Pareto Efficient Multi-Objective Test Case Selection

Shin Yoo and Mark Harman
King's College London, CREST, Department of Computer Science
Strand, London
WC2R 2LS, UK
{Shin.Yoo,Mark.Harman}@kcl.ac.uk

ABSTRACT

Previous work has treated test case selection as a single objective optimisation problem. This paper introduces the concept of Pareto efficiency to test case selection. The Pareto efficient approach takes multiple objectives such as code coverage, past fault-detection history and execution cost, and constructs a group of non-dominating, equivalently optimal test case subsets. The paper describes the potential benefits of Pareto efficient multi-objective test case selection, illustrating with empirical studies of two and three objective formulations.

Categories and Subject Descriptors: D.2.5 [Software Engineering]: Testing and Debugging

General Terms: Algorithms

Keywords: Test case selection, Multi-objective evolutionary algorithm

1. INTRODUCTION

Regression testing is the testing performed in order to guarantee that newly introduced changes in a software do not affect the unchanged parts of the software. One possible approach to regression testing is the *retest-all* method, in which the tester simply executes all of the existing test cases to ensure that the new changes are harmless. Unfortunately, this is a very expensive process; time limitations force a consideration of test case selection and prioritisation techniques[1, 2, 8, 13, 17, 19, 20, 22].

Test case selection techniques try to reduce the number of test cases to be executed, while satisfying the testing requirements denoted by a test criterion. Test case prioritisation techniques try to order the test cases in such a way that increases the rate of early fault-detection.

In real world testing, there are often multiple test criteria. For example, different types of testing, such as functional testing and structural testing, require different testing criteria [9]. There also can be cases where it is beneficial for the tester to consider multiple test criteria because the single most ideal test criterion is simply unobtainable. For example, testers face the problem that real fault detection information cannot be known until regression testing is actually finished. Code coverage is one possible surrogate test adequacy criterion that is used in place of fault detection, but it is not the only one. Because one cannot be certain of a link between code coverage and fault detection it would be natural to supplement coverage with other test criteria, for example, past fault detection history.

Of course, the quality of the test data is not the only concern. Cost is also one of the essential criteria, because the whole purpose of test case selection and prioritisation is to achieve more efficient testing in terms of the cost. One important cost driver, considered by other researchers [13, 20] is the execution time of the test suite.

In order to provide automated support for the selection of regression test data it therefore seems inevitable that a multi-objective approach is required that is capable of taking into account the subtleties inherent in balancing many, possibly competing and conflicting objectives. Existing approaches to regression test case selection (and prioritisation) have been single objective approaches that have sought to optimise a single objective function.

For the prioritisation problem, there has been recent work on a two objective formulation [13], that takes account of coverage and cost, using a single objective of coverage per unit cost. However, this approach conflates the two objectives into a single objective. Where there are multiple competing and conflicting objectives the optimisation literature recommends the consideration of a Pareto optimal optimisation approach [4, 18]. Such a Pareto optimal approach is able to take account of the need to balance the conflicting objectives, all of which the software engineer seeks to optimise.

This paper presents the first multi-objective formulation of the test case selection problem, showing how multiple objectives can be optimised using a Pareto efficient approach. We believe that such an approach is well suited to the regression test case selection problem, because it is likely that a tester will want to optimise several possible conflicting constraints.

The primary contributions of this paper are as follows:

1. The paper introduces a multi-objective formulation of the regression test case selection problem and instantiates this with two versions: A two objective formulation that combines coverage and cost and a three objective formulation that combines coverage, cost and fault history. The formulation facilitates a theoretical

treatment of the optimality of the greedy algorithm and allows us to establish a relationship between the multi-objective problems of test case prioritisation and test case selection.

2. The paper presents three algorithms for solving the two and three objective instances of the test case selection problem: a re-formulation of the single–objective greedy algorithm, the Non Dominating Sorting Genetic Algorithm (NSGA-II) of Deb et al. [6] and an island genetic algorithm variant of NSGA-II, which we call vNSGA-II.

3. The paper presents the results for these algorithms, when applied to the two objective version of the problem using, as subjects, four programs from the Siemens suite [11], together with `space`. The results confirm the theoretical analysis, revealing cases where the search based algorithms out–perform the greedy approach. However, they also show that the greedy approach is capable of producing good approximations to the Pareto front, indicating that the results from the greedy approach can be used to augment those from the search–based algorithms in a hybrid approach.

4. The paper also presents results from an empirical study of the three algorithms applied to the three objective formulation of the problem. These results also show that the search–based approaches can out–perform the greedy approach (it is statistically significantly out–performed for all programs in the Siemens suite by NSGA-II). However, the results also show that the greedy approach is capable of producing strong results, indicating that a hybrid approach is also required for the three objective formulation.

The rest of this paper is organised as follows. Section 2 presents the background of the existing single objective formulation of the test case selection and prioritisation problems, while Section 3 introduces the multi-objective formulation, giving theoretical results and connections between the selection and prioritisation problems. Section 4 presents two empirical studies of multi-objective test case selection for two and three objective version of the multi-objective formulation. The results of the empirical studies are analysed in Section 5. Section 6 describes related work, while Section 7 concludes with directions for future work.

2. SINGLE OBJECTIVE PARADIGM

Selection and prioritisation of test cases are proposed as two important solutions to the problem of test case management. Test case selection techniques try to improve the *retest-all* approach by selecting a subset of the entire test suite based on some test criteria. Test case prioritisation techniques try to find an ordering of test cases so that some test adequacy can be maximised as early as possible.

One of the criteria for selecting test cases is to have a *safe* selection of test cases. Let us assume that we have a program P, its new version, P', and a test suite, T. A test case in T is said to be *modification-traversing* if it executes code that was changed or inserted into P (to create a new version P'), or deleted from P. A subset T' of T is said to be *safe* if it includes all the modification-traversing test cases in T:

Safe Test Case Selection *Given:* a program P, its new version P', and a test suite, T *Problem:* to find T' such that $T' \subset T$, $(\forall t \in T)$ [t is modification-traversing $\Rightarrow t \in T'$].

Leung and White showed that the selective method is more economical provided that the cost of the selection process does not exceed the gain in cost that is achieved by omitting the non-selected test cases [10].

The high cost of regression testing, especially in regards to time, also means that it is ideal to have a scheduling of test cases such that any fault that can be revealed by the given test suite is revealed as early as possible. Test case prioritisation is a problem of choosing an ordering of the test suite that maximises some test adequacy. It is defined by Rothermel et al. as follows [17]:

Test Case Prioritisation *Given:* a test suite, T, the set of permutations of T, PT; a function from PT to real numbers, f.
Problem: to find $T' \in PT$ such that $(\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')]$.

