UCL DEPARTMENT OF
COMPUTER SCIENCE

Research Note
RN/14/07

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

6 June 2014

Yuanyuan Zhang

Mark Harman

Gabriela Ochoa

Guenther Ruhe

Sjaak Brinkkemper

Abstract

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.

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

Y. Zhang[1], M. Harman[1], G. Ochoa[2], G. Ruhe[3] and S. Brinkkemper[4]
[1]UCL, UK; [2]University of Stirling, UK; [3]University of Calgary, Canada; [4]Utrecht University, Netherlands

ABSTRACT

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

D.2.1 [**Software Engineering**]: Requirements/Specifications

General Terms

Algorithms, Experimentation, Measurement

Keywords

Strategic Release Planning, Meta-Heuristics, Hyper-Heuristics

1. INTRODUCTION

Release planning is the problem of determining the sets of requirements that should be included in each release 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 release planners.

In order to help decision-makers navigate these complex solution spaces, meta-heuristic search has been widely studied as a candidate solution technique [5, 51, 61, 70]. This work has placed release planning within the general area of Search Based Software Engineering (SBSE) [35]. Table 1 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 1 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 [34], 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).

[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.

As a sanity check, recommended for SBSE work [4, 37], 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 [10] 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 a 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 and threats to validity. Section 6 concludes the paper.

2. THE CONTEXT OF OUR STUDY

Bagnall *et al.* [5] first suggested the term *Next Release Problem* and described various meta-heuristic optimisation algorithms for solving it. Feather and Menzies [27] were the first to use a real world dataset, but this dataset is no longer publicly available. Ruhe *et al.* [3, 51, 52, 55] introduced the software release planning process together with exact optimisation algorithms [1, 2, 50, 53] and meta-heuristics, such as genetic algorithms. Van den Akker *et al.* [44, 60, 61, 62] also studied exact approaches to single objective constrained requirements selection problems.

Zhang *et al.* [70] introduced the Multi-Objective Next Release Problem formulation as a Pareto optimal problem with a set of objectives. However, Feather *et al.* [25, 26] had previously used Simulated Annealing to construct a form of Pareto front for visualisation of choices. Also, at the same time, Saliu and Ruhe [54] introduced a multi-objective search-based optimisation to balance the tension between user-level and system-level requirements.

Table 1: Previous meta-heuristic algorithm studies

Paper Authors (Year)	Algorithm(s)	Dataset
Bagnall *et al.* [5] (2001)	Greedy, HC, SA	Synthetic
Feather & Menzies [27] (2002), Feather *et al.* [26] (2004), Feather *et al.* [25] (2006)	SA	NASA
Ruhe & Greer [51] (2003), Ruhe & Ngo-The[52] (2004), Amandeep *et al.*[3] (2004), Ruhe [50] (2010)	GA	Synthetic
Zhang *et al.* [70] (2007)	NSGA2, Pareto GA	Synthetic
Durillo *et al.* [24] (2009), Durillo *et al.* [23] (2011)	NSGA2, MOCell, PAES	Synthetic & Motorola
Colares *et al.* [15] (2009)	NSGA2	Synthetic
Finkelstein *et al.* [31] (2008), Zhang *et al.* [68] (2011)	NSGA2, Two-Archive	Motorola
Zhang & Harman [67] (2010), Zhang *et al.* [69] (2013)	NSGA2, Archive-based NSGA2	Synthetic & RALIC
Sagrado & Aguila [19] (2009), Sagrado *et al.*[20] (2010)	ACO, SA, GA	Synthetic
Jinag *et al.* [39] (2010), Xuan *et al.* [65] (2012)	Backbone Multilevel	Eclipse, Mozilla, Gnome
Zhang *et al.* [66] (2010)	NSGA2	Ericsson
Jinag *et al.* [40] (2010)	ACO, GRASP, SA	Synthetic
Kumari *et al.* [43] (2012)	QEMEA	Synthetic
Souza *et al.* [17] (2011), Ferrreira & Souza [21] (2012)	ACO	Synthetic
Cai *et al.* [13] (2012)	NSGA2, SPEA2	Synthetic
Brasil *et al.* [7] (2012)	NSGA2, MOCell	Synthetic
Cai & Wei [12] (2013)	MOEAs	Synthetic

Subsequently, Finkelstein *et al.* [32] 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.

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.

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.

