credit, and therefore have lower chances of selection.

The impact on the objective score is measured by the fitness improvement, concerning the quality (fitness value) of the offspring with those of its parents [Fialho et al. 2008]. In the case of multi-objective hyper-heuristic release planning, there are multiple fitness values evaluated due to the multi-objective fitness functions. In this work, the fitness improvement is calculated by the fitness hypervolume differences between the offspring and its parents. Fitness hypervolume is the volume covered by the solutions in the objective space [Zitzler and Thiele 1999].

Formally, An estimate of the current operator k credit is denoted as $\hat{q}_{k,t}$. The current fitness improvement is added to a window in a First In First Out (FIFO) manner. The size of window is W. The fitness improvement observed at each iteration t is denoted as $\Delta(t)$. t_i denote the time iteration where operator k was seleted. The best (maximal) fitness improvement observed in the sliding window is rewarded for operator k. That is, the credit assigned for operator k is

calculated as:

$$\hat{q}_{k,t} = argmax\{\Delta(t_i), i = 1, \ldots, W\} \quad (4)$$

The credit assignment mechanism needs to be coupled with a selection mechanism that uses the accumulated credits to select the operator to apply in the current iteration. Most operator selection rules in the literature attach a probability to each operator and implement a randomised process to select the operator according to these probabilities. We used the simplest of these rules, called Probability Matching [Thierens 2005], which corresponds to the well-known roulette wheel selection used by meta-heuristic SBSE work [Harman et al. 2012b].

More formally, let K denote the number of search operators. The Probability Matching selection mechanism maintains a probability vector that is updated at each iteration t, $(p_{k,t})_{k=1,\ldots,K}$. The goal is to make $p_{k,t}$ proportional to $\hat{q}_{k,t}$.

An operator that performs poorly during a long period of the search will have its quality estimate decreased to a low value (possibly even zero). To avoid such operators becoming completely ignored (which may be undesirable since it would reduce diversity), we assign a minimal selection probability $p_{min} > 0$. Equation 5 describes the Probability Matching rule we used:

$$p_{k,t+1} = p_{min} + (1 - K * p_{min}) \frac{\hat{q}_{k,t+1}}{\sum_{k=1}^{K} \hat{q}_{k,t+1}} \quad (5)$$

4. EXPERIMENTAL SET UP

This section explains our experimental methodology. We motivate the choice of algorithms studied (Section 4.1); the datasets to which we apply these algorithms (Section 4.2); the metrics we used to assess quality, diversity and speed (Section 4.3); the inferential statistical techniques used to assess differences and correlations (Section 4.4); and the research questions we answer (Section 4.5).

4.1. Algorithms

More than 15 different meta-heuristic algorithms have been used in SBSE research [Harman et al. 2012b]. Many of these have also been used in research on the NRP/RP (as outlined in Table I). Indeed, all of these 15 (and many more meta-heuristics [Burke and Kendall 2005]) could be used, in principle, since the formulation of the problem is sufficiently generic that any search based technique could be applied.

We chose to investigate the performance of six search techniques: three meta-heuristics, Hill Climbing, Simulated Annealing and the Nondominated Sorting Genetic algorithm (NSGA-II), which we denote HC, SA and GA respectively, together with hyper-heuristic versions of each of the three meta-heuristics, which we denote HHC, HSA and HGA. The motivation for this choice derives from the way in which computational search algorithms can be classified as either 'local' or 'global'. Local search techniques, such as HC, tend to be fast, but they can become stuck in a local optimum, thereby producing sub-optimal solutions. By contrast, global search techniques, such as genetic algorithms, may be computationally more expensive, but they incorporate mechanisms to avoid local optima confinement. In this study, HC is selected as a representative for local search algorithms.

Many techniques embody elements of both local and global search, with the local search facilitating 'exploitation' in the search landscape, while the global search facilitates 'exploration' of the landscape [Crepinsek et al. 2013]. One widely studied algorithm that does this is SA, which augments the basic HC approach with a 'cooling' coefficient that mimics cooling in annealing processes [van Laarhoven and Aarts 1987]. This process can enable the algorithm to escape local optima. The probability that the search will accept and less fit neighbouring solution is directly proportional to the temperature. This temperature decreases as the simu-

Table III. Overview of meta- and hyper-heuristic algorithms applied

Algorithm	Mechanism	Move Strategy in the search space	Operator
R	Random search	Random (Uniform distribution)	-
HC	Local search	Neighbourhood-based	Random generating neighbourhood
HHC			10 Search operators (Table II) generating neighbourhood
SA		Neighbourhood-based & Temperature controlling	Random generating neighbourhood
HSA			10 Search operators (Table II) generating neighbourhood
GA	Global search	Population & Sorting-based	Single-point Crossover & Uniform Mutation
HGA			Single-point Crossover & 10 Search operators (Table II)

lated annealing process progresses until, in the limit, the search becomes a pure hill climb. In the SBSE literature, SA is the most widely used compromise between global and local search [Harman et al. 2012b]. Therefore, SA is chosen to represent a local search with a "global strategy".

The NSGA-II algorithm (denoted as GA in the paper) is selected since it is the most widely used and most effective global algorithm for the multi-objective optimisation problems. We also use purely Random Search (R) to provide a baseline against which to benchmark more 'intelligent' search techniques.

Our choice of the three meta-heuristic algorithms reflects our desire to sample from the set of possible algorithm choices, three that, in some sense, 'cover' the spectrum of algorithmic behaviours from local to global search. As Table I shows, all three of these meta-heuristic search techniques have been proposed and studied for release planning problems. We also wish to study the effect of hyper-heuristics as well as meta-heuristics, motivating our choice of the three hyper-heuristic variants of the three meta-heuristics we selected for study. Table III provides an overview of meta- and hyper-heuristic algorithms used in the study.

In the hyper-heuristic algorithms (HGA, HSA and HHC), the mutation operator of the meta-heuristic version (GA, SA and HC respectively) is replaced by the adaptive operator selection mechanism outlined in Section 3.2. That is, HGA, HSA and HHC search the space of the permutation of 10 search operators described in Table II. In the meta-heuristic algorithms, the mutation operator mutates each gene of a chromosome with a certain mutation probability. The hyper-heuristic algorithms adaptively choose different search operators based on credit assignment feedback and Probability Matching selection mechanism in the search process. The sequences of search operators generated are the heuristics to mutate the chromosomes in the solution space, thereby generating the new offsprings of solution chromosomes. The length of sequences of search operators can be fixed or adaptive. The length of operator sequences is fixed to 1 in the experiment. That is, one operator is applied to mutate a chromosome at each time with a certain probability. For SA, HC, HSA and HHC, a random seeding is used for generating the initial population. 10% of genes in a chromosome are randomly selected and mutated to generate the neighbourhood in SA and HC. For the GA and HGA, we set the population size to 100 and the number of generations to 50. Single-point crossover is used for GA and HGA. The crossover probability is set to $P_c = 0.8$ and mutation probability to $P_m = 1/n$ (where n is the chromosome length of GA) is used. To ensure a fair comparison, SA, HSA, HC, HHC and R were all given the same number of fitness evaluations as GA and HGA. The non-dominated solutions in the population based on multiple objectives are selected as the best solutions for each algorithm.

4.2. Datasets

We used 10 datasets, of which 7 are drawn from real world requirements selection problems in a variety of different organisations. Of these, 5 have been used in separate previous studies in the literature and 2 (StoneGate and MS Word) are newly introduced for the first time here. The other 3 datasets contain bug fixes requested for Eclipse, Mozilla and Gnome. They

Name and Source of Dataset	Number of R	SH	Objectives Max	Min	Summary Description of Dataset and Software System
Baan [Van den Akker et al. 2008]	100	17	Revenue	Cost	ERP product developed by 600 engineers spread over four countries
StoneGate	100	91	Sales Value	Impact	Industrial software security release planning project (confidential source)
Motorola [Zhang et al. 2011]	35	4	Revenue	Cost	UK service provider requirements for range of handheld communication devices
RalicP [Zhang et al. 2013]	143	77	Revenue	Cost	Library and ID Card System in current use at University College London (UCL)
RalicR [Zhang et al. 2013]	143	79	Revenue	Cost	Library and ID Card System in current use at UCL (a variant of RalicP)
Ericsson [Zhang et al. 2010]	124	14	Importance (for today & the future)	Cost	Requirements for a software testing tool for now and into the future
MS Word	50	4	Revenue	Urgency	Text editing system for use with Microsoft Word
Eclipse [Xuan et al. 2012]	3502	536	Importance	Cost	The Eclipse environment with bug fix requests treated as requirements
Mozilla [Xuan et al. 2012]	4060	768	Importance	Cost	The Mozilla system with bug fix requests treated as requirements
Gnome [Xuan et al. 2012]	2690	445	Importance	Cost	The Gnome desktop system with bug fix requests treated as requirements

Fig. 1. The 10 datasets used and their numbers of Requirements (R), stakeholders (SH) and objectives to be Maximised (Max) and Minimised (Min). Those datasets with accompanying citations are taken from previous studies; those without citations are used in this paper for the first time.

might be regarded as 'pseudo real world'; they are taken from real world applications but it is debatable whether they truly denote 'requirements'.

We include these three in the study since they have previously been used to act as a surrogates for real world datasets. Their use in previous work was motivated by the need to overcome the difficulty of finding sufficiently many real world datasets on which to evaluate [Xuan et al. 2012]. A summary of each dataset studied in this paper can be found in Figure 1.

Previous studies have included at most two real world requirements datasets (or all three of the bug fixing pseudo-real world datasets), often augmenting these with synthetic data to compensate for the lack of real world data. Our study is therefore the most comprehensive study of meta-heuristics for release planning so far reported in the literature. Our use of these 10 datasets is sufficient to allow us to ask an important research question that has, hitherto, eluded the research community: 'how well do the algorithms scale with respect to the size of the real world requirements problem to which they are applied?'.

Baan dataset is extracted from the 5.2 release plan of an ERP product developed by about 600 software engineers and staff located in the Netherlands, India, Germany and the USA. StoneGate is a dataset stemming an industrial software security release planning project with 100 feature, including 91 resources and 5 different resource types considered for planning. Motorola dataset concerns a set of requirements for hand held communication devices and the stakeholders are mobile service providers in the UK. RalicP and RalicR datasets are extracted from RALIC (Replacement Access, Library and ID Card) project at University College London, which initiated to replace the existing access control system at UCL and consolidate the new system with library access and borrowing. According to two different requirements priority measurements, RalicP and RalicR datasets were collected. Ericsson dataset is extracted from questionnaire forms for test management tools from Ericsson, which were completed by the groups of Ericsson software engineers to measure how important each requirement is. RP Benchmark dataset is a (synthetic) benchmark problem and included 198 features, 8 stakeholders and 30 different resource types. MS Word data set is for planning 50 features considered for release planning of a text editing system. Eclipse, Mozilla and Gnome datasets are the RP instances of the bug repositories for Eclipse (which is an integrated development envi-

ronment (IDE)), Mozilla (which is an open source project including a set of web applications) and Gnome (which is a desktop environment and development platform).

When reviewing the datasets, we speculated that 'revenue' and 'sales value' were likely to be similar objectives. However, they come from different datasets which originated from different organisations, so it is likely that different terminology and possibly slightly different definitions would be pertinent to each. From an optimisation point of view, all that matters is to have a quantification of each objective, since this is the input to the search.

The choice of the objectives to be considered in any multi-objective NRP/RP instance is governed by the specifics of the dataset and scenario for which the search based optimiser seeks requirement sets. In the 10 datasets used in this paper the objectives are to maximize our *Revenue*, *Sales Value* and *Importance*, while those to be minimised are *Cost* and effect on *Urgency*. All that we require the dataset is the identification of the objective is to be minimised and maximised. The estimates of requirements are provided by the stakeholders of projects. Figure 1 provides descriptive statistics that characterise the datasets, their sources and the requirement optimisation objectives pertinent to each dataset.