The function f assigns a real value to a permutation of T according to the test adequacy of the particular permutation. The ideal order would be the one that reveals faults soonest, the rate of which can be expressed in Average Percentage of Fault Detection. However, since fault detection cannot be known in advance, most of the currently proposed techniques have been code-based, replacing the fault detection with code coverage. Rothermel et al. investigated several prioritisation techniques with different coverage measurements, and showed that coverage prioritisation can improve the rate of fault detection [17].

3. MULTI OBJECTIVE PARADIGM

This section introduces the multi-objective formulation of test case selection. Section 3.1 introduces the Pareto optimal formulation of the test case selection problem. Section 3.2 explores the theoretical properties of the two objective greedy algorithm, while section 3.3 shows the relationship between multi-objective selection and prioritisation.

3.1 Pareto Optimality

Pareto optimality is a notion from economics with broad range of applications in game theory and engineering. The original presentation of the Pareto optimality is that, given a set of alternative allocations and a set of individuals, allocation A is an improvement over allocation B only if A can make at least one person better off than B, without making any other worse off.

Based on this, the multi-objective optimisation problem can be defined as to find a vector of decision variables x, which optimises a vector of M objective functions $f_i(x)$ where $i = 1, 2, \ldots, M$. The objective functions are the mathematical description of the optimisation criteria, which are often in conflict with each other.

Without the loss of generality, let us assume that we want to maximise f_i where $i = 1, 2, \ldots, M$. A decision vector x is said to dominate a decision vector y (also written $x \succ y$) if and only if their objective vectors $f_i(x)$ and $f_i(y)$satisfies:

$$f_i(x) \geq f_i(y) \forall i \in \{1, 2, \ldots, M\}; and$$

$$\exists i \in \{1, 2, \ldots, M\} | f_i(x) > f_i(y)$$

All decision vectors that are not dominated by any other decision vectors are said to form the *Pareto optimal set*, while the corresponding objective vectors are said to form the *Pareto frontier*. Now the multi-objective optimisation problem can be defined as follows:

Given: a vector of decision variables, x, and a set of objective functions, $f_i(x)$ where $i = 1, 2, \ldots, M$
Definition: maximise $\{f_1(x), f_2(x), \ldots, f_M(x)\}$ by finding the Pareto optimal set over the feasible set of solutions.

Identifying the Pareto frontier is particularly useful in engineering because the decision maker can use the frontier to make a well-informed decision that balances the trade-offs between the objectives.

The multi-objective test case selection problem is to select a Pareto efficient subset of the test suite, based on multiple test criteria. It can be defined as follows:

Multi Objective Test Case Selection *Given:* a test suite, T, a vector of M objective functions, f_i, $i = 1, 2, \ldots, M$.
Problem: to find a subset of T, T', such that T' is a Pareto optimal set with respect to the objective functions, f_i, $i = 1, 2, \ldots, M$.

The objective functions are the mathematical descriptions of test criteria concerned. A subset t_1 is said to dominate t_2 when the decision vector for t_1 ($\{f_1(t_1), \ldots, f_M(t_1)\}$) dominates that of t_2. The resulting subset of the test suite, T', has several benefits in regards to the regression testing, as shown in Section 3.2.

3.2 Properties of 2 Objective Coverage Based Selection

In this paper, we will instantiate the two objective formulation with code coverage as a measure of test adequacy and execution time as a measure of cost. Thus, code coverage becomes one of the two objectives, and it should be maximised for a given cost. Time is the other objective, which should be minimised for a given code coverage.

In this instantiation of the problem, should there exist a subset of test suite s with coverage c_1 and execution time t_1 on the Pareto frontier, it means that:

- **T1.** No other subset of s can achieve more coverage than c_1 without spending more time than t_1.

- **T2.** No other subset of s can finish in less time than t_1 while achieving a coverage that is equal to or greater than c_1.

This is the implication of Pareto optimality. Rather than obtaining a single answer that approximates the global optimum in the search space for a single objective, we obtain a set of points, each of which denotes one possible way of balancing the two objectives in a globally optimal way. Each member of the Pareto frontier is therefore a candidate solution to the problem, upon which it is not possible to improve.

In the single objective formulation of test case selection, greedy algorithms have been used to maximise coverage. The greedy approach starts with an empty test set as the 'current solution' and iteratively adds a test case which gives the most coverage of those that remain. A variant, additional greedy, improves on this by adding to the current solution the test case that gives the best *additional* coverage to the current solution. Each addition by the greedy algorithm of a new test case to the 'current solution' denotes a candidate element of the Pareto frontier.

Greedy algorithms have proved effective for the single objective formulation, so they make a sensible starting point for the consideration of the multi-objective formulation. In order to optimise both coverage and cost, the additional greedy algorithm will need to be formulated to measure not coverage, but coverage per unit time. This produces a single objective cost cognizant variant of the greedy algorithm, similar to that used by Rothermel et al. for the single objective prioritisation problem [13].

Suppose the algorithm has chosen a test case T_1 with coverage c_1 and time t_1. By definition, there is no single test case T' such that $c' > c$ and $t' = t$ (otherwise the algorithm would have picked T'). Therefore, the two objective cost cognizant additional greedy algorithm cannot be improved upon by the addition of a single case. However, this leaves open the possibility that there may be a *set* of test cases that, taken together, could have produced a better approximation to the Pareto front.

It turns out that any point on the Pareto front that dominates a point found in the course of the additional greedy algorithm can only do so by improvement with respect to **T2**. It is not possible to improve on the additional greedy algorithm with respect to **T1**. This observation is stated and proved more formally below.

Proposition 1 (Partial Optimality). *The set of points produced by the additional greedy algorithm cannot be improved upon with respect to* **T1**.

Proof. Suppose the contrary. That is, let $T_1 = c_1/t_1$ be the solution found by the additional greedy algorithm. Suppose there exists a pair of test cases, T_2 and T_3, each with coverage of c_2/c_3 and time of t_2/t_3 that, together, improve upon T_1 by achieving more coverage without spending more time. By definition, we have

$$\frac{c_2}{t_2} < \frac{c_1}{t_1} \wedge \frac{c_3}{t_3} < \frac{c_1}{t_1}$$

because, otherwise, the additional greedy algorithm would not have selected T_1. In order for $T_2 \cup T_3$ to be a better choice than T_1 we require $coverage(T_2 \cup T_3) > c_1, t_2 + t_3 = t_1$. From which it follows that: $t_1(c_2 + c_3) < (t_2 + t_3)c_1$

Replacing t_1 with $t_2 + t_3$ we get: $c_2 + c_3 < c_1$. Now, because code coverage is a set theoretic concept, it is not possible for the coverage of the union to be greater than the sum of the coverage of the parts, so we have: $coverage(s_2 \cup s_3) \le c_2 + c_3$

Therefore $coverage(T_2 \cup T_3) \le c_2 + c_3$ and by transitivity, $coverage(T_2 \cup T_3) < c_1$, which breaks the initial assumption, so we must conclude that it is not possible to dominate the results from the additional greedy algorithm by breaking **T1**. □