2.1 Representation, Objectives and Fitness

In any approach to SBSE it is necessary to define the representation of the problem and the objectives, which define the fitness function(s) used to guide the search [36, 37]. 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.

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. The RP problem seeks a weighted assignment that optimises for a set of objectives. In this paper we use the three-release formulation, with weights 5,3 and 1 for the first, second and third releases respectively.

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. 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 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 [10] are search methodologies, more recently introduced to the optimisation literature than the meta-heuristics that have been enthusiastically adopted by the SBSE community [36, 49]. Hyper-heuristics differ from meta-heuristics because they search the space of heuristics rather than the space of solutions [11].

Hyper-heuristics encapsulate problem specific information using a pool of low-level heuristics (known as search operators). 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 [9, 46].

Though hyper-heuristics have been suggested as a future direction for SBSE [34], there has been no previous investigation of the potential of hyper-heuristics for any SBSE problem, so their performance on release planning has, hitherto, been unknown. 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.

We implemented 10 search operators for hyper-heuristic release planning. These are explained in Figure 2. 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 [48]. 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.

Table 2: 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)

3.1 Adaptive Operator Selection

An adaptive operator selection scheme consists of two components, called the 'credit assignment' mechanism and the 'selection' mechanism [30]. Credit assignment involves the attribution of credit (or reward) to the hyper-heuristic's operators, determined by their performance during the search process.

Our hyper-heuristic release planners use the scheme proposed by Fialho *et al.* [29], 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 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' [59], which corresponds to the well-known roulette wheel selection used by meta-heuristic SBSE work [36].

4. EXPERIMENTAL SET UP

This section explains our experimental methodology.

4.1 Algorithms

More than 15 different meta-heuristic algorithms have been used in SBSE research [36]. Many of these have also been used in research on the NRP/RP (as outlined in Table 1). Indeed, all of these 15 (and many more meta-heuristics [8]) 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 (NSGA2), 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 hill climbing, 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.

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 [16]. One widely studied algorithm that does this is Simulated Annealing, which augments the basic hill climbing approach with a 'cooling' coefficient that mimics cooling in annealing processes [63]. This process can enable the algorithm to escape local optima. In the SBSE literature, SA is the most widely used compromise between global and local search [36].

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 1 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.

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 3.1. For the GA and HGA we set the population size to 100 and the number of generations to 50. To ensure a fair comparison, SA, HSA, HC and HHC and the random search were all given the same budget of fitness evaluations as GA and HGA.

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 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 [65]. 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?'.

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.

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 [72]. 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 [18]:

Diversity measures the extent of distribution in the obtained solutions and spread achieved between approximated

Name and Source of Dataset	Number of R	SH	Objectives Max	Min	Summary Description of Dataset and Software System
Baan [62]	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 [68]	35	4	Revenue	Cost	UK service provider requirements for range of handheld communication devices
RalicP [69]	143	77	Revenue	Cost	Library and ID Card System in current use at University College London (UCL)
RalicR [69]	143	79	Revenue	Cost	Library and ID Card System in current use at UCL (a variant of RalicP)
Ericsson [66]	124	14	Importance	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 [65]	3502	536	Importance	Cost	The Eclipse environment with bug fix requests treated as requirements
Mozilla [65]	4060	768	Importance	Cost	The Mozilla system with bug fix requests treated as requirements
Gnome [65]	2690	445	Importance	Cost	The Gnome desktop system with bug fix requests treated as requirements

Figure 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.

solutions and the reference front [18].

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

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 [4, 37] 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 [37].

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.

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 [4, 28, 37]. However, many widely used nonparametric statistical techniques are not as robust as the researcher might hope.

For example, though the widely used Mann-Whitney [45] (and closely-related Wilcoxon [64]) test and the Kruskal-Wallis test [42], make fewer assumptions than parametric tests they do, nevertheless assume that that *variance is consistent* across all populations [71]. Unfortunately, we can make no such assumption about our data in these experiments.

Therefore, we use Cliff's method [14] 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 [57]. Furthermore, unlike other popular nonparametric inferential statistical techniques such as the Kruskal-Wallis and Wilcoxon-Mann-Whitney tests, 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 [4]). 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.

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 [22]. Unfortunately, this is been shown to be highly conservative; while it avoids Type I errors, it does so at the risk of introducing Type II errors (the error of incorrectly accepting the Null Hypothesis), thereby reducing statistical power.

Fortunately, 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 [38] 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 [6].