In all but one case, the problem is a bi-objective problem in which there is a single objective to be maximised (such as revenue) and a single objective to be minimised (such as cost). The exception is the Ericsson dataset. It has two 'importance' objectives to be maximised: one for the present and one for the future. The third objective is to minimise the cost. Ericsson dataset includes questionnaire forms for test management tools, which were completed by 14 stakeholders (each stakeholder was a software testing sub-organisation within Ericsson) [Zhang et al. 2010]. To complete the questionnaires, the 14 stakeholders measured how important each requirement is to them in two ways. One is to evaluate the degree of importance for today, the other is the importance for the future. This approach was adopted by Ericsson and not suggested by the authors. Each measurement was graded using four levels: ignore, low, medium or high. However, the quality of these estimates might be inaccurate or uncertain, thereby leading to one possible construct validity issue. The issue will be discussed in Section 6.

4.3. Performance Metrics

We use 4 quality metrics to compare the performance of each of the 6 search-based optimisation algorithms (and random search). In most multi-objective optimisation problems the globally optimal Pareto front is unobtainable. Release planning is no exception to this. In such situations it is customary to construct a 'reference' front. The reference front is defined to be the largest nondominated subset of the union of solutions from all algorithms studied. As such, the reference front represents the best current approximation available to the true location of the globally optimal Pareto front. Three of the quality metrics we use (Contribution, Unique Contribution and Convergence) are computed in terms of each algorithm's distance from or contribution to this reference front:

Contribution (denoted '*Contrib*' in our results tables) for algorithm A is the number of solutions produced by A that lie on the reference front. This is the simplest (and most intuitive) quality metric. It assesses how many of the best solutions found overall are found by algorithm A.

Unique Contribution (denoted '*UContrib*' in our results tables) for algorithm A is the number of solutions produced by A that lie on the reference front and are not produced by any algorithm under study except A. This is a variant of the 'Contribution' metric that takes account of the fact that an algorithm may contribute relatively few of the best solutions found, but may still be valuable if it contributes a set of unique best solutions that no other algorithm finds.

Convergence (denoted '*Conv*' in our results tables) for algorithm A is the Euclidean distance between the Pareto front produced by A and the reference front. More formally, $C = \frac{\sum_{i=1}^{N} d_i}{N}$, where *N* is the number of solutions obtained by an algorithm and d_i is the smallest Euclidean distance of each solution i to the reference Pareto front. The smaller the calculated value of *C*, the better the convergence. This metric $C = 0$ if the obtained solutions are exactly on the reference Pareto front. In this paper, the reported value of *Conv* is normalised and maximised $Conv = 1 - normalised(C)$, so the higher numbers of *Conv* mean better convergence.

Hypervolume (denoted '*HVol*' in our results tables) is the volume covered by the solutions in the objective space. *HVol* is the union of hypercubes of solutions on the Pareto front [Zitzler and Thiele 1999]. For each solution i, a hypercube v_i is formed with the solution i and a reference point R. The reference point is usually a vector of the worst fitness values. More formally, $HVol = volume(\bigcup_{i=1}^{N} v_i)$. The larger the value of *HVol*, the better the hypervolume. The normalised fitness values are used for calculating *HVol*. By using a volume rather than a count (as used by the 'contribution' metrics), this measure is less susceptible to bias when the numbers of points on the two compared fronts are very different.

Quality of solutions is clearly important, but diversity is also an important secondary criterion for algorithms that exhibit acceptable solution quality. We measured the diversity using a standard metric introduced by Deb [Deb 2001]:

Diversity measures the extent of distribution in the obtained solutions and spread achieved between approximated solutions and the reference front [Deb 2001]. $\Delta = \frac{\sum_{k=1}^{M} d_k + \sum_{j=1}^{N-1} |d_j - \overline{d}|}{\sum_{k=1}^{M} d_k + (N-1)\overline{d}}$, where $k(1 \leq k \leq M)$ is the number of objectives for a multi-objective algorithm. d_k is the Euclidean distance to the extreme solutions of the reference Pareto front in the objective space. *N* denotes the number of solutions obtained by an algorithm. $d_j(1 \leq j \leq N-1)$ is the Euclidean distance between consecutive solutions, $\overline{d}$ is the average of all the distance d_j. The smaller the value of Δ, the better the diversity.

Finally, in order to assess the compensation or effort required to produce the quality and diversity of solutions observed, we measure the computational effort:

Speed is measured in terms of the wall clock time required to produce the solutions reported, averaged over 30 executions. All experiments were carried out on a desktop computer with a 6 core 3.2GHz Intel CPU and 8GB memory.

In order to facilitate a more easy comparison of the six overall metrics used in our study, we normalise all of them to lie between 0.0 and 1.0 and convert all of them to 'maximising metrics' (such that higher values denote superior performance). For example, 'speed' (to give it a name that captures it 'maximising form') is computed as $1 - T$, where T is the normalised wall clock time. Thus, in all tables of data presented in this paper (including the correlation analyses) the reader can safely assume 'higher means better'. We normalise a value x, drawn from a set of observed values, ranging from x_{min} to x_{max}, using the standard normalising equation: $\frac{x - x_{min}}{x_{max} - x_{min}}$.

Our algorithms are executed 30 times each to cater for their stochastic natures, so the normalised metric values reported are averaged (using mean and median) over these 30 runs. Averaging means that there is often no value reported in our results that happens to be exactly 1.0 or exactly 0.0, despite normalisation using maxima and minima.

4.4. Statistical Testing

The selection of appropriate statistical techniques that provide robust answers to the research questions we seek to address is vital to the construct validity of our investigation. Therefore, we explain and motivate the statistical testing techniques used to investigate the research questions in our study. We need to take account of the stochastic nature of each algorithm

due to their partial reliance on randomisation. This is a well-known phenomenon for which it is widely advised [Arcuri and Briand 2011; Harman et al. 2012c] that inferential statistical testing should be used as an appropriate way to compare algorithm performance. The pseudo random number sequence used by the algorithms is the cause of uncertainty. We are therefore sampling over the population of pseudo random number sequences [Harman et al. 2012c].

We use inferential statistical testing techniques to draw inferences about the population of all possible executions of the algorithm on a particular instance, based on a sample of these executions. In our experiments we set our sample size to 30. That is, each of the 7 algorithms is executed 30 times on each of the 10 datasets. The null hypothesis is that all 7 algorithms have the same performance. Rejection of the null hypothesis can tell us whether the algorithms performance are significantly different to one another.

We had no knowledge of the distribution of the population from which we sample executions so we use nonparametric statistical techniques, thereby increasing the robustness of our statistical inferences [Arcuri and Briand 2011; Ferguson 1965; Harman et al. 2012c]. Many widely used nonparametric statistical techniques, such as, Mann-Whitney [Mann and Whitney 1947] (and closely-related Wilcoxon [Wilcoxon 1945]) test and the Kruskal-Wallis test [Kruskal and Wallis 1952], make fewer assumptions than parametric tests do, nevertheless assume that *variance is consistent* across all populations [Zimmerman 2000]. In our experiments, we can make no such assumption about our data.

Therefore, we use Cliff's method [Cliff 1996] for assessing statistical significance. Cliff's method is not only nonparametric, but it is also specifically designed for ordinal data. Our research questions are ordinal and we have no reason to believe that our measurement scales are anything but ordinal [Shepperd 1995]. Furthermore, Cliff's method makes no assumptions about the variance of the data, thereby making it more robust.

We use the Vargha-Delaney $\hat{A}_{12}$ metric for effect size (as recommended by Arcuri and Briand [Arcuri and Briand 2011]). Vargha-Delaney $\hat{A}_{12}$ also makes few assumptions and is highly intuitive: $\hat{A}_{12}(A, B)$ is simply the probability that algorithm A will outperform algorithm B in a head-to-head comparison.

Most statistical tests produce a test statistic, the value of which must exceed a certain threshold in order for the observed mean to lie outside the confidence interval defined by the experimenters' chosen α level. The α level is the experimenters' tolerance for Type I errors (the error of incorrectly rejecting the Null Hypothesis). Often, the test statistic computation is expressed as a so-called 'p value'; the value of which must be less than the chosen α level in order for the experimenter to claim 'statistical significance'. We have set our α level to the widely used 'standard' value of 0.05. Each comparison of a p value with the chosen α level is essentially a claim about probability; the probability of committing Type I error (the error of incorrectly rejecting the Null Hypothesis). However, if the experimenter conducts a series of tests, then the chances of committing a Type I error increase, potentially quite dramatically, unless some adjustment (or 'correction') is made.

One popular (but not necessarily ideal) adjustment is the Bonferroni correction, which was first used to control for multiple statistical inferences by Dunn [Dunn 1961]. More recent techniques have been developed that retain the property of the Bonferroni correction (avoiding Type I errors), while simultaneously reducing its tendency to increase Type II errors. In our work we use just such a technique, the Hochberg's method [Hochberg 1988] for controlling multiple hypothesis testing. This method ranks the statistical tests applied, adjusting the α level for each successive test. It is a less conservative adjustment in the Bonferroni correction, while retaining its ability to avoid Type I errors [Bejamini and Hochberg 1995].

Overall, given that we know little about the distribution of the populations of executions of the algorithms studied in this paper, we believe that the use of Cliff's method with the Hochberg correction provides the most robust and appropriate statistical testing available.

As well as investigating the quality of solutions produced by each algorithm, we also want to investigate the correlation between the size of the problem instance and the behaviour of each algorithm. Though we believe there may be correlations of interest, we have no reason to believe that they will be linear. Furthermore, since our data is measured on an ordinal scale, the use of the Pearson correlation [Galton 1889; Pearson 1895; Salkind 2007] is inappropriate; we need to choose an ordinal correlation method. In order to ensure robustness of our conclusions we chose to use both Kendall's τ [Kendall 1948] and Spearman rank correlation [Salkind 2007; Spearman 1904], both of which are nonparametric, rank-based assessments of correlation.

4.5. Research questions

This section explains and motivates the five research questions we ask in our study. When comparing different algorithms for release planning problems, a natural question to ask is the quality of the solutions produced, according to the standard measures of multi-objective solution quality. Our first research question therefore investigates solution quality:

RQ1: **Quality**: According to each of the 4 quality measures, and on each of the 10 datasets, which algorithm performs best?

To answer this question we use the Cliff's inferential statistical comparison, as explained in Section 4.4 to determine which algorithms significantly outperform others and the $\hat{A}_{12}$ measure of effect size in each case.

Quality of solutions is important, but from the release planner's point of view a wide diversity of candidate solutions may also be important. In the most extreme case, a degenerate Pareto front (containing only a single solution) may have maximum quality but will have no diversity and will thus offer the release planner no choice at all. We therefore also investigate the diversity of solutions produced by each algorithm:

RQ2: **Diversity**: What is the diversity of the solutions produced by each algorithm?

We used Cliff's method to report on statistically significant differences in Diversity and $\hat{A}_{12}$ to assess the effect size of any such differences observed.

Naturally, the computational effort required to produce these solutions is also important. An algorithm that produces slightly lower quality solutions, but which does so almost instantaneously will have different applications to one that produces better quality solutions, but takes several minutes to produce them. The former can be used in a situation where the release planner wants to repeatedly investigate 'what if' questions, rebalancing estimates of cost and value in real time. In this scenario, speed trumps quality, provided quality is sufficient to be acceptable for 'what if' analysis. The latter will be more useful in scenarios where requirements optimisation provides decision support for an important overall choice about release planning. In this situation, quality trumps speed, provided a solution can be found in reasonable time (which might be hours or even days).

RQ3: **Speed**: How fast can the algorithms produce the solutions?

An algorithm that produces good solutions with acceptable diversity in reasonable time for small problems may scale less well to larger problems. In release planning problems, scalability is not merely a question of the increased computational effort required for a larger problem; it is to be expected that computational effort will be directly proportional to problem size.