However it is possible to construct an example that shows that the additional greedy algorithm does not produce solutions that are Pareto efficient with respect to **T2**. Such an example is shown in Table 1. The first choice of the additional greedy algorithm will be T_1, which has the additional coverage per unit time value of $\frac{0.8}{4} = 0.2$ (T_2, T_3, T_4 each

	Program Points										Exec. Time
T_1	X	X	X	X	X	X	X	X			4
T_2	X	X		X	X	X	X	X	X	X	5
T_3	X	X	X						X		3
T_4	X	X	X	X	X						3

Table 1: An example of a test suite where the additional greedy algorithm produces suboptimal selection of test cases

has 0.18, $0.1\tilde{3}$, and $0.1\tilde{6}$). The second choice will be T_2 with the additional coverage per unit time value of $\frac{0.2}{5} = 0.04$, whereas T_3 and T_4 each has $0.0\tilde{3}$ and 0. At this point, the algorithm achieves 100% coverage in 9 units of time. However, the same amount of coverage is also achievable in 8 units of time by selecting T_2 and T_3, so the subset $\{T_2, T_3\}$ dominates the subset $\{T_1, T_2\}$.

Furthermore, though the additional greedy algorithm may produce points that are Pareto efficient with respect to **T1**, it does not produce a complete Pareto frontier. The existence of T_4 in the above example demonstrates this. According to the additional greedy algorithm, the first decision point chosen for this example would be the subset of $\{T_1\}$, which achieves 80% coverage in 4 units of time. The subset $\{T_1\}$ is on the Pareto frontier because no other test case can achieve 80% coverage in 4 units of time. However, the subset of $\{T_4\}$ is *also* on the Pareto frontier, because no other test case can achieve 50% coverage in 3 units of time. This point $\{T_4\}$ on the Pareto frontier is ignored by the additional greedy algorithm. As we will see in the next subsection, this issue is important, because it is necessary to produce the most complete approximation to the Pareto front possible in order to exploit the relationship between multi-objective selection and prioritisation.

3.3 The Relationship Between Multi Objective Selection and Prioritisation

While they are formally different concepts, test case selection and prioritisation problems are closely related to each other in real world decision making process, where the tester wants more efficient regression testing. Test case prioritisation concerns the most ideal ordering of a given test suite. Since it only changes the order of a given test suite, it is not capable of producing an efficient test case scheduling when the available time is shorter than the total time required by the test suite.

Figure 1 shows the result that the additional greedy algorithm produces with the test data shown in Table 1, along with the real Pareto frontier of the test data. If the budget allows 9 units of time for the testing, the result of the additional greedy algorithm can be applied with a final coverage of 100%. Now suppose that the budget allows only 6 units of time. From the result of the additional greedy algorithm, the next feasible solution is to execute just T_1, achieving 80% coverage. However, the Pareto frontier tells us that the subset of T_2 can be executed in 5 units of time, achieving 90% coverage. It also shows us that a coverage of 100% is achievable in only 8 units of time. It also reveals that, should the budget allow only 3 units of time, it is still possible to achieve 50% coverage by executing T_4.

The benefit of knowing the existence of $\{T_2, T_3\}$ and $\{T_4\}$ as candidate selections of test cases becomes clear under

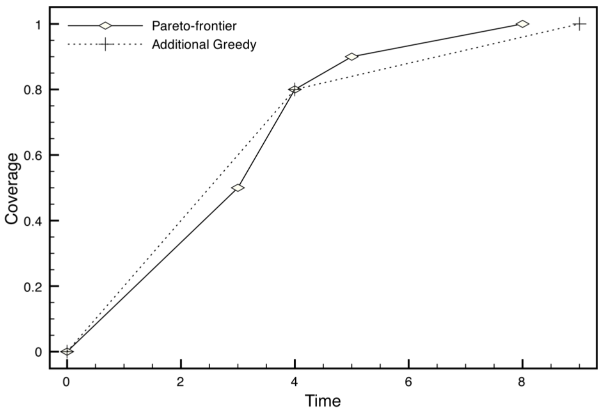

Figure 1: Comparison between the Pareto frontier and the results of the additional greedy algorithm from the test data shown in Table 1

the assumption that there is a cost constraint, i.e., testing budget. Prioritisation techniques make no such assumption; they assume that whatever ordering of test cases they produce can be executed in its entirety. However, there can be situations when the exact amount of the available budget is known before the testing begins. Test case prioritisation techniques cannot optimise the testing process in such a situation, because they are not capable of *selecting* test cases. In order to construct an efficient test sequence under cost constraint, an appropriate subset of test cases should be selected first. This subset can *subsequently* be prioritised in order to achieve the ideal ordering among the selected test cases. This way, test case selection and prioritisation techniques can be used in combination in order to achieve more efficient regression testing.

4. EMPIRICAL STUDIES

This section explains the experiments conducted to explore the two and three objective formulations of the multi-objective selection problem. Sections 4.1 and 4.2 set out the research questions and subjects studied. Section 4.3 describes the objectives to be optimised. Section 4.4 describes the algorithms studied, while Section 4.5 explains the mechanisms by which these algorithms will be evaluated in the two empirical studies.

4.1 Research Questions

The first three research questions can be answered quantitatively using the approaches described in Section 4.5. The last research question is more qualitative in nature.

RQ1: Do the situations theoretically predicted in Section 3.2 arise in practise? That is, does there exist a situation in which the greedy algorithm can be out–performed by solutions that achieve identical coverage in less time? Do there exist situations where the Pareto efficient approaches produce more points on the Pareto front than the greedy algorithm?

RQ2: How well do the greedy and search–based algorithms perform compared to each other and to the global optimum for the 2-objective formulation?

RQ3: How well do the greedy and search–based algorithms perform compared to each other for the 3-objective formulation?

RQ4: What can be said about the shape of the Pareto frontiers, both approximated and optimal? What insights do they reveal concerning the tester's dilemma as to how to balance the trade-offs between objectives?

4.2 Subjects

A total of 5 programs were studied in this paper: a part of the Siemens suite (`printtokens, printtokens2, schedule, schedule2`), and the program `space` from the European Space Agency. These programs range from 374 to 6,199 lines of code, and include a real world application. The software artifacts were available from Software-artifact Infrastructure Repository (SIR) [7].

Each program has a large number of available test suites. Four test suites were randomly selected for each program; therefore a total of 20 test suites were used as input to the multi-objective Pareto optimisation. The size of the programs and their test suites are shown in Table 2.

Program	Lines of Code	Avg. test suite size
printtokens	726	16
printtokens2	570	17
schedule	412	8
schedule2	374	8
space	6199	153

Table 2: Size of test suites of studied programs

4.3 Objectives

It is not the aim of this paper to enter into a discussion concerning which objectives are more important for regression testing. We simply note that, irrespective of arguments about their suitability, coverage and fault histories are likely candidate objectives for assessing test adequacy and that execution time is one realistic measure of effort.

