Information and Software Technology 86 (2017) 20–36

Contents lists available at ScienceDirect

Information and Software Technology

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

Handling constraints in combinatorial interaction testing in the presence of multi objective particle swarm and multithreading

Bestoun S. Ahmed[a,b,*], Luca M. Gambardella[a], Wasif Afzal[c], Kamal Z. Zamli[d]

[a] Istituto Dalle Molle di Studi sullIntelligenza Artificiale (IDSIA), CH-6928 Manno-Lugano, Switzerland
[b] Department of Computer Science, Faculty of Electrical Engineering, Czech Technical University, Karlovo nam. 13, 121 35 Praha 2, Czech Republic
[c] School of Innovation, Design and Engineering, Mlardalen University, Sweden
[d] Faculty of Computer Systems and Software Engineering, University Malaysia Pahang, Gambang, Malaysia

ARTICLE INFO

Article history:
Received 13 February 2016
Revised 14 February 2017
Accepted 14 February 2017
Available online 20 February 2017

Keywords:
Constrained combinatorial interaction
Multi-objective particle swarm optimisation
Test generation tools
Search-based software engineering
Test case design techniques

ABSTRACT

Context: Combinatorial testing strategies have lately received a lot of attention as a result of their diverse applications. In its simple form, a combinatorial strategy can reduce several input parameters (configurations) of a system into a small set based on their interaction (or combination). In practice, the input configurations of software systems are subjected to constraints, especially in case of highly configurable systems. To implement this feature within a strategy, many difficulties arise for construction. While there are many combinatorial interaction testing strategies nowadays, few of them support constraints.
Objective: This paper presents a new strategy, to construct combinatorial interaction test suites in the presence of constraints.
Method: The design and algorithms are provided in detail. To overcome the multi-judgement criteria for an optimal solution, the multi-objective particle swarm optimisation and multithreading are used. The strategy and its associated algorithms are evaluated extensively using different benchmarks and comparisons.
Results: Our results are promising as the evaluation results showed the efficiency and performance of each algorithm in the strategy. The benchmarking results also showed that the strategy can generate constrained test suites efficiently as compared to state-of-the-art strategies.
Conclusion: The proposed strategy can form a new way for constructing of constrained combinatorial interaction test suites. The strategy can form a new and effective base for future implementations.

1. Introduction

In the last decade, various studies on combinatorial interaction approaches have gained a lot of awareness and several test generation approaches were developed. In software engineering, combinatorial interaction testing (CIT) aims to generate minimised test suites that manipulate the variables of input parameters based on their combination. Each combination could form a specific configuration of the software-under-test (SUT). The goal is to cover all possible t-combinations (sometimes called $t-tuples$) by an optimised set (where t is the interaction strength) [1]. This could be a difficult task in case of highly configurable systems, which leads to combinatorial explosion and non-deterministic polynomial-time hard (NP-hard) problems [2]. In addition, the problem of constrained interactions has recently appeared [3].

Nowadays, software development is shifted from building an isolated, product-by-product approach to software product lines (SPL) [4]. In addition to the many features this approach provides like minimising the cost and market reachability time, it facilitates the idea of customisable software products. With this approach, there are many customisable features that could be added to or extracted from the functionality of a specific software based on the needs of the developers or customers. Eclipse represents a well-known example of SPL in which many functions and plug-ins (i.e., having different features) can be added or extracted from its main framework [5]. Evidence showed that most of the faults may result due to the interaction among these features [1,6]. Hence, all the interactions must be tested carefully. However, in reality, there are many constraints among the features. Some features must or must not appear with others and they may be included or excluded from the test suite during the testing process. CIT strategies can tackle these interactions efficiently, however, with ordinary

* Corresponding author. Tel.: +420 224 355 752.
E-mail addresses: albeybes@fel.cvut.cz, bestoon82@gmail.com (B.S. Ahmed), luca@idsia.ch (L.M. Gambardella), wasif.afzal@mdh.se (W. Afzal), kamalz@ump.edu.my (K.Z. Zamli).

http://dx.doi.org/10.1016/j.infsof.2017.02.004
0950-5849/

strategies, it is not possible to satisfy all the constraints and they may contain some invalid combinations in the final constructed test suite.

There are many approaches to construct CIT test suites in the literature. Among those approaches, evidences revealed that the use of meta-heuristic algorithms could achieve an optimum or near-optimum combinatorial set, covering every possible interaction of input parameters (or functions) [7,8]. Most recently, different meta-heuristic algorithms have been adapted to solve this problem such as Simulated Annealing (SA) [9], Genetic Algorithms (GA) [10], Tabu Search (TS) [11], Ant Colony Algorithm (ACA) [12], Cuckoo Search (CS) [1], and many other algorithms. Despite the wide range of approaches and algorithms used in generating the combinatorial interaction set, there is no "universal" strategy that can generate optimised sets for all configurations since this problem is NP-hard problem [13]. Hence, each strategy could be useful for specific kinds of configurations and applications.