However, perhaps more importantly, there may also be some degradation solution quality and/or diversity as the size of the problem scales up. We therefore investigate scalability from the point of view of all of the metrics we collected in our answers to the foregoing three research questions.

RQ4: **Scalability**: What is the scalability of each of the algorithms with regard to solution quality, solution diversity and speed?

In order to answer this question we report the rank of correlation between the size of the problem (measured in terms of the number of requirements in the dataset) and each of the metrics for quality, diversity and speed.

Since each algorithm is executed 30 times to facilitate statistical comparisons, we report correlations between the number of requirements and each of the mean, median and best case for quality, diversity and speed. As explained in Section 4.4, we use Kendall's τ and Spearman rank correlation to assess the degree of correlation between the quality, diversity and speed metrics and the size of the problem (measured as the number of requirements).

RQ5: **Inclusion**: Is there any correlation between requirements inclusion in solutions on the Pareto front and requirement characteristics?

In order to aid decision making support and understand the characteristic of large solution sets, we investigate which attributes of a requirement are correlated with inclusion in Pareto optimal solutions. We calculate the number of inclusions on the Pareto front for each requirement over 30 executions of algorithms. We rank the requirements according to their inclusion and we use Kendall's τ to statistically describe the correlation between the inclusion ranking of a requirement and its attributes.

5. RESULTS AND ANALYSIS

This section presents the results of the empirical study of meta- and hyper-heuristic search techniques for multi-objective release planning.

RQ1: Quality: Table V presents the mean and median values of the metrics for quality, diversity and speed for each of the 7 algorithms on each of the 10 datasets. The results in the table are highlighted in dim grey, dark grey and light grey colours, which represent the best, the second and the third order in terms of the performance of the algorithms.

Table VI presents the results of the inferential statistical tests. Since we need to compare 7 different algorithms with each other, this yields 21 pairwise comparisons (and thus 21 columns of data). Each of these columns contains the Vargha-Delaney $\hat{A}_{12}$ effect size measure where the result is significant (at the 0.05 α level), and is left blank where the result is not significant. The column heading indicates which algorithm is being compared with which others, in groups of 6, 5, 4, 3, 2, and 1 pairwise comparisons (6+5+4+3+2+1 = 21 pairwise comparisons in total).

For example, in the pair of columns headed by the title $\frac{\text{HHC}}{\text{HC} \mid \text{R}}$, the HHC algorithm is compared against each of the HC and R algorithms. The value 1.00 is highlighted in dark grey and the values between 0.51 and 0.99 are highlighted in light grey. The value 1.00 in the first row under the first of these two columns indicates the following: HHC outperforms HC significantly for its contribution to the Baan dataset's reference front (because the entry is not blank) and the probability of this observation being made is 1.00. That is, HHC always beats HC for its contribution to the Baan dataset reference front in our sample of 30 runs. The fifth row of data in this same column contains the effect size measure 0.06, which being nonblank, indicates a significant result. However, this time the probability of HHC beating HC is 0.06, so the HC algorithm significantly outperforms the HHC algorithm on the metric assessed (Diversity for the Baan dataset).

From these two entries in the table of results we can see that, for the Baan dataset, HHC contributes far more strongly to the reference front than HC, but HC is far more diverse. As can be seen, the other three quality metrics for the comparison of HHC and HC on the Baan dataset also strongly favour HHC. Based on these observations we would prefer HHC to HC, since diversity is only interesting if the algorithm's quality is strong; a highly diverse set of sub-optimal solutions is easy to achieve and is of little value to the release planner.

The forgoing discussion indicates the density of information contained in Table VI. Some general observations do emerge: From Table V, we can see that SA, HC and R make few contributions to the reference front: R contributes in two cases, while the other two algorithms only manage a contribution in a single case: the Ericsson dataset. Even when these three search strategies do make a contribution to the reference front they contribute only a tiny proportion of solutions (no more than 2%).

Looking at the results for the three meta-heuristic algorithms (GA, SA and HC), we see that GA performs best overall for quality on smaller datasets, while SA performs noticeably better on the three larger datasets (Eclipse, Mozilla and Gnome). This highlights the risk of drawing conclusions based on too narrow a selection of real world datasets.

The results for the three hyper-heuristic algorithms are less equivocal; they outperform their meta-heuristic counterparts according to all 4 quality measures. That is, HSA and HHC each significantly outperform both SA and HC on all 10 datasets with high effect size in every case, while HGA significantly outperforms GA on 9 out of the 10 datasets with high effect size. HGA is beaten by its meta-heuristic counterpart only on the Ericsson dataset.

Both HGA and HSA show good performance on the three larger datasets. HGA clearly offers the best performance over all datasets, algorithms and quality metrics: It significantly outperforms the other algorithms in 85% (205/240) of the pairwise algorithm quality comparisons. However, though HGA performs strongest in terms of solution quality, it would be a mistake to conclude that is the *only* algorithm that should be used. HSA outperforms the HGA for 'contribution' in 4 of the 10 datasets and, perhaps more importantly, for '*unique* contribution' in three cases. Even HHC significantly outperforms HGA in terms of 'contribution' in one case.

We also observe further evidence that suggests that results from one dataset may not generalise to others. The most extreme example of this is the Ericsson dataset, for which all of the algorithms behaved very differently (when compared to each other) than they did for the other datasets. HSA and HHC have closely equivalent behaviours and outperform other algorithms in terms of quality metric. Several factors may have influence on the performance of the algorithms. One aspect is the Ericsson dataset is the only dataset with three objectives as *Importance for Today*, *Importance for the Future* and *Cost*. When the number of objectives increases, the performance of the algorithms might behaviour differently. For example, NSGA-II (denoted by GA here) has much worse performance when the number of objectives is greater than two. The other factor is the character of the Ericsson dataset itself. Based on the previous study [Zhang et al. 2010] on the dataset, the requirements' importance for today and for the future graded by the stakeholders have a strong positive correlation. That is, a solution in which *Importance for Today* does not differ greatly from *Importance for the Future*. Therefore, the narrowed search space might make local search algorithms (HSA and HHC) more effective than GA and HGA.

RQ2: Diversity: As might be expected, random search performs very well in terms of diversity. From Table VI we can see that it outperforms almost every other algorithm for almost every dataset and often does so significantly and with a large effect size.

However, we know from the answer to RQ1 that random search only contributes to the reference front for 2 of the 10 datasets, and even then it only contributes at most 1% of the unique solutions. We therefore conclude that the diversity exhibited by the random search is largely suboptimal; any solutions it offers (diverse or otherwise) are likely to be dominated by solutions found by one of the other algorithms (if not all of them).

Of the three hyper-heuristic algorithms (which were competitive for the quality metrics), HGA exhibits the best diversity. It significantly outperforms HSA in 9 of the 10 datasets and HHC in 8 of the 10. As with the quality metrics studied in answer to RQ1, we observe that the Ericsson dataset also produces very different behaviour in terms of Diversity. The NSGA-II algorithm (on which both GA and HGA algorithms are based) was designed to promote diver-

sity and so we might expect that it should perform best, both its meta- and hyper-heuristic versions.

Even the meta-heuristic version (GA) outperforms the hyper-heuristic versions of simulated annealing (HSA) and hill climbing (HHC) with respsect to Diversity for 9 of the 10 datasets. However, there is no evidence that it outperforms its own hyper-heuristic version (HGA) with respect to Diversity. That is, GA significantly outperforms HGA for one dataset (Ericsson), while HGA significantly outperforms GA on 2 datasets (Mozilla and Gnome). In all other cases neither significantly outperforms the other.

RQ3: Speed: One might expect that a random search would be fast, since it is such a simple algorithm. However, we find (quite surprisingly) that the speed of random search is worse than all other algorithms studied for the larger datasets. We studied these results further and found that the explanation lies in the cost of invalid solutions: As the problem scale increases, a randomly constructed solution to the release planning problem is increasingly likely to be invalid. For example, it is increasingly likely to contain gaps in the release plan. The computational effort of random search becomes dominated by repairing such invalid release plans as the problem scale increases.

By contrast, meta-heuristic and hyper-heuristic algorithms spend most of their time adapting existing release plans. This makes them more scalable than random search, even though they are more sophisticated. Interestingly, HGA is fastest overall: it significantly outperforms its rival in 70% (42/60) of the pairwise comparisons.

On the largest dataset, Mozilla, which has more than 4,000 requirements, each of the executions of random search took more than 13 minutes to complete, while each HGA execution took just over 3 minutes. Neither of these durations makes a big difference to the kind of release planning applications that could be undertaken.

For the smaller datasets (with fewer than 200 requirements), each HGA executions completed in fewer than 10 seconds (sometimes merely 1 or 2 seconds). This puts HGA tantalisingly close to the threshold at which it could be used to investigate 'what if' scenarios; the release planner could modify the available requirements and/or their attributes and explore the impact of such changes in real time.

RQ4: Scalability: Figure 2 highlights a scalability problem for meta-heuristic NSGA-II (denoted GA in the tables): as the number of requirements increase, GA's contributions to the reference front decrease. This observation remains consistent whether we measure the mean, median or the best performance of each algorithm and also holds whether we use Kendall's or Spearman's correlation.

Figure 2 also reveals a negative correlation between the number of requirements and convergence of meta-heuristic NSGA-II and the best performance of meta-heuristic NSGA-II for hypervolume. Taken together, these negative correlation results for meta-heuristic NSGA-II quality metrics suggest that the quality of solutions it produces tend to decrease as the problem size increases.

We also observe a slightly less strong, but positive, correlation between the number of requirements and diversity of solutions produced by meta-heuristic NSGA-II. This suggests meta-heuristic NSGA-II increases its diversity with scale. However, since its contribution and quality tend to decrease as the scale of the problem increases its diversity is of considerably lesser value; it is simply producing a wider range of increasing sub-optimal solutions as the problem scales.

Fortunately, the other algorithms found to perform well in answer to RQ1 do not exhibit any such evidence for negative correlation between problem size and solution quality. In particular, hyper-heuristic NSGA-II exhibits no such correlation. Even more encouraging for this algorithm, we find consistent evidence, across all six correlation values, that it *increases its diversity* as the scale of the problem increases.

We find that there is a positive correlation between $Speed$ and the number of requirements for a few algorithms, such as HSA and HHC.

"Speed" in Figure 2 is the $Median$, $Best$ and $Mean$ of the normalised values (in Table V) of all datasets, not the absolute wall clock time. Therefore, a positive correlation between the $Median$, $Best$ and $Mean$ normalised "Speed" and the number of requirements indicates that when the number of requirements increase, the rank of (speed) performance of one specific algorithm also increase. It means that such algorithm (such as HHC) has worse performance (lower speed rank) than others for the smaller datasets and better performance for the larger ones.

RQ5: Inclusion: Table IV shows the correlation between the attributes of requirements and their inclusion rankings. As illustrated in the table, the name given to these attributes in the 10 datasets are different. They are the names provided by the stakeholders, including *Revenue*, *Sales Value*, *Cost*, *importance* etc. that correspond to costs and values for the specific data set considered. Attribute A/B represents a requirement's A-to-B ratio.

The results of Kendall's tau (τ) correlation analysis reveal that, in general, the requirement's A-to-B ratios, such as R/C, S/I, IT/C, IF/C and R/U have strong correlation with its likelihood of inclusion for the smaller datasets. For Baan and Motorola datasets, requirement inclusion is also correlated with the *Cost* of requirements. For the large datasets, there is no correlation between the attributes of a requirement and its inclusion.

RQ5 attempts to help decision makers understand the large solution space and possible tension between the multiple objectives and attributes. Such analysis allows us to identify useful information about solution sets, how the solutions' composition and which requirements tend to be selected. In general, the requirements with higher A-to-B ratios have higher probability to be present on the Pareto front according to the correlation analysis. That is, the search process tends to balance multiple conflicting objectives, so such requirements are more favourable in the solution sets. What we can conclude from the results of large datasets is that the space search is considerably large to explore for finding the optimal solutions.