For the two objective formulation, statement coverage and computational cost of test cases will be used as objectives. The additional objective used in the three objective formulation is the past fault detection history. Each software artifact used in this paper has several seeded faults (taken from the data available on the SIR [7]), which are associated with the test cases that reveal them. Using this information, it is possible to assign past fault coverage to each test case subset, which corresponds to how many of the known, past faults in the previous version this subset would have revealed.

Physical execution time of test cases is hard to measure accurately. It involves many external parameters that can affect the physical execution time; different hardware, application software and operating system. In particular, any measurement of execution time is likely to be affected by aspects of the environment unconnected to the the choice of test cases. Such factors include concurrent execution, caching and other low-level processor optimisations.

In this paper we circumvent these issues by using the software profiling tool, Valgrind, which executes the program binary code in an emulated, virtual CPU [14]. The computational cost of each test case was measured by counting the number of virtual instruction codes executed by the emulated environment. Valgrind was created to allow just this sort of precise and unequivocal assessment of computational effort; it allows us to argue that these counts are directly proportional to the cost of the test case execution.

4.4 Algorithms

Two different Pareto efficient genetic algorithms, NSGA-II and its variation were used in this paper. NSGA-II is a multi-objective genetic algorithm developed by Deb et al. [6]. The output of NSGA-II is not a single solution, but the final state of the Pareto frontier that the algorithm has constructed. Pareto optimality is used in the process of selecting individuals. This leads to the problem of selecting one individual out of a non-dominated pair. NSGA-II uses the concept of crowding distance to make this decision; crowding distance measures how far away an individual is from the rest of the population. NSGA-II tries to achieve a wider Pareto frontier by selecting individuals that are far from the others. NSGA-II is based on elitism; it performs the non-dominated sorting in each generation in order to preserve the individuals on the current Pareto frontier into the next generation.

A variation of NSGA-II, which we call vNSGA-II, was also implemented. Two major modifications to NSGA-II were made for vNSGA-II. First, the algorithm uses a group of sub-populations that are separate from each other, in order to achieve wider Pareto frontiers. When performing a pairwise tournament selection on individuals that form a non-dominated pair, each of the sub-populations slightly prefers different objectives so that the Pareto frontier can be advanced in all the directions. vNSGA-II also extends the elitism of NSGA-II by keeping a *best-so-far* record of the Pareto frontier separate from the sub-populations.

Two Greedy Algorithms were also implemented. For the two objective formulation, the cost cognizant version of the additional greedy algorithm was implemented. For the three objective formulation, the three objectives were combined into a single objective according to the classical weighted-sum approach. With M different objectives, f_i with $i = 1, 2, \ldots, M$, the weighted-sum approach calculates the single objective, f', as follows:

$$f' = \sum_{i=1}^{M} (w_i \cdot f_i), \sum_{i=1}^{M} w_i = 1$$

Both the additional code coverage per until time and additional past fault coverage per unit time were combined using coefficients of 0.5 and 0.5, thereby giving equal weighting to each objective.

4.5 Evaluation Mechanisms

The difficulty of evaluating Pareto frontiers lies in the fact that the absolute frame of reference is the *real* Pareto frontier, which by definition, is impossible to know *a priori*. Instead, a reference Pareto frontier can be constructed and used when comparing different algorithms with respect to the Pareto frontiers they produce. The reference frontier represents the hybrid of all approaches, combining the best of each. It is one of the advantages of Pareto optimality that results for various approaches can be combined in this way.

More formally, let us assume that we have N different Pareto frontiers, P_i with $i = 1, 2, \ldots, N$. A reference Pareto frontier, P_{ref}, can be formulated as follows. Let P' be the union of all P_i with $i = 1, 2, \ldots, N$. Then:

$$P_{ref} \subset P', (\forall p \in P_{ref})(\nexists q \in P')(q \succ p)$$

For the programs from the Siemens suite, the search spaces were sufficiently small to allow us to perform an exhaustive search to locate the true Pareto frontier. This allows us to compare the results from the algorithms to the globally optimal solution in these cases. For the program `space` this was not possible, so the reference Pareto frontier was formed as described.

One of the methods to compare Pareto frontiers is to look at the number of solutions that are not dominated by the reference Pareto frontier. By definition, P_{ref} is not dominated by any of the N different Pareto frontiers, because it consists of the best parts of the different Pareto frontiers. However, each of N different Pareto frontiers may be partly dominated by P_{ref}. Therefore, these N different Pareto frontiers can be compared with each other by counting the number of solutions that are not dominated by P_{ref} in each Pareto frontier.

Another meaningful measurement is the size of each Pareto frontier. Achieving wider Pareto frontiers is one of the important goals of Pareto optimisation. This is particularly of concern in engineering application, because a wider Pareto frontier means a larger number of alternatives available to the decision maker.

Both the number of non-dominated solutions and the size of Pareto frontiers were measured and statistically analysed in this paper using Welch's t-test. Welch's t-test is a statistical hypothesis test for two groups with different variance values. It tests the null hypothesis that the means of two normally distributed groups are equal. In the context of this paper, the null hypothesis is that with two different algorithms, the mean values of the number of solutions that are not dominated by the reference Pareto frontier are equivalent. For these tests the α level was set to 0.95. Significant $p - values$ suggest that the null hypothesis should be rejected in favour of the alternative hypothesis, which states that one of the algorithm produces a larger number of non-dominated solutions.

NSGA-II and vNSGA-II algorithms were both executed 20 times for each test suite to account for their inherent randomness. Both algorithms use single-point crossover and bit-flip mutation. NSGA-II is configured with the recommended setting of $\{population = 100$, and $maximum\ fitness\ evaluation = 25,000\}$ for the Siemens suite. vNSGA-II uses three different sub-population groups of $\{population = 300$, and $maximum\ iteration = 250\}$ for the Siemens suite. For `space`, both algorithm use the setting of $\{population = 1,500$ and $\{maximum\ iteration = 180\}$. In the case of vNSGA-II, this means three sub-populations with 500 individuals.

5. RESULTS AND ANALYSIS

The results for the 2-objective formulation for the five different subjects are shown in Figure 2. The figures are provided for illustration and qualitative evaluation only. For complete quantitative data, see Table 3.

In particular, it should be noted that the lines connecting the data points are drawn merely in order to aid the visual comprehension of the plot; no meaning can be ascribed concerning the results that may or may not exist along these lines, apart from the points plotted. In case with vNSGA-II and NSGA-II, a single result was chosen out of the 20 experiments in order to produce readable images. The variance in their complete results over 20 runs can be seen in Table 3.

The results from the programs from the Siemens suite confirm the theoretical argument set out in Section 3.1; there do exist data points that achieve the same amount of coverage as the additional greedy algorithm, but in less time. The size of Pareto frontiers produced by the Pareto efficient genetic algorithms are larger than those produced by the additional greedy algorithm, giving more information to the tester. In all four smaller programs, the Pareto efficient genetic algorithms produce subsets of test cases that can be executed in fewer than 200 units of cost, something for which the additional greedy algorithm is incapable. These results provide a positive answer to **RQ1**.