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 [33, 47, 56] 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 [41] and Spearman rank correlation [56, 58], both of which are nonparametric, rank-based assessments of correlation.

4.5 Research questions

This section explains and motivates the four 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).

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 3 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. Table 4 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 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 4. Space does not permit a further individual discussion of each of the 21 pairwise comparisons, but some general observations do emerge: From Table 3, we can see that the random search (R) and also HC and SA make few contributions to the reference front: Random search 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 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 datasset.

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.

RQ2: Diversity: As might be expected, random search performs very well in terms of diversity. From Table 4 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 NSGA2 algorithm (on which both GA and HGA algorithms are based) was designed to promote diversity 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: The tables in Figure 2 highlight a scalability problem for meta-heuristic NSGA2 (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 NSGA2 and the best performance of meta-heuristic NSGA2 for hypervolume. Taken together, these negative correlation results for meta-heuristic NSGA2 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 NSGA2. This suggests meta-heuristic NSGA2 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 con-

siderably 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 NSGA2 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.

Actionable Findings and Threats to Validity: 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 100 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 [34], 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.

We have attempted to control 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. These findings suggest that hyper-heuristics are an attractive new direction for release planning optimisation and that, in particular, hyper-heuristic NSGA2 (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.

6. CONCLUSION

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 NSGA2 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 also make some contribution to the best solutions found and are also relatively scalable.

This finding suggests that if only a single algorithm is to be used then it should be hyper-heuristic NSGA2, 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.

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.

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

Figure 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 NSGA2 (labelled 'GA') tends to degrade with size, the quality of hyper-heuristic NSGA2 (labelled 'HGA') does not.

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 3: 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 all cases. 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 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

Table 4: 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.

References

[1] A. Al-Emran, D. Pfahl, and G. Ruhe. A Hybrid Method for Advanced Decision Support in Strategic Product Release Planning. Technical Report 088/2010, University of Calgary, March 2010.

[2] T. AlBourae, G. Ruhe, and M. Moussavi. Lightweight Replanning of Software Product Releases. In *Proceedings of the 1st International Workshop on Software Product Management (IWSPM '06)*, pages 27–34, Minneapolis, MN, USA, 12-12 September 2006. IEEE.

[3] A. Amandeep, G. Ruhe, and M. Stanford. Intelligent Support for Software Release Planning. pages 248–262, Kansai Science City, Japan, 5-8 April 2004. Springer.

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

[5] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley. The Next Release Problem. *Information and Software Technology*, 43(14):883–890, December 2001.

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

[7] M. M. A. Brasil, T. G. N. da Silva, F. G. de Freitas, J. T. de Souza, and M. I. Cortes. A Multiobjective Optimization Approach to the Software Release Planning with Undefined Number of Releases and Interdependent Requirements. pages 300–314. Springer, 2012.

[8] E. Burke and G. Kendall. *Search Methodologies. Introductory tutorials in optimization and decision support techniques*. Springer, 2005.

[9] E. Burke, J. Silva, and E. Soubeiga. Multi-Objective Hyper-Heuristic Approaches for Space Allocation and Timetabling. pages 129–158. Springer US, 2005.

[10] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: A Survey of the State of the Art. *Journal of the Operational Research Society*, pages 1695–1724, 2013.

[11] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward. chapter A Classification of Hyper-heuristic Approaches, pages 449–468. Springer, 2010. Chapter 15.

[12] X. Cai and O. Wei. 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)*, 2013.

[13] X. Cai, O. Wei, and Z. Huang. Evolutionary Approaches for Multi-Objective Next Release Problem. *Computing and Informatics*, 31:847–875, 2012.

[14] N. Cliff. *Ordinal Methods for Behavioral Data Analysis*. Lawrence Erlbaum Associates, Inc., New Jersey, USA, 1996.

[15] F. Colares, J. T. de Souza, R. A. F. do Carmo, C. Pádua, and G. R. Mateus. A New Approach to the Software Release Planning. In *Proceedings of the 23rd Brazilian Symposium on Software Engineering (SBES '09)*, pages 207–215, Fortaleza, Ceará, Brazil, 5-9 October 2009. IEEE.

[16] M. Crepinsek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: a survey. *ACM Computing Surveys*, 45(3):35:1–35:33, June 2013.

[17] J. T. de Souza, C. L. B. Maia, T. Ferreira, R. A. F. do Carmo, and M. Brasil. An Ant Colony Optimization Approach to the Software Release Planning with Dependent Requirements. pages 142–157, Szeged, Hungary, 10-12 September 2011. Springer.