Although different strategies have been developed, the problem of search space complexity is still the same. As mentioned earlier, the main aim of CIT strategies is to cover entire interactions of input parameters by using smallest set. Hence, the strategy needs to search for a combination that can cover much of those interactions. To determine the number of interactions covered, the strategy must search for them among a large number of interactions which will definitely consume the program run time as well as other resources. It will likewise cause the program to take more iteration for searching within the meta-heuristic algorithm. In addition to this problem, nowadays SPLs have many features to customise that leads to many input parameters for CIT strategy. The problem of generating input parameters combination represents another serious problem, given consumption of time and resources. This problem appears clearly as input parameters continue to grow in size, since generally, most of the algorithms complexities are growing alongside the number of parameters. To overcome this problem, a special algorithm is needed to be combined with efficient data structures in order to speed up the generation and sorting process. In addition to these existing challenges to the implemented strategies, a few of them can satisfy the constraints in the final generated test suite. This will add extra complexity in designing efficient CIT strategies.

In our earlier research [1,14,15], we have examined Particle Swarm Optimisation (PSO) within a CIT strategy to generate ordinary combinatorial test suites. The strategy has been modified and implemented also for the same purpose by other researchers recently [16]. In both cases, PSO has outperformed other strategies in different experiments. However, none of these researches are suitable for the constrained CIT. Adding this feature to the strategy will change the nature of the problem and the search space itself. It also changes the fitness function of the optimisation algorithm. In addition, the aforementioned problems within those strategies, are still not solved properly.

In order to solve these problems and to cope with the practical test generation process, this paper presents a new strategy that tries to generate constrained combinatorial interaction test suites efficiently and provides a new approach for the design and implementation of these strategies. Owing to the nature of constrained combinatorial generation, multi-objective PSO is used within the strategy. Furthermore, the strategy expands our earlier research and gives more practical results depending on the new algorithms that we have designed and implemented.

The rest of this paper is organised as follows. Section 2 illustrates a practical model of the problem as a motivating example, using an SPL case study. Section 3 presents the mathematical notations, definitions, and theories behind the CIT and constrained CIT. Section 4 summarises recent related works and reviews the existing literature. Section 5 reviews multi-objective PSO and discusses its features for this research. Section 6 discusses the design concepts of the strategy and its implementation. The section also illustrates how we adapt different algorithms for constrained CIT. Section 7 contains the results of different stages of the evaluation process. Section 8 discusses the experimental results. Section 9 shows some threats to validity in this research. Finally, Section 10 concludes the paper.

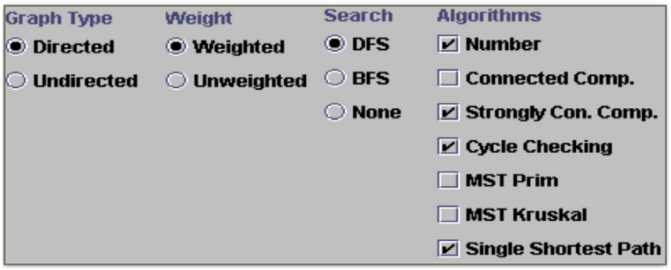

Fig. 1. GPL specification GUI.

2. Motivating example

To illustrate the constrained combinatorial interaction in practice, we adopt a real canonical example from Software Product Lines (SPL) called Graph Product Line (GPL) [17]. GPL is a configurable system in which the combinations lead to a product with basic graph algorithms and graph types [18]. It is implemented in a way that all the applications that come out as a product do not have the same combination of features. Fig. 1 shows a GUI with all specifications of the GPL.

As can be seen in Fig. 1, the produced graph type could be Directed or Undirected, and the edges of the graph could be Weighted or Unweighted (i.e., non-negative numbers). In addition to these two features, the graph needs at least one search algorithm which can be a breadth-first search (BFS) or a depth-first search (DFS). Finally, the graph needs one or more of the following algorithms: Vertex Numbering (Number), Connected Components (Connected), Strongly Connected Components (Strongly Connected), Cycle Checking (Cycle), Minimum Spanning Tree (MST Prim, MST Kruskal), and Single-Source Shortest Path (Shortest). Fig. 2 shows a feature model for these graph features.

The feature model in Fig. 2 shows a standard model of the illustrated features in Fig. 1. The graph illustrates that the product in the root feature (GPL) has four core functionalities and one driver program (Driver). The driver chooses the example from Benchmark feature to apply the graph Algorithms to. Each of the functionalities (Graph Type) and (Search) have two alternative features. Finally, one of the following algorithms must be followed with the product: connected components (CC), numbering of nodes in the traversal order (Num), cycle checking (Cycle), strongly connected components (SCC), shortest path (Shortest), Kruskals algorithm (Kruskal), or minimum spanning trees with Prims algorithm (Prim).

To apply the CIT method on this feature model, the parameters and values must first be specified. Here, the four functions become parameters and their features become values for these parameters. Table 1 illustrates this.

The arrangement in Table 1 shows clearly that there are $2 \times 2 \times 2 \times 7$ possible feature configurations, which equals 57 possibilities. We can reduce this exhaustive test suite by taking an interaction of the features among the functions. In addition to the reduction advantage, this can tackle the faults caused by the interaction of these features. This idea is supported by much evidence in the literature in which it was shown that faults are likely to oc-