It can also be observed that NSGA-II is capable of identifying the entire reference frontier, producing the exhaustive result. The results from vNSGA-II are not always exhaustive, but they still outperform the additional greedy algorithm.

However, the result for the (larger) program `space` shows the contrary; the additional greedy algorithm performs very well, dominating the rest of the algorithms. NSGA-II, which is very competitive with the smaller programs, manages to produce results that are close to those produced by the additional greedy algorithm but none of the points on its approximation to the Pareto frontier dominates those found by the additional greedy algorithm. vNSGA-II partly dominates NSGA-II, but its results are still inferior to those of the additional greedy algorithm. The good performance of the additional greedy algorithm suggests that the existing test case prioritisation techniques are capable of producing solutions that are strongly Pareto efficient. This is a very attractive finding, given the computations efficiency of the greedy algorithms, compared to the alternatives.

These findings provide a mixed message for the answer to **RQ2**. The data show that the additional greedy algorithm may be dominated by the Pareto efficient genetic algorithms, but also that, for some programs the additional greedy algorithm produces the best results. This suggests that for optimal quality test data selection it may be advisable to combine the results from greedy and evolutionary algorithms. This is one of the attractive aspects of the Pareto efficient approach; results from several algorithms can be merged to form a single Pareto front that combines the best of all approaches.

Figure 3 shows the results for the three objective formulation. The 3D plots display the solutions produced by the weighted-sum additional greedy algorithm (depicted by a line in the figures), and the reference Pareto frontier (depicted by square–shaped points). The weighted-sum additional greedy algorithm produces very strong results because the line can be seen to connect the data points forming the reference Pareto frontier, meaning that the solutions from the weighted-sum and additional greedy algorithm form a part of the reference Pareto frontier (which is later confirmed by a statistical analysis).

These results suggest that the answer to **RQ3** is also mixed. Even where there are more than 2 objectives, the greedy approach is capable of reasonable performance. Therefore, a combination of results may be appropriate. Of course, these findings will depend upon the three objectives chosen. More work is required to experiment with other objectives. It is not possible to extrapolate from these results to con-

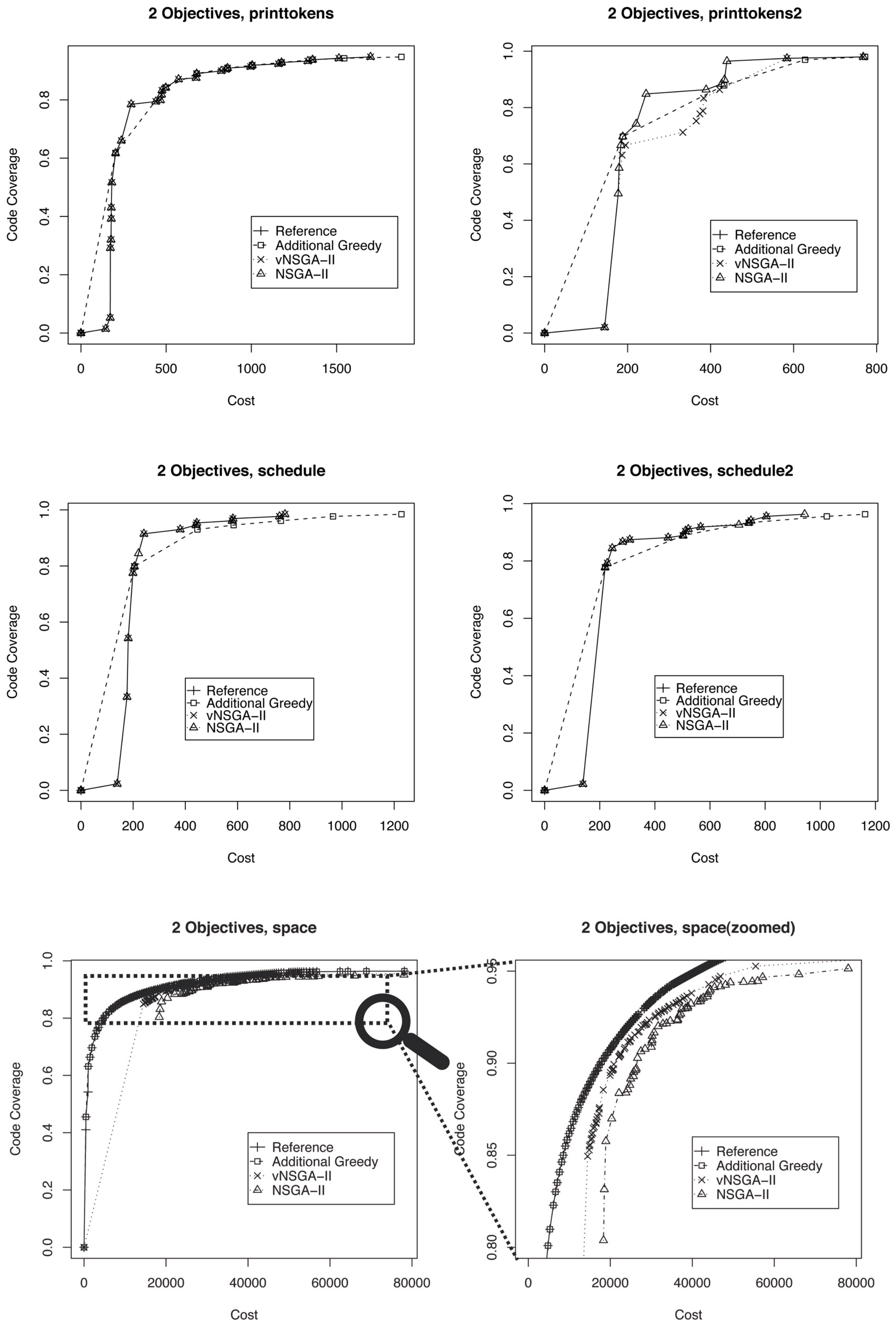

Figure 2: Plot of Pareto frontier for two objective formulation. With the Siemens suite, the results from the additional greedy algorithm are dominated by the reference Pareto frontier obtained by an exhaustive search, which NSGA-II is also capable of finding. However, in the zoomed plot of space, it can be observed that the additional greedy algorithm dominates the rest of the algorithms.