[18] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[19] J. Del Sagrado and I. M. Del Águila. Ant Colony Optimization for Requirement Selection in Incremental Software Development. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*, Cumberland Lodge, Windsor, UK, 13-15 May 2009. IEEE.

[20] J. Del Sagrado, I. M. Del Águila, and F. J. Orellana. 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)*, pages 67–76, Benevento, Italy, 7-9 September 2010. IEEE.

[21] T. do Nascimento Ferreira and J. T. de Souza. 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, 23 September 2012.

[22] O. J. Dunn. Multiple Comparisons Among Means. *Journal of the American Statistical Association*, 56(293), 1961.

[23] J. J. Durillo, Y. Zhang, E. Alba, M. Harman, and A. J. Nebro. A Study of the Bi-Objective Next Release Problem. *Empirical Software Engineering*, 16(1):29–60, February 2011.

[24] J. J. Durillo, Y. Zhang, E. Alba, and A. J. Nebro. A Study of the Multi-Objective Next Release Problem. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*, pages 49–58, Cumberland Lodge, Windsor, UK, 13-15 May 2009. IEEE.

[25] M. S. Feather, S. L. Cornford, J. D. Kiper, and T. Menzies. 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)*, pages 10–10, Minnesota, USA, 11 September 2006. IEEE.

[26] M. S. Feather, J. D. Kiper, and S. Kalafat. 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, 20-24 June 2004.

[27] M. S. Feather and T. Menzies. Converging on the Optimal Attainment of Requirements. In *Proceedings of the 10th IEEE International Conference on Requirements Engineering (RE '02)*, pages 263–270, Essen, Germany, 9-13 September 2002. IEEE.

[28] G. A. Ferguson. *Nonparametric Trend Analysis: A practical guide for research workers*. McGill Uniervsoty Press, Montréal, Canada, 1965.

[29] A. Fialho, L. D. Costa, M. Schoenauer, and M. Sebag. Extreme Value Based Adaptive Operator Selection. pages 175–184. Springer, 2008.

[30] A. Fialho, L. D. Costa, M. Schoenauer, and M. Sebag. Analyzing bandit-based adaptive operator selection mechanisms. *Ann. Math. Artif. Intell.*, 60(1-2):25–64, 2010.

[31] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang. "Fairness Analysis" in Requirements Assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE '08)*, pages 115–124, Barcelona, Catalunya, Spain, 8-12 September 2008. IEEE.

[32] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang. 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):231–245, December 2009.

[33] F. Galton. *Natural Inheritance*. Macmillan and Co., London, UK, 1889.

[34] M. Harman, E. Burke, J. A. Clark, and X. Yao. Dynamic adaptive search based software engineering. In 6^{th} *IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2012)*, pages 1–8, Lund, Sweden, September 2012.

[35] M. Harman and B. F. Jones. Search-based Software Engineering. *Information and Software Technology*, 43(14):833–839, December 2001.

[36] M. Harman, S. A. Mansouri, and Y. Zhang. Search-based Software Engineering: Trends, Techniques and Applications. *ACM Computing Surveys (CSUR)*, 45(1), November 2012.

[37] M. Harman, P. McMinn, J. Souza, and S. Yoo. Search based software engineering: Techniques, taxonomy, tutorial. In

B. Meyer and M. Nordio, editors, *Empirical software engineering and verification: LASER 2009-2010*, pages 1–59. Springer, 2012. LNCS 7007.

[38] Y. Hochberg. A sharper bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802, 1988.

[39] H. Jiang, J. Xuan, and Z. Ren. Approximate Backbone based Multilevel Algorithm for Next Release Problem. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 1333–1340, Portland, Oregon, USA, 7-11 July 2010. ACM.

[40] H. Jiang, J. Zhang, J. Xuan, Z. Re, and Y. Hu. A Hybrid ACO Algorithm for the Next Release Problem. In *Proceedings of the 2nd International Conference on Software Engineering and Data Mining (SEDM '10)*, pages 166–171, Chengdu, China, 23-25 June 2010. IEEE.

[41] M. Kendall. *Rank correlation methods*. Charles Griffin and Company Limited, London, UK, 1948.

[42] W. H. Kruskal and W. A. Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[43] A. Kumari, K. Srinivas, and M. Gupta. 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)*, pages 782–787, Nagapattinam, Tamil Nadu, India, 30-31 March 2012 2012. IEEE.

