Search–Based Approaches to the Component Selection and Prioritization Problem

Mark Harman
King's College
Strand, London
WC2R 2LS, UK.

Alexandros Skaliotis
King's College
Strand, London
WC2R 2LS, UK.

Kathleen Steinhöfel
King's College
Strand, London
WC2R 2LS, UK.

{mark.harman, alexandros.skaliotis, kathleen.steinhofel}@kcl.ac.uk

ABSTRACT

This poster paper addresses the problem of choosing sets of software components to combine in component–based software engineering. It formulates both ranking and selection problems as feature subset selection problems to which search based software engineering can be applied. We will consider selection and ranking of elements from a set of software components from the component base of a large telecommunications organisation.

Categories and Subject Descriptors

D.2 [**Software Engineering**]: Miscellaneous

General Terms

Management

Keywords

Feature Subset Selection, Prioritization, Search

1. INTRODUCTION

Software engineering is becoming less and less associated with the development, *ab initio*, of a single software system and more and more a component-based activity, in which systems are constructed from a set of components to meet a set of constraints. These components may be pre-existing (some in-house while others are provided by third parties) or planned.

The problem faced by managers working with component–based software engineering is that of determining which set of components to use in order to balance the trade offs inherent in the finished product. A generic instantiation of this problem finds the manager considering several candidate components for which the data available includes estimates of the cost of acquisition (third party purchase or in-house development), customer desirability, development time and expected revenue. The manager may also have information about the dependencies between components and may wish to include other factors in the decision making process, such as the priority given to each of the customers.

From the set of all components, the manager must pick a subset which balances these competing concerns in the best way possible. This is clearly an optimization problem. For systems with more than a few simple components the search space is unmanageably large and complex, with the consequence that no manager can be expected to find optimal choices that balance the constraints without some form of automated support. This is the 'Component Selection Problem'. A closely related problem is the 'Component Prioritization Problem', in which the manager seeks to order the components under consideration into a priority ranking. The Component Selection Problem helps the manager decide which component combinations will make sensible choices for products, while the Component Prioritization Problem helps with planning the order in which components should be tackled first when planning the software development process.

This poster paper formulates both problems in terms of a series of feature subset selection problems such that automated approaches employing search based software engineering can be developed and tested in future work.

GECCO'06, July 8–12, 2006, Seattle, Washington, USA.
ACM 1-59593-186-4/06/0007.

2. PROBLEM STATEMENT

In the application domain reported upon here, we consider a set of software components from the component base of a large telecommunications organisation. These components are independent 'add-ons' to the base system, i.e., they determine special, additional features or functions of the software. Each component is characterised by a set of values, including estimates of the cost of acquisition, customer desirability, development time and expected revenue. We associate a weight (or score) and a cost with each component where we combine the cost of acquisition and development time to a single cost value c_i, and customer desirability and expected revenue to a weight value w_i, where i is an index of the components.

In the feature or component selection problem, we are given a set of such components and want to determine a subset that maximises the total sum of weights while minimizing the total cost of the selected components.

This problem, like most realistic optimisation problems, requires the simultaneous optimisation of more than one objective function. Similar to the traditional portfolio optimisation problem that attempts to simultaneously minimize the risk and maximise the fiscal return, we need to find some trade-off between the criteria to ensure a satisfactory design.

Such multiobjective problems can be solved by combining

the multiple objectives into one scalar objective. Another approach is to bound one criteria while optimising the other. This yields solutions that are optimum and nondominated; there are no other solutions superior in all objectives. These, so–called Pareto optimal solutions, plot a pareto optimum curve that gives the best possible insight into trade-off solutions.