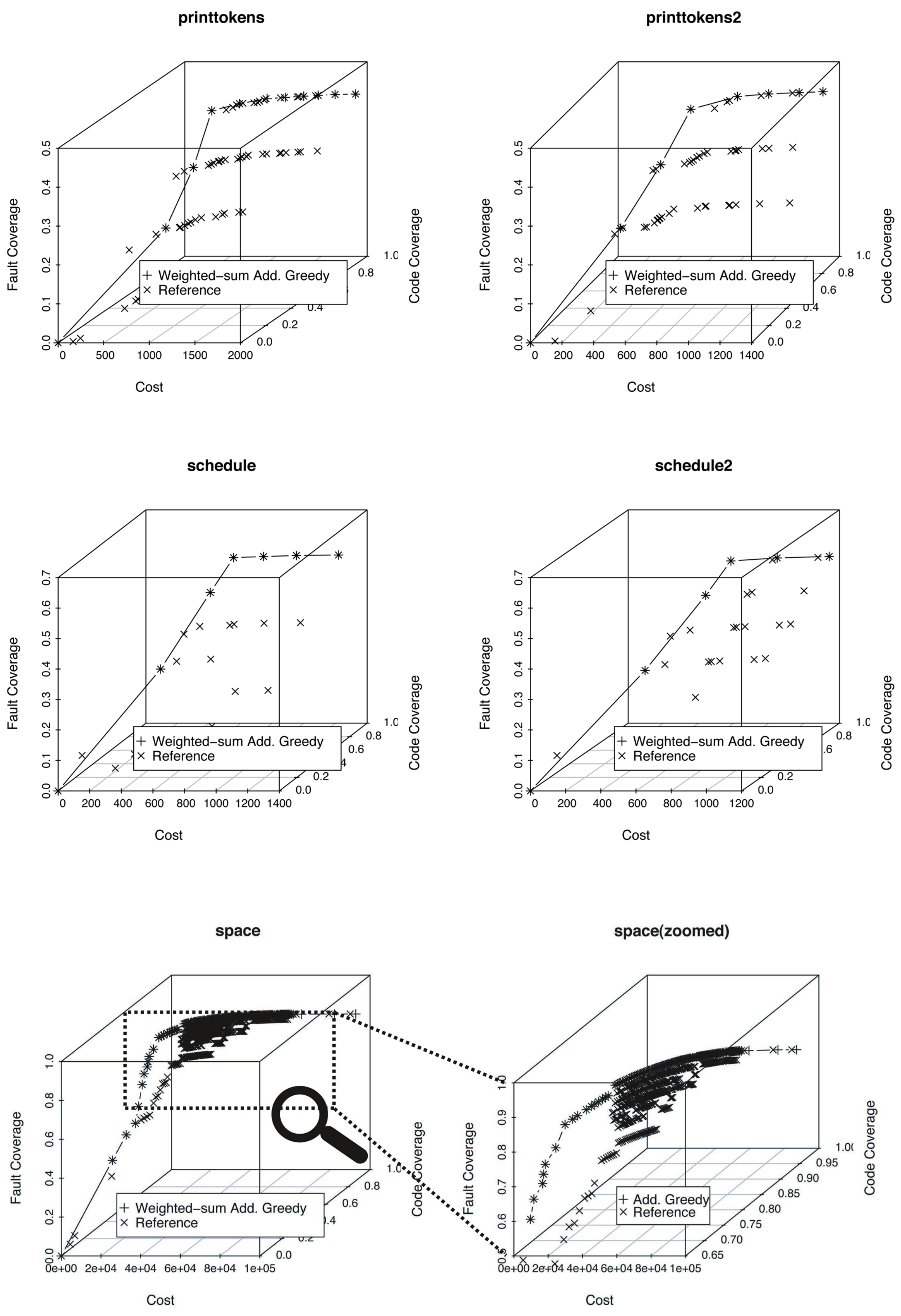

Figure 3: Plot of Pareto frontier for three objective formulation. The line shows the result from weighted-sum additional greedy, while the points correspond to the reference Pareto frontier.

clude than the additional greedy algorithm will perform well for any 3 objective instantiation of the multi-objective test case selection problem. For example, the strong results that we have been able to obtain may be a result of the relative sparseness of the fault history data, which may favour the additional greedy approach.

In order to provide a more concrete quantitative analysis of the answers to **RQ2** and **RQ3**, we compare the results obtained using tests for statistical significance. For the Siemens suite programs, NSGA-II shows the best performance by producing the entire reference Pareto frontier. The results from NSGA-II for the Siemens suite are also very stable; the variance in the size of the Pareto frontier produced by NSGA-II is 0. On the other hand, the results from vNSGA-II show some variance, and therefore are compared to the additional greedy algorithm using the t-test analysis with the confidence level of 95% and using the null hypothesis that there is no difference in the results for $\bar{n}_{vNSGA-II}$ and $\bar{n}_{AdditionalGreedy}$ and the alternative hypothesis that $\bar{n}_{vNSGA-II}$ is greater than $\bar{n}_{AdditionalGreedy}$. The observed $p-values$ for these t–tests are significant at the 95% level, confirming the alternative hypothesis.

However, the weighted-sum additional greedy algorithm produces the best result with `space`. For the analysis of vNSGA-II results from `space`, the alternative hypothesis is that $\bar{n}_{vNSGA-II}$ is smaller than $\bar{n}_{AdditionalGreedy}$. The observed $p-values$ are significant at the 95% level, confirming the alternative hypothesis. For some test suites of `space`, the results were constant, making the t-test inapplicable. For the program `space`, the Pareto frontiers produced by NSGA-II are completely dominated by the results from the additional greedy algorithm.

Both of the genetic algorithms produce much wider Pareto frontiers than the additional greedy algorithm, which is expected because they are designed to produce Pareto frontiers. However, in terms of the number of solutions that are not dominated by the reference frontier, statistical analysis confirms the results shown in Figure 2 and 3. With smaller programs, NSGA-II performs significantly better than the others in both two and three objective formulations, while `space` shows the contrary. However, the higher deviation observed in the results of vNSGA-II with test suite T_2 of `space` suggest that both the random nature of the genetic algorithm and the composition of a particular test suite may affect the result; further research on wider range of subjects will confirm or refute this.

Turning to the last research question, **RQ4**, a more qualitative analysis is required. This is made possible by the visualisations of the solutions plotted in Figures 2 and 3. It can be seen that the shapes of the lines and the reference Pareto frontiers are relatively similar to each other across all programs, suggesting a similar relationship between coverage and fault detection for these programs. The shape is an interesting observation on the relation between the code coverage and past fault coverage, because it seems to illustrate a relatively strong correlation between the two objectives. Such a correlation may suggest that the concerned faults are not concentrated in a limited part of the code. There also appear to exist a point on the line in every program, where the rate of increase in the fault coverage changes. Such *elbow points* are considered important in the study of Pareto optimality. They indicate points of particular interest where the balance of trade offs inherent in the objectives changes.

In the case of test case selection, the location of this elbow point may tell us the percentage of faults that require certain amount of code coverage to be detected. The selections *below* the elbow point generally contain smaller numbers of test cases, which results in the limited fault detection capability; once a sufficient amount of cost is available, combinations of test cases can be picked up, which improves the rate of the past fault coverage. These results provide evidence to suggest a 'critical mass' phenomenon in test case selection. These observations form a partial answer to **RQ4**, but more data is required to see whether this critical mass phenomenon is generic to test case selection, or whether it is merely an artefact of the set of programs we have chosen to study in the present paper.