Actionable Findings: Our results are based on only 10 datasets. This is considerably larger than any previous empirical study of release planning. These datasets are obtained from open source as well as closed source, system tools as well as enterprise applications, and have sizes varying from 35 to 4,000 requirements. Nevertheless, it remains insufficient to generalise to every type of project in every scenario. Indeed, we have seen evidence in our results that algorithms can behave very differently with respect to different datasets.

Therefore, as with other experimental/empirical SBSE work [Harman et al. 2012a], this finding suggests that the use of synthetic datasets in experimental work on release planning should be augmented with the study of real world datasets most likely to share the characteristics of the problem domain to which the proposed algorithms are to be applied.

6. THREATS TO VALIDITY

We set out threats to potential validity, indicating how they might affect the degree to which it is possible to generalise the results. We also provide some mitigation strategies to eliminate these threats.

Construct Validity

In this work, the objects studied are sets of requirements, stakeholders and their attributes. Their attributes are the measurements associated with each requirement, which come from estimates from the stakeholders. Before the proposed techniques can be applied, the information of these estimates associated with requirements and stakholders must be at hand.

Our assumption is that an initial set of requirements has been collected and the stakeholders have been identified using a requirements elicitation process. The task of quantifying

Table IV. The Kendall Correlation between the Attributes of Requirement and its Ranking of Inclusion

Sets Data	Attributes	HGA		GA		HSA		SA		HHC		HC		R	
		τ	p-Value	τ	p-Value	τ	p-Value	τ	p-Value	τ	p-Value	τ	p-Value	τ	p-Value
Baan	*Revenue*	0.43	<0.00	0.47	<0.00	0.50	<0.00	0.45	<0.00	0.49	<0.00	0.45	<0.00	0.47	<0.00
	Cost	0.66	<0.00	0.62	<0.00	0.59	<0.00	0.61	<0.00	0.60	<0.00	0.62	<0.00	0.60	<0.00
	R/C	0.89	<0.00	0.90	<0.00	0.80	<0.00	0.77	<0.00	0.79	<0.00	0.79	<0.00	0.74	<0.00
Stone-Gate	*Sales Value*	0.26	<0.00	0.29	<0.00	0.24	<0.00	0.23	<0.00	0.18	0.01	0.24	<0.00	0.25	<0.00
	Impact	0.34	<0.00	0.32	<0.00	0.36	<0.00	0.32	<0.00	0.40	<0.00	0.28	<0.00	0.31	<0.00
	S/I	0.80	<0.00	0.85	<0.00	0.76	<0.00	0.68	<0.00	0.70	<0.00	0.63	<0.00	0.71	<0.00
Motorola	*Revenue*	0.05	0.66	0.00	1.00	0.06	0.63	0.05	0.65	0.07	0.57	0.07	0.54	0.11	0.37
	Cost	0.85	<0.00	0.81	<0.00	0.71	<0.00	0.77	<0.00	0.71	<0.00	0.75	<0.00	0.70	<0.00
	R/C	0.95	<0.00	0.97	<0.00	0.83	<0.00	0.90	<0.00	0.85	<0.00	0.90	<0.00	0.84	<0.00
RalicP	*Revenue*	0.57	<0.00	0.45	<0.00	0.37	<0.00	0.35	<0.00	0.37	<0.00	0.37	<0.00	031	<0.00
	Cost	0.19	<0.00	0.34	<0.00	0.40	<0.00	0.37	<0.00	0.36	<0.00	0.36	<0.00	0.40	<0.00
	R/C	0.65	<0.00	0.62	<0.00	0.50	<0.00	0.48	<0.00	0.49	<0.00	0.50	<0.00	0.43	<0.00
RalicR	*Revenue*	0.51	<0.00	0.47	<0.00	0.51	<0.00	0.38	<0.00	0.49	<0.00	0.41	<0.00	0.40	<0.00
	Cost	0.46	<0.00	0.51	<0.00	0.42	<0.00	0.48	<0.00	0.42	<0.00	0.50	<0.00	0.48	<0.00
	R/C	0.83	<0.00	0.85	<0.00	0.75	<0.00	0.64	<0.00	0.70	<0.00	0.68	<0.00	0.63	<0.00
Ericsson	*Importance for Today (IT)*	0.31	<0.00	0.46	<0.00	0.59	<0.00	0.49	<0.00	0.64	<0.00	0.49	<0.00	0.47	<0.00
	Importance for the Future (IF)	0.33	<0.00	0.48	<0.00	0.61	<0.00	0.49	<0.00	0.65	<0.00	0.50	<0.00	0.47	<0.00
	Cost	0.30	<0.00	0.45	<0.00	0.40	<0.00	0.51	<0.00	0.34	<0.00	0.50	<0.00	0.53	<0.00
	IT/C	0.43	<0.00	0.75	<0.00	0.78	<0.00	0.83	<0.00	0.74	<0.00	0.84	<0.00	0.87	<0.00
	IF/C	0.44	<0.00	0.75	<0.00	0.78	<0.00	0.84	<0.00	0.73	<0.00	0.83	<0.00	0.86	<0.00
MS Word	*Revenue*	0.34	<0.00	0.28	<0.00	0.27	<0.00	0.43	<0.00	0.27	<0.00	0.39	<0.00	0.42	<0.00
	Urgency	0.47	<0.00	0.42	<0.00	0.41	<0.00	0.56	<0.00	0.41	<0.00	0.51	<0.00	0.55	<0.00
	R/U	0.80	<0.00	0.82	<0.00	0.79	<0.00	0.67	<0.00	0.79	<0.00	0.70	<0.00	0.71	<0.00
Eclipse	*Importance*	0.13	<0.00	0.14	<0.00	0.14	<0.00	0.09	<0.00	0.09	<0.00	0.08	<0.00	0.07	<0.00
	Cost	0.12	<0.00	0.13	<0.00	0.13	<0.00	0.14	<0.00	0.11	<0.00	0.14	<0.00	0.07	<0.00
	I/C	0.17	<0.00	0.19	<0.00	0.17	<0.00	0.15	<0.00	0.13	<0.00	0.14	<0.00	0.09	<0.00
Mozilla	*Importance*	0.14	<0.00	0.13	<0.00	0.11	<0.00	0.08	<0.00	0.07	<0.00	0.07	<0.00	0.07	<0.00
	Cost	0.15	<0.00	0.13	<0.00	0.10	<0.00	0.11	<0.00	0.06	<0.00	0.08	<0.00	0.06	<0.00
	I/C	0.21	<0.00	0.18	<0.00	0.15	<0.00	0.13	<0.00	0.09	<0.00	0.10	<0.00	0.10	<0.00
Gnome	*Importance*	0.15	<0.00	0.18	<0.00	0.14	<0.00	0.10	<0.00	0.08	<0.00	0.07	<0.00	0.09	<0.00
	Cost	0.19	<0.00	0.23	<0.00	0.18	<0.00	0.14	<0.00	0.15	<0.00	0.14	<0.00	0.11	<0.00
	I/C	0.21	<0.00	0.25	<0.00	0.20	<0.00	0.14	<0.00	0.13	<0.00	0.12	<0.00	0.11	<0.00

requirements is usually regarded to be a challenging and hard problem in itself. The quantifying process may be time-consuming and the information collected may not be accurate. Nevertheless, there are still several feasible approaches in previous work that address this problem [Gilb 2005; Karlsson and Ryan 1997; Lim 2010]. The benefits of using quantified requirements include better support for budget estimates and feedback, and improved communication of the requirements.

The results of optimisation and analysis rely on the quality of these estimates. However, this might lead to one possible construct validity issue when the estimates are inaccurate or uncertain. To mitigate the impact of uncertainty, sensitivity analysis and uncertainty handling could be carried out during or after optimisation search process, which is one direction of our future work.

In addition, the release planing model used in this study does not take account of all the factors that are able to reflect every possible real-world scenario. Some simplifications have been introduced to make the model easy to tailor to a variety of real-world projects. For example, 8 of the 10 datasets we collected did not have the information of dependencies between requirements. Therefore, the requirement dependencies are not considered in the model. We also do not handle other hard and soft constraints in this study as well as the situation where the estimated costs change over time. However, providing such fine-grain RP model should be entirely achievable as the future work. Therefore, we argue that this study models the RP

problem at a reasonable level of granularity and the model can be extended and flexible to more detailed real-world scenarios.

Furthermore, as the global optimal Pareto front is unobtainable, we construct a reference front as a close and reasonable approximation of global optimal Pareto front. However, in theory, a global Pareto front could contain a large number (or even infinitely many) points. This might lead to one possible construct validity issue when an approximate solution only contain a limited number of them in practice.

Besides, we have attempted to control another potential threats to construct validity by the use of ordinal inferential statistical techniques which make no assumptions about distribution (including variance). For correlation analysis we use two different nonparametric correlations in order to increase confidence in our findings.

Internal Validity

The primary comparison tests for statistically significant results concerned the relative performance of the algorithm. The dependent variables measured are standard, widely used convergence, hypervolume, diversity and speed, and two assessments of each algorithm's contribution to the Pareto front. The findings suggest that hyper-heuristics are an attractive new direction for release planning optimisation and that, in particular, hyper-heuristic NSGA-II (the algorithm labelled 'HGA') is highly attractive: it typically outperforms the other algorithms studied in terms of quality, diversity *and* speed. Furthermore, it appears that it scales well compared to its meta-heuristic counterpart and other algorithms studied. Another potential threat to internal validity concerns the algorithmic parameter tuning. In this work, the results presented are based on the same parameter setting for all the algorithms. Performance of the algorithms could have been improved by individual fine tuning empirically or through systematic experimentation.

External Validity

The results presented are based on the study of 10 real-world datasets to support the claim, but it is initial evidence that hyperheuristics are able to dynamically select search operators for different instances and provide good-quality solutions to support decision making. It would be too strong to claim that it will generalise the results with other problem instances or datasets with different characteristics or structures.

7. CONCLUSION AND FUTURE WORK

We have presented a comprehensive study of meta-heuristic and hyper-heuristic release planning on 10 real world datasets. Overall, we found that hyper-heuristic NSGA-II performs the best in terms of quality, diversity, speed and scalability. However, our results also indicate that the hyper-heuristic versions of Simulated Annealing and Hill Climbing make some contribution to the best solutions found and are relatively scalable.

This finding suggests that if only a single algorithm is to be used then it should be hyper-heuristic NSGA-II, but if resources allow, it may be advantageous to combine its results with those from other hyper-heuristic algorithms.

Furthermore, we found that algorithm behaviour can differ greatly from one dataset to another, indicating that research on synthetic datasets needs to be augmented with analysis of appropriate real world datasets.

As future work we will provide a fine-grain RP model to consider dependencies between requirements, the different hard and soft constraints, and also try to handle the situation where the estimated costs of requirements change over time. To mitigate the impact of uncertainty of requirements' estimates, sensitivity analysis, uncertainty handling and risk awareness could be carried out during or after optimisation search process, which is one direction of our future work.

Regarding the algorithms, we will observe each operator's efficiency and effectiveness at different search stages. New search operators will be incorporated and we will investigate more about how the hyper-heuristic approach is learning in the adaptive operator selection process. We will also look at different parameter settings for the algorithm to scale to large problem instance.

In terms of release planning and search based requirements selection in general, overcoming the scalability challenges of the techniques in non-trivial software projects is critically important . Furthermore, the future work should also aim to manage different kinds of uncertainties, such as, requirements uncertainty, algorithmic uncertainty, uncertainty in resource constraints, uncertainty and dynamics in development time. Finally, more empirical investigations need to be carried out to assess the applicability of the proposed solutions in a real-world context.

Table V. The performance (Mean and Median) of the 7 algorithms for the 10 datasets. All metrics reported in this table are normalised and maximising so the reader can assume that 'higher numbers mean better performance' in a dataset. Unsurprisingly, the results show that Random search tends to produce low quality solutions. A little more surprisingly, the meta-heuristic algorithms (HC and SA) also contribute little to the best solutions found (as assessed by the metrics 'Contrib' and 'UContrib' in the table). The results also show that the Ericsson dataset occasions very different behaviour from the algorithms compared to the other datasets, indicating that the dataset studied really does matter in empirical studies of release planning.