With bounding the total sum of cost to K, we can formulate the component selection problem for a particular given bound of total cost as an optimisation 0-1 knapsack problem:

$$\begin{array}{lll} \text{maximise} & \sum_{j=1}^{n} & w_j x_j \\ \text{subject to} & \sum_{j=1}^{n} & c_j x_j \leq K, \qquad x_j \in \{0,1\}. \end{array}$$

The knapsack problem is known to be NP-hard. However, it can be solved by a pseudo-polynomial algorithm using dynamic programming [6]. The algorithm runs in $O(n^2 w)$ time and therefore depends on the optimum value for w that can be found within K. One can transform this algorithm into a polynomial-time approximation scheme (PTAS) by truncating the last t decimal digits of all w_i.

However, for this application to software engineering, where choices of components can have a significant impact on the organisation's profitability, we are interested in solutions as close as possible to the pareto optimum. Therefore, we propose to solve the individual knapsack problems by employing search based heuristics that find optimum or close to optimum solutions in order to sample the pareto curve as closely as possible.

Providing the manager with the pareto curve enables the expert to look at the trade-off between cost and revenue of subsets of components. From an analysis of components that are common to interesting points on the pareto curve, representing subsets of components, one can develop a procedure for prioritization.

3. RELATED WORK

General ordering and feature subset selection problems have been widely studied in the operations research, metaheuristic search and evolutionary computation communities [6]. The work reported in the present poster paper on feature subset selection and ordering (ranking) follows this general pattern, but is most closely related to the previous work on ordering and selection problems in Search Based Software Engineering (SBSE). This previous SBSE work has concerned regression testing (selection and prioritization), project planning (ordering and selection) and requirements analysis (selection).

Much of the work on test case prioritization [7, 9] has concerned the application of greedy algorithms. Wong et al. [9] presented a technique that combines test set minimization and prioritization to select test cases, according to the criterion of 'increasing cost per additional coverage'.

The work reported in the present poster paper is related to this work because it is concerned with the application of selection and prioritization to software engineering problems, but the two problem domains — regression selection / prioritization and component selection / prioritization — occur at opposite ends of the traditional software development life-cycle.

Chicano and Alba [3] use search techniques for the software project planning problem. In their approach, the problem is formulated as a multi objective search problem in which each objective is combined into a single fitness function by means of weights applied to each sub-fitness function.

One of the first papers that presented a solution to a software engineering problem as a feature subset selection search was the work on the 'Next Release Problem' by Bagnall et al. [1]. In this work, the problem was to determine the set of requirements that should be included in the next release of a software system.

The work in the present poster paper is the first to characterise the problem of component selection and prioritization as a problem in search based software engineering and to present results from the application of search to a real world instantiation of this problem.

4. CONCLUSION

This poster paper shows how ideas from search based software engineering can be applied to problems of software component selection and ranking.

5. ADDITIONAL AUTHORS

Paul Baker (Motorola Labs, Viables Estate, Basingstoke, UK. email: `paul.baker@motorola.com`)

6. REFERENCES

[1] A. Bagnall, V. Rayward-Smith, and I. Whittley. The next release problem. *Information and Software Technology*, 43(14):883–890, Dec. 2001.

[2] A. W. Brown, editor. *Component-Based Software Engineering*. IEEE Press, 1997.

[3] F. Chicano and E. Alba. Management of software projects with gas. In 6^{th} *Metaheuristics International Conference (MIC2005)*, Vienna, Austria, Aug. 2005.

[4] J. Clark, J. J. Dolado, M. Harman, R. M. Hierons, B. Jones, M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings — Software*, 150(3):161–175, 2003.

[5] M. Harman and B. F. Jones. Search based software engineering. *Information and Software Technology*, 43(14):833–839, Dec. 2001.

[6] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, 1998.

[7] G. Rothermel, S. Elbaum, A. G. Malishevsky, P. Kallakuri, and X. Qiu.. On test suite composition and cost-effective regression testing. *ACM Trans. Softw. Eng. Methofol.*, 13(3):277-331, 2004.

[8] M. J. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

[9] W. E. Wong, J. R. Horgan, S. London, and H. A. Bellcore. A study of effective regression testing in practice. In *ISSRE '97: Proceedings of the Eighth International Symposium on Software Reliability Engineering (ISSRE '97)*, page 264. IEEE Computer Society, 1997.