5.1 Threats to Validity

Threats to internal validity concern the factors that might have affected the multi-objective optimisation techniques used in the paper. One potential concern involves the accuracy of the instrumentation of the subject software, e.g. the correctness of the coverage information. To address this, a professional and commercial software tool (Cantata++ from IPL ltd.) was used to collect code coverage information. The fault coverage information was extracted from SIR - a well-managed software archive [7]. Precisely determined computational cost was used in place of the physical execution time in order to raise the precision of the cost information using the Valgrind profiling tool [14].

Another potential internal threat comes from the selection and optimisation of the meta-heuristic techniques themselves. No particular algorithm is known to be effective for the multi-objective test case selection problem. However the genetic algorithm used in this paper is known to be effective for a wide range of multi-objective problems [3, 5], and can serve as a basis for the future research.

Threats to external validity concern the conditions that limit generalisation from the result. The primary concern for this paper is the representativeness of the subjects that were studied. This threat can be addressed only by additional research using a wider range of software artifacts and optimisation techniques.

6. RELATED WORK

The existing literature on test case management can be categorised in to three different areas of investigation; *test suite reduction* (or *minimisation*), *test case selection*, and *test case prioritisation*.

Test suite reduction shares many similarities with test case selection, except the fact that the reduction of the test cases is permanent compared to the temporary selection of test cases for a specific testing run. It is known that test suite reduction can be efficient provided that the cost of the reduction is smaller than the gain in the cost of the reduced test suite [10]. However, a weakness of test suite reduction is that the removal of some test cases from the test suite may potentially reduce the fault detecting capability of the test suite too. Some studies have shown that the fault-detection capability of the test suite was indeed damaged [16], while others have shown that the reduced test suite still preserved its fault-detection capability [21]. The reduction technique studied by Harrold et al. is of particular interest in the context of this paper because the technique considers multiple criteria when deciding whether to preserve a test case or not.

2 Objectives											
Program	Suite	vNSGA-II				NSGA-II			Additional Greedy		
		$\bar{n}$	σ	p	Avg. Size	$\bar{n}$	σ	Avg. Size	$\bar{n}$	σ	Size
printtokens	T_1	13.00	2.51	3.5e-16	16.75	23.00	0.00	23.00	2.00	0.00	10.00
	T_2	14.15	2.08	6.0e-13	21.30	33.00	0.00	33.00	8.00	0.00	11.00
	T_3	13.70	3.61	4.3e-08	18.20	30.00	0.00	30.00	8.00	0.00	10.00
	T_4	10.15	2.16	2.4e-14	13.40	19.00	0.00	19.00	2.00	0.00	8.00
printtokens2	T_1	9.05	2.85	9.9e-18	12.55	24.00	0.00	24.00	4.00	0.00	6.00
	T_2	14.00	1.97	1.0e-12	16.00	25.00	0.00	25.00	7.00	0.00	7.00
	T_3	7.75	1.52	3.1e-13	9.40	14.00	0.00	14.00	2.00	0.00	5.00
	T_4	8.60	1.96	1.2e-09	11.90	24.00	0.00	24.00	4.00	0.00	7.00
schedule	T_1	8.90	1.59	2.6e-14	9.00	11.00	0.00	11.00	2.00	0.00	6.00
	T_2	11.95	1.47	< 2.2e-16	12.00	15.00	0.00	15.00	2.00	0.00	7.00
	T_3	10.30	1.13	6.5e-15	10.30	12.00	0.00	12.00	5.00	0.00	5.00
	T_4	10.45	1.47	< 2.2e-16	10.70	13.00	0.00	13.00	2.00	0.00	6.00
schedule2	T_1	7.75	0.91	< 2.2e-16	7.75	9.00	0.00	9.00	2.00	0.00	5.00
	T_2	12.70	1.49	< 2.2e-16	13.15	17.00	0.00	17.00	4.00	0.00	6.00
	T_3	8.70	1.38	3.6e-15	8.70	11.00	0.00	11.00	2.00	0.00	6.00
	T_4	8.40	1.19	5.2e-16	8.45	10.00	0.00	10.00	2.00	0.00	6.00
space	T_1	1.40	1.56	< 2.2e-16	99.50	0.00	0.00	55.55	117.00	0.00	117.00
	T_2	1.05	0.22	< 2.2e-16	107.55	0.00	0.00	54.15	118.00	0.00	118.00
	T_3	1.00	0.00	N/A	94.90	0.00	0.00	55.55	92.00	0.00	119.00
	T_4	1.00	0.00	N/A	104.45	0.00	0.00	54.25	120.00	0.00	121.00

Table 3: Average number of solutions that are not dominated by the reference Pareto frontier($\bar{n}$), standard deviation(σ), and the size of Pareto frontier in two objective formulation

3 Objectives											
Program	Suite	vNSGA-II				NSGA-II			Weighted-sum Greedy		
		$\bar{n}$	σ	p	Avg. Size	$\bar{n}$	σ	Avg. Size	$\bar{n}$	σ	Size
printtokens	T_1	11.95	3.02	1.6e-04	15.20	23.00	0.00	23.00	9.00	0.00	9.00
	T_2	25.25	4.78	2.5e-12	39.60	65.65	1.42	66.70	9.00	0.00	11.00
	T_3	20.80	3.30	9.0e-13	26.70	51.00	0.00	51.00	9.00	0.00	9.00
	T_4	13.55	2.78	6.0e-14	21.25	33.00	0.00	33.00	2.00	0.00	8.00
printtokens2	T_1	21.75	4.84	1.6e-12	27.70	55.00	0.00	55.00	5.00	0.00	7.00
	T_2	28.40	5.37	3.1e-13	35.20	59.75	0.64	60.60	8.00	0.00	8.00
	T_3	14.40	2.54	3.6e-12	16.15	24.00	0.00	24.00	6.00	0.00	6.00
	T_4	17.45	3.50	1.2e-10	25.40	48.00	0.00	48.00	8.00	0.00	8.00
schedule	T_1	12.05	1.82	3.2e-12	12.50	16.00	0.00	16.00	6.00	0.00	6.00
	T_2	16.90	2.25	2.1e-14	18.20	23.00	0.00	23.00	7.00	0.00	7.00
	T_3	12.80	1.32	1.2e-15	13.20	16.00	0.00	16.00	6.00	0.00	6.00
	T_4	14.20	1.32	< 2.2e-16	14.45	18.00	0.00	18.00	6.00	0.00	6.00
schedule2	T_1	11.20	0.95	< 2.2e-16	11.40	14.00	0.00	14.00	5.00	0.00	5.00
	T_2	21.30	2.18	< 2.2e-16	22.30	29.00	0.00	29.00	6.00	0.00	6.00
	T_3	12.00	1.62	3.0e-14	12.45	16.00	0.00	16.00	5.00	0.00	5.00
	T_4	11.50	1.19	4.2e-16	11.60	14.00	0.00	14.00	5.00	0.00	6.00
space	T_1	1.50	1.82	< 2.2e-16	268.65	0.00	0.00	119.20	118.00	0.00	118.00
	T_2	19.05	23.50	2.3e-14	263.30	0.00	0.00	119.95	119.00	0.00	122.00
	T_3	1.00	0.00	N/A	208.55	0.00	0.00	96.10	114.00	0.00	119.00
	T_4	2.80	3.65	< 2.2e-16	183.15	0.00	0.00	88.80	67.00	0.00	122.00