Data Sets	Metrics		HGA		GA		HSA		SA		HHC		HC		R	
			Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median
Baan	Quality	Contrib	0.83	0.86	0.02	0.00	0.20	0.20	0.00	0.00	0.14	0.13	0.00	0.00	0.00	0.00
		UContrib	0.88	0.90	0.03	0.00	0.20	0.18	0.00	0.00	0.13	0.12	0.00	0.00	0.00	0.00
		Conv	0.98	0.98	0.79	0.79	0.78	0.79	0.26	0.26	0.71	0.71	0.19	0.19	0.19	0.20
		HVol	0.98	0.98	0.75	0.75	0.69	0.69	0.26	0.27	0.62	0.62	0.20	0.21	0.15	0.15
	Diversity		0.54	0.54	0.49	0.48	0.17	0.18	0.73	0.75	0.14	0.13	0.57	0.61	0.88	0.88
	Speed		0.94	0.95	0.96	0.96	0.37	0.36	0.78	0.78	0.15	0.16	0.67	0.70	0.98	0.98
Stone-Gate	Quality	Contrib	0.83	0.86	0.18	0.15	0.47	0.46	0.00	0.00	0.37	0.35	0.00	0.00	0.00	0.00
		UContrib	0.86	0.85	0.19	0.15	0.45	0.44	0.00	0.00	0.36	0.32	0.00	0.00	0.00	0.00
		Conv	0.96	0.96	0.67	0.68	0.71	0.70	0.16	0.19	0.68	0.68	0.14	0.14	0.21	0.21
		HVol	0.96	0.97	0.60	0.58	0.70	0.71	0.23	0.24	0.71	0.72	0.20	0.21	0.29	0.28
	Diversity		0.55	0.53	0.48	0.48	0.15	0.15	0.65	0.66	0.14	0.13	0.55	0.59	0.87	0.88
	Speed		0.99	0.99	0.99	0.99	0.39	0.39	0.86	0.89	0.16	0.16	0.85	0.85	0.99	0.99
Motorola	Quality	Contrib	0.81	0.80	0.46	0.47	0.39	0.39	0.00	0.00	0.38	0.39	0.00	0.00	0.00	0.00
		UContrib	0.85	0.86	0.49	0.48	0.32	0.31	0.00	0.00	0.31	0.30	0.00	0.00	0.00	0.00
		Conv	0.98	0.98	0.94	0.94	0.80	0.81	0.37	0.37	0.79	0.80	0.39	0.40	0.20	0.21
		HVol	0.97	0.97	0.91	0.92	0.75	0.74	0.29	0.29	0.74	0.74	0.34	0.33	0.08	0.07
	Diversity		0.26	0.23	0.29	0.29	0.28	0.29	0.53	0.53	0.31	0.30	0.45	0.48	0.70	0.69
	Speed		0.90	0.90	0.88	0.88	0.25	0.28	0.75	0.78	0.16	0.16	0.71	0.70	0.97	0.97
RalicP	Quality	Contrib	0.75	0.76	0.22	0.21	0.16	0.15	0.00	0.00	0.13	0.12	0.00	0.00	0.00	0.00
		UContrib	0.81	0.78	0.26	0.25	0.16	0.15	0.00	0.00	0.12	0.11	0.00	0.00	0.00	0.00
		Conv	0.96	0.96	0.85	0.84	0.60	0.61	0.24	0.23	0.55	0.55	0.19	0.18	0.19	0.18
		HVol	0.96	0.96	0.75	0.76	0.53	0.52	0.27	0.27	0.51	0.50	0.20	0.19	0.13	0.13
	Diversity		0.56	0.57	0.55	0.55	0.50	0.48	0.70	0.77	0.48	0.47	0.66	0.65	0.88	0.89
	Speed		0.95	0.96	0.98	0.98	0.25	0.27	0.79	0.82	0.17	0.17	0.76	0.79	0.96	0.96
RalicR	Quality	Contrib	0.84	0.88	0.10	0.07	0.12	0.12	0.00	0.00	0.08	0.08	0.00	0.00	0.00	0.00
		UContrib	0.78	0.75	0.10	0.07	0.10	0.10	0.00	0.00	0.07	0.07	0.00	0.00	0.00	0.00
		Conv	0.97	0.97	0.80	0.81	0.74	0.74	0.21	0.20	0.67	0.67	0.13	0.13	0.17	0.17
		HVol	0.97	0.97	0.71	0.73	0.68	0.68	0.24	0.25	0.61	0.62	0.17	0.18	0.11	0.11
	Diversity		0.39	0.42	0.39	0.40	0.18	0.18	0.50	0.54	0.17	0.18	0.47	0.49	0.78	0.76
	Speed		0.96	0.96	0.98	0.98	0.38	0.37	0.84	0.84	0.18	0.17	0.71	0.71	0.97	0.97
Ericsson	Quality	Contrib	0.00	0.00	0.01	0.01	0.96	0.96	0.02	0.02	0.97	0.97	0.02	0.02	0.01	0.01
		UContrib	0.00	0.00	0.01	0.01	0.94	0.94	0.02	0.02	0.95	0.97	0.02	0.02	0.01	0.01
		Conv	0.51	0.50	0.56	0.42	0.99	0.99	0.16	0.15	0.99	0.99	0.19	0.18	0.15	0.15
		HVol	0.30	0.41	0.33	0.40	0.96	0.96	0.75	0.74	0.95	0.93	0.76	0.76	0.79	0.80
	Diversity		0.44	0.45	0.87	0.88	0.77	0.75	0.64	0.84	0.75	0.77	0.64	0.74	0.85	0.81
	Speed		0.98	0.98	0.99	0.99	0.10	0.10	0.72	0.74	0.11	0.10	0.72	0.74	0.88	0.88
MS Word	Quality	Contrib	0.62	0.61	0.42	0.41	0.83	0.83	0.00	0.00	0.76	0.76	0.00	0.00	0.00	0.00
		UContrib	0.73	0.73	0.49	0.49	0.81	0.80	0.00	0.00	0.73	0.73	0.00	0.00	0.00	0.00
		Conv	0.96	0.97	0.88	0.89	0.82	0.82	0.22	0.20	0.82	0.82	0.27	0.28	0.20	0.21
		HVol	0.93	0.92	0.81	0.80	0.83	0.84	0.26	0.25	0.84	0.85	0.32	0.30	0.23	0.24
	Diversity		0.30	0.32	0.31	0.29	0.14	0.14	0.55	0.62	0.21	0.20	0.51	0.56	0.79	0.79
	Speed		0.98	0.98	0.97	0.97	0.27	0.26	0.86	0.90	0.12	0.13	0.84	0.85	0.99	0.99
Eclipse	Quality	Contrib	0.60	0.58	0.00	0.00	0.81	0.82	0.00	0.00	0.58	0.58	0.00	0.00	0.00	0.00
		UContrib	0.53	0.53	0.00	0.00	0.73	0.77	0.00	0.00	0.52	0.52	0.00	0.00	0.00	0.00
		Conv	0.98	0.99	0.59	0.61	0.92	0.92	0.68	0.68	0.92	0.92	0.63	0.68	0.75	0.75
		HVol	0.98	0.98	0.62	0.62	0.92	0.92	0.75	0.75	0.90	0.90	0.59	0.66	0.80	0.80
	Diversity		0.69	0.69	0.65	0.65	0.20	0.19	0.45	0.48	0.14	0.14	0.27	0.30	0.93	0.94
	Speed		0.95	0.95	0.99	0.98	0.79	0.80	0.96	0.96	0.79	0.80	0.93	0.94	0.05	0.05
Mozilla	Quality	Contrib	0.73	0.73	0.00	0.00	0.85	0.84	0.00	0.00	0.61	0.62	0.00	0.00	0.00	0.00
		UContrib	0.63	0.65	0.00	0.00	0.75	0.75	0.00	0.00	0.54	0.53	0.00	0.00	0.00	0.00
		Conv	0.97	0.97	0.70	0.72	0.92	0.93	0.72	0.72	0.92	0.92	0.64	0.69	0.77	0.77
		HVol	0.98	0.98	0.66	0.68	0.92	0.92	0.75	0.75	0.90	0.90	0.57	0.66	0.80	0.80
	Diversity		0.72	0.72	0.64	0.65	0.20	0.20	0.41	0.43	0.14	0.14	0.34	0.36	0.91	0.91
	Speed		0.96	0.96	0.99	0.99	0.83	0.83	0.97	0.97	0.83	0.83	0.95	0.96	0.06	0.06
Gnome	Quality	Contrib	0.61	0.64	0.00	0.00	0.84	0.84	0.00	0.00	0.56	0.55	0.00	0.00	0.01	0.01
		UContrib	0.48	0.49	0.00	0.00	0.68	0.67	0.00	0.00	0.46	0.45	0.00	0.00	0.01	0.01
		Conv	0.98	0.98	0.59	0.58	0.92	0.92	0.67	0.67	0.91	0.91	0.62	0.64	0.74	0.74
		HVol	0.97	0.97	0.61	0.61	0.92	0.92	0.74	0.74	0.89	0.90	0.65	0.70	0.80	0.80
	Diversity		0.72	0.70	0.68	0.68	0.21	0.22	0.42	0.47	0.15	0.15	0.42	0.45	0.95	0.94
	Speed		0.94	0.94	0.98	0.98	0.71	0.72	0.95	0.95	0.71	0.71	0.92	0.93	0.06	0.07

Table VI. Results of the statistical analysis. The numbers reported in this table are the Vargha-Delaney $\hat{A}_{12}$ metric for effect size, where the result is significant at the 0.05 α level according to Cliff's test. Where the entry is blank, the corresponding algorithmic comparison result is not significant. The value of $\hat{A}_{12}$ is the probability that the algorithm labelled at the head of the column outperforms the algorithms labelled immediately below. For example, in the pair of columns headed $\frac{\text{HHC}}{\text{HC} \mid \text{R}}$, the HHC algorithm is compared to each of the HC and R algorithms. Consider the first of this pair of columns, which compares HHC to HC. Where the $\hat{A}_{12}$ is greater than 0.5, this means that HHC (significantly) outperforms HC and where it is below 0.5, HC outperforms HHC. The closer $\hat{A}_{12}$ is to 1.0, the greater the probability that HHC would be expected to outperform HC; the closer to 0.0, the greater the probability that HC would be expected to outperform HHC.