[44] C. Li, M. Van den Akker, S. Brinkkemper, and G. Diepen. An Integrated Approach for Requirement Selection and Scheduling in Software Release Planning. *Requirements Engineering*, 15(4):375–396, November 2010.

[45] H. B. Mann and D. R. Whitney. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[46] K. McClymont and E. C. Keedwell. 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*, pages 2003–2010, New York, NY, USA, 2011. ACM.

[47] K. Pearson. Notes on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, June 1895.

[48] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers and Operations Research*, 34:2403–2435, 2007.

[49] O. Räihä. A survey on search–based software design. *Computer Science Review*, 4(4):203–249, 2010.

[50] G. Ruhe. *Product Release Planning: Methods, Tools and Applications*. CRC Press, June 2010.

[51] G. Ruhe and D. Greer. Quantitative Studies in Software Release Planning under Risk and Resource Constraints. In *Proceedings of the International Symposium on Empirical Software Engineering (ISESE '03)*, pages 262–270, Rome, Italy, 29 September-4 October 2003. IEEE.

[52] G. Ruhe and A. Ngo-The. Hybrid Intelligence in Software Release Planning. *International Journal of Hybrid Intelligent Systems*, 1(1-2):99–110, April 2004.

[53] G. Ruhe and M. O. Saliu. The Art and Science of Software Release Planning. *IEEE Software*, 22(6):47–53, November 2005.

[54] M. O. Saliu and G. Ruhe. 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*, pages 105–114, Dubrovnik, Croatia, 3-7 September 2007. ACM.

[55] O. Saliu and G. Ruhe. Supporting Software Release Planning Decisions for Evolving Systems. In *Proceedings of the 29th Annual IEEE/NASA on Software Engineering Workshop (SEW '05)*, pages 14–26, Greenbelt, Maryland, USA, 6-7 April 2005. IEEE.

[56] N. J. Salkind. *Encyclopaedia of measurement and statistics*. Sage publications, Inc., California, USA, 2007.

[57] M. J. Shepperd. *Foundations of software measurement*. Prentice Hall, 1995.

[58] C. E. Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, January 1904.

[59] D. Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1539–1546, New York, NY, USA, 2005. ACM.

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

[61] M. Van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Flexible Release Planning using Integer Linear Programming. In *Proceeding of the 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '05)*, pages 247–262, Porto, Portugal, 13-14 June 2005. Essener Informatik Beitrage.

[62] M. Van den Akker, S. Brinkkemper, G. Diepen, and J. Versendaal. Software Product Release Planning Through Optimization and What-if Analysis. *Information and Software Technology*, 50(1-2):101–111, January 2008.

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

[64] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–Ú83, 1945.

[65] J. Xuan, H. Jiang, Z. Ren, and Z. Luo. Solving the Large Scale Next Release Problem with a Backbone Based Multilevel Algorithm. *IEEE Transactions on Software Engineering*, 38(5):1195–1212, Sept.-Oct. 2012.

[66] Y. Zhang, E. Alba, J. J. Durillo, S. Eldh, and M. Harman. Today/Future Importance Analysis. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation (GECCO '10)*, pages 1357–1364, Portland, USA, 7-11 July 2010. ACM.

[67] Y. Zhang and M. Harman. Search Based Optimization of Requirements Interaction Management. In *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*, pages 47–56, Benevento, Italy, 7-9 September 2010. IEEE.

[68] Y. Zhang, M. Harman, A. Finkelstein, and S. A. Mansouri. Comparing the Performance of Metaheuristics for the Analysis of Multi-stakeholder Tradeoffs in Requirements Optimisation. *Information and Software Technology*, 53(7):761–773, July 2011.

[69] Y. Zhang, M. Harman, and S. L. Lim. Empirical Evaluation of Search Based Requirements Interaction Management. *Information and Software Technology*, 55(1):126–152, January 2013.

[70] Y. Zhang, M. Harman, and S. A. Mansouri. The Multi-Objective Next Release Problem. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1129–1137 (Best Paper Award), London, UK, 7-11 July 2007. ACM.

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

[72] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, Nov. 1999.

Note to referee: We were grateful that the call allows '*a reasonable number of additional pages for references*'. We realise that 2 pages of reference is likely to be above average, but we think it reasonable since it allows our survey of previous work to include all previous studies. It also facilitates proper citation to the statistical techniques we used.