Table 4: Average number of solutions that are not dominated by the reference Pareto frontier($\bar{n}$), standard deviation(σ), and the size of Pareto frontier in three objective formulation

Their technique converts this multi-objective problem into a series of single objective problems, by solving the problem for the first objective then uses this intermediate solution as the starting point of the solution for the next objective.

Test case selection focuses on selecting a subset of the test suite in order to test software modifications. The selection is typically made in terms of the structure of the program P and the test suite T. Several techniques have been considered, including symbolic execution [22], flow graph based [15] and dependence graph based approaches [1, 2].

Test case prioritisation is a closely related topic, in which the goal is to find an optimal order in which to execute test cases. One of the most widely used metrics is APFD (Average Percentage of Fault Detected), which rewards orderings with earlier fault detection abilities [17]. The additional greedy algorithm is known to produce good results for the test case prioritisation problem [8, 12]. Rothermel et al. introduced APFDc, the cost cognizant version of APFD [13], which inspired our formulation of the weighted objective greedy algorithm.

Walcott et al. also take time into account in their work on the test case prioritisation problem [20]. Their approach to prioritisation combines both selection and prioritisation problems into a single objective, which is the weighted sum of the selection fitness and prioritisation fitness. The coefficients used for weights are defined to ensure that selection fitness is the primary objective, while ordering is secondary.

7. CONCLUSION AND FUTURE WORK

This paper introduced the concept of Pareto efficient multi-objective optimisation to the problem of test case selection. It described the benefits of Pareto efficient multi-objective optimisation, and presented an empirical study that investigated the relative effectiveness of three algorithms for Pareto efficient multi-objective test case selection. The empirical results obtained reveal that greedy algorithms (which perform well for single objective formulations) are not always Pareto efficient in the multi-objective paradigm, motivating the study of metaheuristic search techniques. Future work will consider a wider range of software artifacts with different meta-heuristic multi-objective optimisation techniques.

Acknowledgements: The `space` examples used in this paper are available from the Software-artifact Infrastructure Repository [7]. We are grateful to Zheng Li for the providing the coverage analysis data, which was obtained using the IPL tool Cantata++. We are also grateful to Gregg Rothermel and Sebastian Elbaum for several discussions on the relationship between multi-objective selection and prioritisation that were helpful in formulating some of the work presented here.

Shin Yoo is supported by the EPSRC SEBASE project (EP/D050863). Mark Harman is supported by EPSRC Grants EP/D050863, GR/S93684 & GR/T22872, by EU grant IST-33472 (EvoTest) and also by the kind support of Daimler-Chrysler Berlin and Vizuri Ltd., London.

8. REFERENCES

[1] S. Bates and S. Horwitz. Incremental program testing using program dependence graphs. In *Conference Record of the Twentieth ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 384–396, Charleston, South Carolina, Jan.10–13, 1993. ACM Press.

[2] D. Binkley. Reducing the cost of regression testing by semantics guided test case selection. In *ICSM '95: Proceedings of the International Conference on Software Maintenance*, page 251, Washington, DC, USA, 1995. IEEE Computer Society.

[3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002.

[4] Y. Collette and P. Siarry. *Multiobjective Optimization: Principles and Case Studies*. Springer, 2004.

[5] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, Chichester, UK, 2001.

[6] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.

[7] H. Do, S. G. Elbaum, and G. Rothermel. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering: An International Journal*, 10(4):405–435, 2005.

[8] S. G. Elbaum, A. G. Malishevsky, and G. Rothermel. Prioritizing test cases for regression testing. In *International Symposium on Software Testing and Analysis*, pages 102–112. ACM Press, 2000.

[9] M. J. Harrold, R. Gupta, and M. L. Soffa. A methodology for controlling the size of a test suite. *ACM Trans. Softw. Eng. Methodol.*, 2(3):270–285, 1993.

[10] L. W. H.K.N. Leung. Insight into regressin testing. In *Proceedings of the Conference on Software Maintenance*, October 1989.

[11] M. Hutchins, H. Foster, T. Goradia, and T. Ostrand. Experiments on the effectiveness of dataflow- and control-flow-based test adequacy criteria. In *Proceedings of the 16th International Conference on Software Engineering*, pages 191–200. IEEE Computer Society Press, May 1994.

[12] Z. Li, M. Harman, and R. Hierons. Meta-heuristic search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*. To Appear.

[13] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum. Cost-cognizant test case prioritization. Technical report, Department of Computer Science and Engineering, University of Nebraska-Lincoln, March 2006.

[14] N. Nethercote and J. Seward. Valgrind: A program supervision framework. In *Proceedings of the Third Workshop on Runtime Verification*, Colorado, USA, July 2003. Boulder.

[15] G. Rothermel and M. J. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529–551, Aug. 1996.

[16] G. Rothermel, M. J. Harrold, J. Ostrin, and C. Hong. An empirical study of the effects of minimization on the fault detection capabilities of test suites. In *ICSM*, pages 34–43, 1998.

[17] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold. Test case prioritization: An empirical study. In *Proceedings; IEEE International Conference on Software Maintenance*, pages 179–188, Los Alamitos, California, USA, 1999. IEEE Computer Society Press.

[18] F. Szidarovsky, M. E. Gershon, and L. Dukstein. *Techniques for multiobjective decision making in systems management*. Elsevier, New York, 1986.

[19] P. Tonella, P. Avesani, and A. Susi. Using the case-based ranking methodology for test case prioritization. In *ICSM '06: Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 123–133, Washington, DC, USA, 2006. IEEE Computer Society.

[20] K. R. Walcott, M. L. Soffa, G. M. Kapfhammer, and R. S. Roos. Time aware test suite prioritization. In *ISSTA '06: Proceedings of the 2006 international symposium on Software testing and analysis*, pages 1–12, New York, NY, USA, 2006. ACM Press.

[21] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur. Effect of test set minimization on fault detection effectiveness. *Software — Practice and Experience*, 28(4):347–369, 1998.

[22] S. S. Yau and Z. Kishimoto. A method for revalidating modified programs in the maintenance phase. In *Proceedings of 11th International Computer Software and Applications Conference (COMPSAC '87)*, pages pp. 272–277, October 1987.