Data Set	Metric		HGA						GA					HSA				SA			HHC		HC
			GA	HSA	SA	HHC	HC	R	HSA	SA	HHC	HC	R	SA	HHC	HC	R	HHC	HC	R	HC	R	R
Baan	Quality	Contrib	1.00	1.00	1.00	1.00	1.00	1.00	0.03	0.43	0.05	0.43	0.43	1.00	0.81	1.00	1.00	0.00			1.00	1.00	
		UContrib	1.00	1.00	1.00	1.00	1.00	1.00	0.03	0.43	0.06	0.43	0.43	1.00	0.80	1.00	1.00	0.00			1.00	1.00	
		Conv	1.00	1.00	1.00	1.00	1.00	1.00		1.00	0.93	1.00	1.00	1.00	0.90	1.00	1.00	0.00	0.75	0.81	1.00	1.00	
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.85	1.00	0.98	1.00	1.00	1.00	0.94	1.00	1.00	0.00	0.84	1.00	1.00	1.00	0.83
	Diversity			1.00	0.16	1.00		0.00	1.00	0.10	1.00	0.28	0.00	0.00		0.07	0.00	1.00	0.72	0.20	0.06	0.00	0.04
	Speed		0.12	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	1.00	0.02	0.00	0.99	0.06	0.00	1.00	0.77	0.00	0.00	0.00	0.00
Stone-Gate	Quality	Contrib	1.00	1.00	1.00	1.00	1.00	1.00	0.01	0.97	0.10	0.96	0.97	1.00	0.80	1.00	1.00	0.00			1.00	1.00	
		UContrib	1.00	1.00	1.00	1.00	1.00	1.00	0.06	0.96	0.16	0.96	0.97	1.00	0.75	1.00	1.00	0.00			1.00	1.00	
		Conv	1.00	1.00	1.00	1.00	1.00	1.00		1.00		1.00	1.00	1.00	0.77	1.00	1.00	0.00			1.00	1.00	0.20
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.12	1.00	0.11	1.00	1.00	1.00		1.00	1.00	0.00		0.25	1.00	1.00	0.12
	Diversity			1.00	0.27	1.00		0.02	1.00	0.11	1.00		0.00	0.01		0.03	0.00	0.99		0.05	0.02	0.00	0.02
	Speed			1.00	1.00	1.00	1.00	0.75	1.00	1.00	1.00	1.00		0.00	1.00	0.00	0.00	1.00		0.00	0.00	0.00	0.00
Motorola	Quality	Contrib	0.98	1.00	1.00	1.00	1.00	1.00	0.71	1.00	0.73	1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		UContrib	0.97	1.00	1.00	1.00	1.00	1.00	0.85	1.00	0.88	1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		Conv	0.98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00	0.00		0.93	1.00	1.00	0.95
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00	0.00	0.28	0.99	1.00	1.00	1.00
	Diversity				0.07		0.14	0.02		0.05		0.15	0.01	0.05		0.13	0.00	0.92		0.21	0.19	0.01	0.10
	Speed		0.91	1.00	0.98	1.00	0.96	0.00	1.00	0.95	1.00	0.94	0.00	0.00	0.78	0.00	0.00	1.00		0.00	0.00	0.00	0.00
RalicP	Quality	Contrib	0.99	1.00	1.00	1.00	1.00	1.00		1.00		1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		UContrib	0.99	1.00	1.00	1.00	1.00	1.00		1.00		1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		Conv	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.78	1.00	1.00	0.00			1.00	1.00	
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.71	1.00	1.00	0.00	0.79	0.99	1.00	1.00	0.79
	Diversity			0.74	0.15	0.77	0.22	0.00	0.74	0.12	0.79	0.19	0.00	0.12		0.13	0.00	0.89		0.12	0.12	0.00	0.02
	Speed		0.07	1.00	1.00	1.00	1.00	0.31	1.00	1.00	1.00	1.00	0.81	0.00	0.78	0.00	0.00	1.00		0.00	0.00	0.00	0.00
RalicR	Quality	Contrib	1.00	1.00	1.00	1.00	1.00	1.00		0.83		0.83	0.83	1.00	0.78	1.00	1.00	0.00			1.00	1.00	
		UContrib	1.00	1.00	1.00	1.00	1.00	1.00		0.83		0.83	0.83	1.00	0.74	1.00	1.00	0.00			1.00	1.00	
		Conv	1.00	1.00	1.00	1.00	1.00	1.00	0.88	1.00	0.99	1.00	1.00	1.00	0.91	1.00	1.00	0.00	0.78	0.71	1.00	1.00	
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.71	1.00	0.93	1.00	1.00	1.00	0.97	1.00	1.00	0.00	0.81	0.97	1.00	1.00	0.80
	Diversity			0.92		0.93		0.00	0.96		0.98		0.00	0.11		0.12	0.00	0.89		0.09	0.12	0.00	0.07
	Speed		0.09	1.00	1.00	1.00	0.97	0.18	1.00	1.00	1.00	0.97	0.77	0.00	0.99	0.04	0.00	1.00	0.81	0.00	0.00	0.00	0.03
Ericsson	Quality	Contrib	0.12	0.00	0.00	0.00	0.04		0.00	0.00	0.00	0.12		1.00		1.00	1.00	0.00		1.00	1.00	1.00	0.88
		UContrib	0.08	0.00	0.00	0.00	0.04		0.00	0.00	0.00	0.12		1.00		1.00	1.00	0.00		1.00	1.00	1.00	0.88
		Conv		0.00	1.00	0.00	0.92	1.00	0.00	1.00	0.00	0.88	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		HVol		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00		1.00	1.00	0.00		0.04	1.00	1.00	
	Diversity		0.00	0.00	0.20	0.00	0.24	0.00	0.92	0.84	0.96	0.76					0.20			0.28		0.20	0.28
	Speed		0.15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00		0.00	0.00	1.00		0.00	0.00	0.00	0.00
MS Word	Quality	Contrib	0.96	0.04	1.00	0.13	1.00	1.00	0.00	1.00	0.01	1.00	1.00	1.00	0.70	1.00	1.00	0.00			1.00	1.00	
		UContrib	0.96		1.00		1.00	1.00	0.01	1.00	0.07	1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	
		Conv	0.97	1.00	1.00	1.00	1.00	1.00	0.84	1.00	0.86	1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	0.75
		HVol	0.99	1.00	1.00	1.00	1.00	1.00		1.00		1.00	1.00	1.00		1.00	1.00	0.00			1.00	1.00	0.83
	Diversity			0.78	0.20		0.26	0.01	0.81	0.22		0.27	0.01	0.12	0.23	0.13	0.00	0.83		0.16	0.18	0.00	0.17
	Speed		1.00	1.00	1.00	1.00	1.00	0.12	1.00	1.00	1.00	1.00	0.00	0.00	0.99	0.00	0.00	1.00		0.00	0.00	0.00	0.00
Eclipse	Quality	Contrib	1.00	0.04	1.00		1.00	1.00	0.00		0.00		0.03	1.00	0.97	1.00	1.00	0.00		0.08	1.00	1.00	0.06
		UContrib	1.00	0.05	1.00		1.00	1.00	0.00		0.00		0.03	1.00	0.89	1.00	1.00	0.00			1.00	1.00	0.05
		Conv	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.19	0.00	0.30	0.02	1.00	0.84	1.00	1.00	0.00		0.03	1.00	1.00	0.02
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.03	0.00		0.00	1.00	0.95	1.00	1.00	0.00	0.94	0.00	1.00	1.00	0.00
	Diversity			1.00	0.98	1.00	1.00	0.00	1.00	0.98	1.00	1.00	0.00	0.02	0.94		0.00	1.00	0.83	0.00		0.00	0.00
	Speed		0.00	1.00	0.19	1.00	0.76	1.00	1.00	0.99	1.00	1.00	1.00	0.00		0.02	1.00	1.00	0.92	1.00	0.02	1.00	1.00
Mozilla	Quality	Contrib	1.00	0.18	1.00	0.79	1.00	1.00	0.00		0.00		0.00	1.00	0.98	1.00	1.00	0.00			1.00	1.00	0.06
		UContrib	1.00	0.21	1.00	0.78	1.00	1.00	0.00		0.00		0.00	1.00	0.87	1.00	1.00	0.00			1.00	1.00	0.04
		Conv	1.00	1.00	1.00	1.00	1.00	1.00	0.00		0.00		0.14	1.00	0.76	1.00	1.00	0.00	0.82	0.04	1.00	1.00	0.00
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.10	0.00		0.00	1.00	1.00	1.00	1.00	0.00	0.93	0.00	1.00	1.00	0.00
	Diversity		0.81	1.00	0.99	1.00	0.99	0.00	1.00	0.96	1.00	0.98	0.00	0.08	0.94	0.22	0.00	0.94		0.00	0.13	0.00	0.00
	Speed		0.00	1.00	0.06	1.00		1.00	1.00	0.99	1.00	1.00	1.00	0.00		0.00	1.00	1.00	0.91	1.00	0.00	1.00	1.00
Gnome	Quality	Contrib	1.00	0.08	1.00		1.00	1.00	0.00		0.00		0.04	1.00	0.97	1.00	1.00	0.00		0.08	1.00	1.00	0.03
		UContrib	1.00	0.09	1.00		1.00	1.00	0.00		0.00		0.03	1.00	0.86	1.00	1.00	0.00		0.07	1.00	1.00	0.02
		Conv	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.25	0.00		0.04	1.00	0.82	1.00	1.00	0.00		0.08	1.00	1.00	0.00
		HVol	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.07	0.00	0.32	0.00	1.00	0.97	1.00	1.00	0.00	0.94	0.00	1.00	1.00	0.00
	Diversity		0.69	1.00	0.99	1.00	0.99	0.00	1.00	0.98	1.00	0.99	0.00	0.12	0.93	0.10	0.00	0.90		0.00	0.07	0.00	0.00
	Speed		0.00	1.00	0.20	1.00	0.77	1.00	1.00	0.99	1.00	1.00	1.00	0.00		0.00	1.00	1.00	0.92	1.00	0.00	1.00	1.00

Alg.	**Median** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.21	**-0.52**	0.21	0.30	**0.66**	-0.21
GA	**-0.61**	**-0.56**	**-0.52**	-0.48	**0.61**	0.25
HSA	0.11	0.02	0.11	0.21	0.25	0.43
SA	-0.05	-0.05	0.30	0.43	-0.25	0.36
HHC	0.07	-0.02	0.07	0.11	-0.16	**0.57**
HC	-0.05	-0.05	0.11	0.16	-0.25	0.39
R	0.26	0.26	0.07	**0.52**	**0.71**	**-0.61**

Alg.	**Best** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.17	-0.44	0.05	0.05	**0.52**	-0.25
GA	**-0.57**	**-0.57**	**-0.61**	**-0.57**	0.43	**0.75**
HSA	0.32	0.27	0.11	0.21	-0.07	0.43
SA	0.17	0.12	0.30	0.25	**-0.61**	0.25
HHC	0.02	-0.02	0.07	0.16	-0.39	**0.57**
HC	-0.07	-0.07	0.21	0.11	-0.30	0.21
R	**0.71**	**0.66**	0.34	**0.57**	0.00	**-0.85**

Alg.	**Mean** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.25	-0.48	0.21	0.25	**0.71**	-0.25
GA	**-0.61**	**-0.57**	**-0.52**	-0.48	**0.66**	0.25
HSA	0.11	0.05	0.11	0.21	0.25	0.43
SA	0.02	-0.12	0.25	0.34	-0.39	0.43
HHC	-0.07	-0.02	0.07	0.11	-0.23	**0.57**
HC	-0.12	-0.02	0.07	0.16	-0.21	0.39
R	**0.61**	**0.61**	0.25	**0.52**	**0.66**	**-0.66**

Kendall correlation values

Alg.	**Median** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.28	**-0.63**	0.21	0.41	**0.83**	-0.18
GA	**-0.73**	**-0.71**	**-0.61**	-0.50	**0.73**	0.41
HSA	0.23	0.06	0.31	0.32	0.27	**0.63**
SA	-0.06	-0.06	0.52	0.62	-0.46	0.59
HHC	0.18	0.05	0.27	0.29	-0.17	**0.78**
HC	-0.06	-0.06	0.36	0.31	-0.46	0.62
R	0.31	0.31	0.32	**0.67**	**0.81**	**-0.78**

Alg.	**Best** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.22	-0.53	0.05	0.06	**0.65**	-0.18
GA	**-0.70**	**-0.70**	**-0.78**	**-0.69**	0.54	**0.87**
HSA	0.47	0.36	0.31	0.32	-0.17	**0.63**
SA	0.19	0.07	0.46	0.40	**-0.70**	0.40
HHC	0.05	0.05	0.30	0.31	-0.51	**0.79**
HC	-0.19	-0.19	0.43	0.34	-0.36	0.38
R	**0.82**	**0.79**	0.54	**0.66**	0.00	**-0.94**

Alg.	**Mean** value correlated with number of requirements					
	Quality				Diversity	Speed
	Contrib	UContrib	Conv	HVol		
HGA	-0.38	-0.60	0.21	0.35	**0.85**	-0.20
GA	**-0.80**	**-0.79**	-0.61	-0.50	**0.74**	0.38
HSA	0.23	-0.02	0.31	0.32	0.27	0.59
SA	0.04	-0.22	0.47	0.51	-0.59	0.62
HHC	0.02	0.05	0.27	0.29	-0.26	**0.78**
HC	-0.21	-0.17	0.34	0.31	-0.45	0.57
R	**0.74**	**0.74**	0.42	**0.67**	**0.77**	**-0.81**

Spearman correlation values

Fig. 2. Correlations of metrics for quality, diversity and speed with size of problem. Perhaps surprisingly, the speed of Random search (labelled 'R' in the table), widely believed to be 'fast but low quality' does not scale well (indicated by a negative correlation). Also we observe that while the quality of meta-heuristic NSGA-II (labelled 'GA') tends to degrade with size, the quality of hyper-heuristic NSGA-II (labelled 'HGA') does not.

Acknowledgements: We wish to express their gratitude to Barbara Kitchenham for her 2-day tutorial on 'Statistical Techniques' at UCL CREST in February 2014. Her tutorial and subsequent discussions with us on ordinal inferential statistical techniques greatly influenced our approach to the analysis of results in this paper.

REFERENCES

Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. 2014. A systematic literature review of software requirements prioritization research. *Information and Software Technology* 56, 6 (June 2014), 568–585. DOI:http://dx.doi.org/10.1016/j.infsof.2014.02.001

Ahmed Al-Emran, Dietmar Pfahl, and Günther Ruhe. 2010. *A Hybrid Method for Advanced Decision Support in Strategic Product Release Planning*. Technical Report 088/2010. University of Calgary.

Thamer AlBourae, Günther Ruhe, and Mahmood Moussavi. 2006. Lightweight Replanning of Software Product Releases. In *Proceedings of the 1st International Workshop on Software Product Management (IWSPM '06)*. IEEE, Minneapolis, MN, USA, 27–34. DOI:http://dx.doi.org/10.1109/IWSPM.2006.5

A. Amandeep, Günther Ruhe, and Mark Stanford. 2004. Intelligent Support for Software Release Planning. In *Proceedings of the 5th International Conference on Product Focused Software Process Improvement (PROFES '04) (LNCS)*, Vol. 3009. Springer, Kansai Science City, Japan, 248–262. DOI:http://dx.doi.org/10.1007/978-3-540-24659-6_18

Allysson Allex Araújo, Matheus Paixao, Italo Yeltsin, Altino Dantas, and Jerffeson Souza. 2017. An Architecture based on interactive optimization and machine learning applied to the next release problem. *Automated Software Engineering* 24, 3 (September 2017), 623–671. DOI:http://dx.doi.org/10.1007/s10515-016-0200-3

Andrea Arcuri and Lionel Briand. 2011. A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering. In 33^{rd} *International Conference on Software Engineering (ICSE'11)*. ACM, New York, NY, USA, 1–10.

A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley. 2001. The Next Release Problem. *Information and Software Technology* 43, 14 (December 2001), 883–890. DOI:http://dx.doi.org/10.1016/S0950-5849(01)00194-X

Yoav Bejamini and Yosef Hochberg. 1995. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal statistical Society (Series B)* 57, 1 (1995), 289–300.

Marcia Maria Albuquerque Brasil, Thiago Gomes Nepomuceno da Silva, Fabricio Gomes de Freitas, Jerffeson Teixeira de Souza, and Mariela Ines Cortes. 2012. A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements. In *Enterprise Information Systems*. Vol. 102. Springer, 300–314. DOI:http://dx.doi.org/10.1007/978-3-642-29958-2_20

Edmund Burke and Graham Kendall. 2005. *Search Methodologies. Introductory tutorials in optimization and decision support techniques*. Springer.

Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Ender Özcan Gabriela Ochoa, and Rong Qu. 2013. Hyper-heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society* (2013), 1695–1724. DOI:http://dx.doi.org/10.1057/jors.2013.71

Edmund K. Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R. Woodward. 2010. *Handbook of Metaheuristics*. International Series in Operations Research & Management Science, Vol. 146. Springer, Chapter A Classification of Hyper-heuristic Approaches, 449–468. Chapter 15.

Edmund K. Burke, J. Dario Landa Silva, and Eric Soubeiga. 2005. Multi-Objective Hyper-Heuristic Approaches for Space Allocation and Timetabling. In *Metaheuristics: Progress as Real Problem Solvers*, Toshihide Ibaraki, Koji Nonobe, and Mutsunori Yagiura (Eds.). Operations Research/Computer Science Interfaces Series, Vol. 32. Springer US, 129–158. DOI:http://dx.doi.org/10.1007/0-387-25383-1_6

Xinye Cai and Ou Wei. 2013. A Hybrid of Decomposition and Domination Based Evolutionary Algorithm for Multi-Objective Software Next Release Problem. In *Proceedings of the 10th IEEE International Conference on Control and Automation (ICCA '13)*.

Xinye Cai, Ou Wei, and Zhiqiu Huang. 2012. Evolutionary Approaches for Multi-Objective Next Release Problem. *Computing and Informatics* 31 (2012), 847–875.

José M. Chaves-González and Miguel A. Pérez-Toledano. 2015. Differential Evolution with Pareto Tournament for the Multi-objective Next Release Problem. *Appl. Math. Comput.* 252 (February 2015), 1–13. DOI:http://dx.doi.org/10.1016/j.amc.2014.11.093

Norman Cliff. 1996. *Ordinal Methods for Behavioral Data Analysis*. Lawrence Erlbaum Associates, Inc., New Jersey, USA.

Felipe Colares, Jerffeson Teixeira de Souza, Rafael Augusto Ferreira do Carmo, Clarindo Pádua, and Geraldo Robson Mateus. 2009. A New Approach to the Software Release Planning. In *Proceedings of the 23rd Brazilian Symposium on Software Engineering (SBES '09)*. IEEE, Fortaleza, Ceará, Brazil, 207–215. DOI:http://dx.doi.org/10.1109/SBES.2009.23

Matej Crepinsek, Shih-Hsi Liu, and Marjan Mernik. 2013. Exploration and exploitation in evolutionary algorithms: a survey. *Comput. Surveys* 45, 3 (June 2013), 35:1–35:33.

Jerffeson Teixeira de Souza, Camila Loiola Brito Maia, Thiago Ferreira, Rafael Augusto Ferreira do Carmo, and Marcia Brasil. 2011. An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements. In *Proceedings of the 3rd International Symposium on Search Based Software Engineering (SSBSE '11) (LNCS)*, Vol. 6956. Springer, Szeged, Hungary, 142–157. DOI:http://dx.doi.org/10.1007/978-3-642-23716-4_15

Kalyanmony Deb. 2001. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons.

Isabel María del Águila and José Del Sagrado. 2016. Three Steps Multiobjective Decision Process for Software Release Planning. *Complexity* 21, S1 (September/October 2016), 250–262. DOI:http://dx.doi.org/10.1002/cplx.21739

José Del Sagrado and Isabel María Del Águila. 2009. Ant Colony Optimization for Requirement Selection in Incremental Software Development. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*. IEEE, Cumberland Lodge, Windsor, UK.

José Del Sagrado, Isabel María Del Águila, and Francisco Javier Orellana. 2010. Ant Colony Optimization for the Next Release Problem – A Comparative Study.. In *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*. IEEE, Benevento, Italy, 67–76. DOI:http://dx.doi.org/10.1109/SSBSE.2010.18

José del Sagrado, Isabel María del Águila, and Francisco Javier Orellana. 2015. Multi-objective Ant Colony Optimization for Requirements Selection. *Empirical Software Engineering* 20, 3 (June 2015), 577–610. DOI:http://dx.doi.org/10.1007/s10664-013-9287-3

Thiago do Nascimento Ferreira and Jerffeson Teixeira de Souza. 2012. An ACO approach for the Next Release Problem with Dependency among Requirements. In *Proceedings of the 3rd Brazilian Workshop on Search-Based Software Engineering (WESB '12)*. Natal, RN, Brazil.

Olive Jean Dunn. 1961. Multiple Comparisons Among Means. *J. Amer. Statist. Assoc.* 56, 293 (1961).

Juan J. Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, and Antonio J. Nebro. 2011. A Study of the Bi-Objective Next Release Problem. *Empirical Software Engineering* 16, 1 (February 2011), 29–60. DOI:http://dx.doi.org/10.1007/s10664-010-9147-3

Juan J. Durillo, Yuanyuan Zhang, Enrique Alba, and Antonio J. Nebro. 2009. A Study of the Multi-Objective Next Release Problem. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*. IEEE, Cumberland Lodge, Windsor, UK, 49–58. DOI:http://dx.doi.org/10.1109/SSBSE.2009.21

Martin S. Feather, Steven L. Cornford, James D. Kiper, and Tim Menzies. 2006. Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation. In *Proceedings of the International Workshop on Requirements Engineering Visualization (REV '06)*. IEEE, Minnesota, USA, 10–10. DOI:http://dx.doi.org/10.1109/REV.2006.2

Martin S. Feather, James D. Kiper, and Selcuk Kalafat. 2004. Combining Heuristic Search, Visualization and Data Mining for Exploration of System Design Space. In *The International Council on Systems Engineering (INCOSE '04) - Proceedings of the 14th Annual International Symposium*. Toulouse, France.

Martin S. Feather and Tim Menzies. 2002. Converging on the Optimal Attainment of Requirements. In *Proceedings of the 10th IEEE International Conference on Requirements Engineering (RE '02)*. IEEE, Essen, Germany, 263–270. DOI:http://dx.doi.org/10.1109/ICRE.2002.1048537

George Andrew Ferguson. 1965. *Nonparametric Trend Analysis: A practical guide for research workers*. McGill Uniervsoty Press, Montréal, Canada.

A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. 2008. Extreme Value Based Adaptive Operator Selection. In *Parallel Problem Solving from Nature PPSN X (LNCS)*, Vol. 5199. Springer, 175–184.

Alvaro Fialho, Luis Da Costa, Marc Schoenauer, and Michèle Sebag. 2010. Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.* 60, 1-2 (2010), 25–64.

Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2008. "Fairness Analysis" in Requirements Assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE '08)*. IEEE, Barcelona, Catalunya, Spain, 115–124. DOI:http://dx.doi.org/10.1109/RE.2008.61

Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2009. A Search based Approach to Fairness Analysis in Requirement Assignments to Aid Negotiation, Mediation and Decision Making. *Requirements Engineering Journal (RE '08 Special Issue)* 14, 4 (December 2009), 231–245. DOI:http://dx.doi.org/10.1007/s00766-009-0075-y

Francis Galton. 1889. *Natural Inheritance*. Macmillan and Co., London, UK.

Tom Gilb. 2005. *Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering using Planguage*. Butterworth-Heinemann.

Mark Harman, Edmund Burke, John A. Clark, and Xin Yao. 2012a. Dynamic Adaptive Search Based Software Engineering. In 6^{th} *IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2012)*. Lund, Sweden, 1–8.

Mark Harman and Bryan F. Jones. 2001. Search-based Software Engineering. *Information and Software Technology* 43, 14 (December 2001), 833–839. DOI:http://dx.doi.org/10.1016/S0950-5849(01)00189-6

Mark Harman, S. Afshin Mansouri, and Yuanyuan Zhang. 2012b. Search-based Software Engineering: Trends, Techniques and Applications. *ACM Computing Surveys (CSUR)* 45, 1 (November 2012). DOI:http://dx.doi.org/10.1145/2379776.2379787

Mark Harman, Phil McMinn, Jerffeson Teixeira de Souza, and Shin Yoo. 2012c. Search Based Software Engineering: Techniques, Taxonomy, Tutorial. In *Empirical software engineering and verification: LASER 2009-2010*, Bertrand Meyer and Martin Nordio (Eds.). Springer, 1–59. LNCS 7007.

Yosef Hochberg. 1988. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika* 75, 4 (1988), 800–802.

John H. Holland. 1975. *Adaption in Natural and Artificial Systems*. MIT Press, Ann Arbor.

He Jiang, Jifeng Xuan, and Zhilei Ren. 2010a. Approximate Backbone based Multilevel Algorithm for Next Release Problem. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*. ACM, Portland, Oregon, USA, 1333–1340. DOI:http://dx.doi.org/10.1145/1830483.1830730

He Jiang, Jingyuan Zhang, Jifeng Xuan, Zhilei Re, and Yan Hu. 2010b. A Hybrid ACO Algorithm for the Next Release Problem. In *Proceedings of the 2nd International Conference on Software Engineering and Data Mining (SEDM '10)*. IEEE, Chengdu, China, 166–171.

Muhammad Rezaul Karim and Guenther Ruhe. 2014. Bi-Objective Genetic Search for Release Planning in Support of Themes. In *Proceedings of the 6th International Symposium on Search-Based Software Engineering (SSBSE '14) (Lecture Notes in Computer Science)*, Vol. 8636. Springer, Fortaleza, Brazil, 123–137. DOI:http://dx.doi.org/10.1007/978-3-319-09940-8_9

J. Karlsson and K. Ryan. 1997. A Cost-Value Approach for Prioritizing Requirements. *IEEE Software* 14, 5 (1997).

Maurice Kendall. 1948. *Rank correlation methods*. Charles Griffin and Company Limited, London, UK.

William Henry Kruskal and Wilson Allen Wallis. 1952. Use of Ranks in One-Criterion Variance Analysis. *J. Amer. Statist. Assoc.* 47, 260 (1952), 583–621.

A.Charan Kumari, K. Srinivas, and M.P. Gupta. 2012. Software Requirements Selection using Quantum-inspired Elitist Multi-objective Evolutionary Algorithm. In *Proceedings of International Conference on Advances in Engineering, Science and Management (ICAESM '12)*. IEEE, Nagapattinam, Tamil Nadu, India, 782–787.

Chen Li, Marjan Van den Akker, Sjaak Brinkkemper, and Guido Diepen. 2010. An Integrated Approach for Requirement Selection and Scheduling in Software Release Planning. *Requirements Engineering* 15, 4 (November 2010), 375–396. DOI:http://dx.doi.org/10.1007/s00766-010-0104-x

Lingbo Li, Mark Harman, Emmanuel Letier, and Yuanyuan Zhang. 2014. Robust Next Release Problem: Handling Uncertainty during Optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation (GECCO '14)*. ACM, Vancouver, Canada, 1247–1254. DOI:http://dx.doi.org/10.1145/2576768.2598334

Yan Li, Tao Yue, Shaukat Ali, and Li Zhang. 2017. Zen-ReqOptimizer: a search-based approach for requirements assignment optimization. *Empirical Software Engineering* 22, 1 (February 2017), 175–234. DOI:http://dx.doi.org/10.1007/s10664-015-9418-0

Soo Ling Lim. 2010. *Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation*. PhD Thesis. School of Computer Science and Engineering, University of New South Wales.

Henry Berthold Mann and Donald Ransom Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics* 18, 1 (1947), 50–60.

Kent McClymont and Edward C. Keedwell. 2011. Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11)*. ACM, New York, NY, USA, 2003–2010.

Gabriela Ochoa, Matthew Hyde, Tim Curtois, Jose A. Vazquez-Rodriguez, James Walker, Michel Gendreau, Graham Kendall, Barry McCollum, Andrew J. Parkes, Sanja Petrovic, and Edmund K. Burke. 2012. HyFlex: a benchmark framework for cross-domain heuristic search. In *Proceedings of the 12th European conference on Evolutionary Computation in Combinatorial Optimization, EvoCOP'12 (Lecture Notes in Computer Science)*, Vol. 7245. Springer-Verlag, 136–147.

Matheus Henrique Esteves Paixão and Jerffeson Teixeira de Souza. 2013a. A Recoverable Robust Approach for the Next Release Problem. In *Proceedings of the 5th International Symposium on Search Based Software Engineering (SSBSE '13) (LNCS)*, Vol. 8084. Springer, St. Petersburg, Russia, 172–187. DOI:http://dx.doi.org/10.1007/978-3-642-39742-4_14

Matheus Henrique Esteves Paixão and Jerffeson Teixeira de Souza. 2013b. A Scenario-based Robust Model for The Next Release Problem. In *Proceeding of The 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13)*. ACM, Amsterdam, The Netherlands, 1469–1476. DOI:http://dx.doi.org/10.1145/2463372.2463547

Matheus Henrique Esteves Paixão and Jerffeson Teixeira de Souza. 2015. A Robust Optimization Approach to the Next Release Problem in the Presence of Uncertainties. *Journal of Systems and Software* 103 (May 2015), 281–295. DOI:http://dx.doi.org/10.1016/j.jss.2014.09.039

Karl Pearson. 1895. Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London* 58 (June 1895), 240–242.

D. Pisinger and S. Ropke. 2007. A general heuristic for vehicle routing problems. *Computers and Operations Research* 34 (2007), 2403–2435.

A.M. Pitangueira, P Tonella, A. Susi, R.S.P. Maciel, and M. Barros. 2017. Minimizing the Stakeholder Dissatisfaction Risk in Requirement Selection for Next Release Problem. *Information and Software Technology* 87 (July 2017), 104–118. DOI:http://dx.doi.org/10.1016/j.infsof.2017.03.001

Antônio Mauricio Pitangueira, Rita Suzana P. Maciel, and Márcio de Oliveira Barros. 2015. Software Requirements Selection and Prioritization using SBSE Approaches: A Systematic Review and Mapping of the Literature. *Journal of Systems and Software* 103 (May 2015), 267–280. DOI:http://dx.doi.org/10.1016/j.jss.2014.09.038

Antonio Mauricio Pitangueira, Paolo Tonella, Angelo Susi, Rita Suzana Maciel, and Marcio Barros. 2016. Risk-Aware Multi-stakeholder Next Release Planning using Multi-objective Optimization. In *Proceedings of International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ '16)*. Springer, Gothenburg, Sweden, 3–18. DOI:http://dx.doi.org/10.1007/978-3-319-30282-9_1

Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. 2008. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`. (With contributions by J. R. Koza).

Outi Räihä. 2010. A survey on Search–Based Software Design. *Computer Science Review* 4, 4 (2010), 203–249.

Günther Ruhe. 2010. *Product Release Planning: Methods, Tools and Applications*. CRC Press. 339 pages.

Günther Ruhe and Des Greer. 2003. Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE '03)*. IEEE, Rome, Italy, 262–270. DOI:http://dx.doi.org/10.1109/ISESE.2003.1237987

Günther Ruhe and An Ngo-The. 2004. Hybrid Intelligence in Software Release Planning. *International Journal of Hybrid Intelligent Systems* 1, 1-2 (April 2004), 99–110.

Günther Ruhe and Moshood Omolade Saliu. 2005. The Art and Science of Software Release Planning. *IEEE Software* 22, 6 (November 2005), 47–53. DOI:http://dx.doi.org/10.1109/MS.2005.164

Moshood Omolade Saliu and Günther Ruhe. 2007. Bi-Objective Release Planning for Evolving Software Systems. In *Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*. ACM, Dubrovnik, Croatia, 105–114. DOI:http://dx.doi.org/10.1145/1287624.1287641

Omolade Saliu and Günther Ruhe. 2005. Supporting Software Release Planning Decisions for Evolving Systems. In *Proceedings of the 29th Annual IEEE/NASA on Software Engineering Workshop (SEW '05)*. IEEE, Greenbelt, Maryland, USA, 14–26. DOI:http://dx.doi.org/10.1109/SEW.2005.42

Neil J. Salkind. 2007. *Encyclopaedia of measurement and statistics*. Sage publications, Inc., California, USA.

Martin J. Shepperd. 1995. *Foundations of software measurement*. Prentice Hall.

Charles Edward Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology* 15, 1 (January 1904), 72–101.

Mikael Svahnberg, Tony Gorschek, Robert Feldt, Richard Torkar, Saad Bin Saleem, and Muhammad Usman Shafique. 2010. A Systematic Review on Strategic Release Planning Models. *Information and Software Technology* 52, 3 (March 2010), 237–248. DOI:http://dx.doi.org/10.1016/j.infsof.2009.11.006

Dirk Thierens. 2005. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO '05)*. ACM, New York, NY, USA, 1539–1546.

Paolo Tonella, Angelo Susi, and Francis Palma. 2013. Interactive Requirements Prioritization using a Genetic Algorithm. *Information and Software Technology* 55, 1 (January 2013), 173–187. DOI:http://dx.doi.org/10.1016/j.infsof.2012.07.003

J.M. Van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. 2005a. Determination of the Next Release of a Software Product: an Approach using Integer Linear Programming. In *Proceedings of the CAiSE'05 FORUM*. Porto, Portugal, 119–124.

Marjan Van den Akker, Sjaak Brinkkemper, Guido Diepen, and Johan Versendaal. 2005b. Flexible Release Planning using Integer Linear Programming. In *Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '05)*. Essener Informatik Beitrage, Porto, Portugal, 247–262.

Marjan Van den Akker, Sjaak Brinkkemper, Guido Diepen, and Johan Versendaal. 2008. Software Product Release Planning Through Optimization and What-if Analysis. *Information and Software Technology* 50, 1-2 (January 2008), 101–111. DOI:http://dx.doi.org/10.1016/j.infsof.2007.10.017

P. J. M. van Laarhoven and E. H. L. Aarts. 1987. *Simulated Annealing: Theory and Practice*. Kluwer Academic Publishers, Dordrecht, the Netherlands.

Nadarajen Veerapen, Gabriela Ochoa, Mark Harman, and Edmund K. Burke. 2015. An Integer Linear Programming approach to the single and bi-objective Next Release Problem. *Information and Software Technology* 65 (September 2015), 1–13.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin* 1, 6 (1945), 80–83.

Jifeng Xuan, He Jiang, Zhilei Ren, and Zhongxuan Luo. 2012. Solving the Large Scale Next Release Problem with a Backbone Based Multilevel Algorithm. *IEEE Transactions on Software Engineering* 38, 5 (Sept.-Oct. 2012), 1195–1212. DOI:http://dx.doi.org/10.1109/TSE.2011.92

Yuanyuan Zhang, Enrique Alba, Juan J. Durillo, Sigrid Eldh, and Mark Harman. 2010. Today/Future Importance Analysis. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*. ACM, Portland, USA, 1357–1364. DOI:http://dx.doi.org/10.1145/1830483.1830733

Yuanyuan Zhang and Mark Harman. 2010. Search Based Optimization of Requirements Interaction Management. In *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*. IEEE, Benevento, Italy, 47–56. DOI:http://dx.doi.org/10.1109/SSBSE.2010.16

Yuanyuan Zhang, Mark Harman, Anthony Finkelstein, and S. Afshin Mansouri. 2011. Comparing the Performance of Metaheuristics for the Analysis of Multi-stakeholder Tradeoffs in Re-

quirements Optimisation. *Information and Software Technology* 53, 7 (July 2011), 761–773. DOI:http://dx.doi.org/10.1016/j.infsof.2011.02.001

Yuanyuan Zhang, Mark Harman, and Soo Ling Lim. 2013. Empirical Evaluation of Search Based Requirements Interaction Management. *Information and Software Technology* 55, 1 (January 2013), 126–152. DOI:http://dx.doi.org/10.1016/j.infsof.2012.03.007

Yuanyuan Zhang, Mark Harman, and S. Afshin Mansouri. 2007. The Multi-Objective Next Release Problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*. ACM, London, UK, 1129–1137 (Best Paper Award). DOI:http://dx.doi.org/10.1145/1276958.1277179

Donald W. Zimmerman. 2000. Statistical significance levels of nonparametric tests biased by heterogeneous variances of treatment groups. *Journal of General Psychology* 127, 4 (October 2000), 354–364.

E. Zitzler and L. Thiele. 1999. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 3, 4 (Nov. 1999), 257–271.