Applied Soft Computing 44 (2016) 57–74

Contents lists available at ScienceDirect

Applied Soft Computing

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

A Tabu Search hyper-heuristic strategy for *t-way* test suite generation

Kamal Z. Zamli [a,*], Basem Y. Alkazemi [b], Graham Kendall [c]

[a] IBM Centre of Excellence, Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, Lebuhraya Tun Razak, 26300 Kuantan, Pahang Darul Makmur, Malaysia
[b] College of Computer and Information Systems, Umm Al-Qura University, Saudi Arabia
[c] School of Computer Science, University of Nottingham Malaysia Campus, Jalan Broga, 43500 Semenyih, Selangor Darul Ehsan, Malaysia

ARTICLE INFO

Article history:
Received 26 September 2015
Received in revised form 17 March 2016
Accepted 18 March 2016
Available online 4 April 2016

Keywords:
Software testing
t-way Testing
Hyper-heuristic
Particle Swarm Optimization
Cuckoo Search Algorithm
Teaching Learning based Optimization
Global Neighborhood Algorithm

ABSTRACT

This paper proposes a novel hybrid *t-way* test generation strategy (where *t* indicates interaction strength), called High Level Hyper-Heuristic (HHH). HHH adopts Tabu Search as its high level meta-heuristic and leverages on the strength of four low level meta-heuristics, comprising of Teaching Learning based Optimization, Global Neighborhood Algorithm, Particle Swarm Optimization, and Cuckoo Search Algorithm. HHH is able to capitalize on the strengths and limit the deficiencies of each individual algorithm in a collective and synergistic manner. Unlike existing hyper-heuristics, HHH relies on three defined operators, based on improvement, intensification and diversification, to adaptively select the most suitable meta-heuristic at any particular time. Our results are promising as HHH manages to outperform existing *t-way* strategies on many of the benchmarks.

1. Introduction

Interaction (*t-way*) testing is a methodology to generate a test suite for detecting interaction faults. The generation of a *t-way* test suite is an *NP* hard problem [1]. Many *t-way* strategies have been presented in the scientific literature. Some early algebraic *t-way* strategies exploit exact mathematical properties of orthogonal arrays. These *t-way* strategies are often fast and produce optimal solutions, yet they impose restrictions on the supported configurations and interaction strength. Computational *t-way* strategies remove such restrictions, allowing for the support of arbitrary configurations at the expense of producing (potentially) non-optimal solution.

By formulating interaction testing as an optimization problem, recent efforts have focused on the adoption of meta-heuristic algorithms as the basis for *t-way* strategies. Search Based Software Engineering (SBSE) [2–4], is a relatively new field that has proposed meta-heuristic based *t-way* strategies (e.g. Genetic Algorithms (GA) [5], Particle Swarm Optimization (PSO) [6,7], Harmony Search Algorithm (HS) [8], Ant Colony Algorithm (ACO) [5], Simulated Annealing (SA) [9,10] and Cuckoo Search (CS) [11]). The adoption of these meta-heuristic based strategies appears to be effective for obtaining good quality solutions, as reported in benchmarking experiments related to *t-way* testing [7,8]. Nevertheless, as suggested by the *No Free Lunch theorem* [12], no single meta-heuristic can outperform all others even over different instances of the same problem. For this reason, hybridization of meta-heuristics can be the key to further enhance the performance of *t-way* strategies. Since each meta-heuristic has its own advantages, meta-heuristic hybridization is beneficial for compensating the limitation of one with the strengths of another. In fact, the best results of many optimization problems are often obtained by hybridization [13].

In this paper we explore the hybridization of meta-heuristics based on a hyper-heuristic approach. We present a new *t-way* testing strategy. Specifically, our contributions can be summarized as follows:

- A novel hyper-heuristic based strategy, which we have termed High Level Hyper-Heuristic (HHH), for general combinatorial *t-way* test suite generation. HHH employs Tabu Search (TS) as its high level meta-heuristic (HLH) and leverages on the strength of four low level meta-heuristics (LLH), comprising Teaching Learning based Optimization (TLBO) [14], Global Neighborhood Algorithm (GNA) [15], Particle Swarm Optimization (PSO) [16], and Cuckoo Search Algorithm (CS) [17]. To the best of our

* Corresponding author.
E-mail addresses: kamalz@ump.edu.my (K.Z. Zamli), bykazemi@uqu.edu.sa (B.Y. Alkazemi), Graham.Kendall@nottingham.edu.my (G. Kendall).

http://dx.doi.org/10.1016/j.asoc.2016.03.021
1568-4946/

knowledge, HHH is the first hyper-heuristic based strategy that addresses the problem of *t-way* test suite generation.
- A new hyper-heuristic approach for the meta-heuristic selection and acceptance mechanism based on three operators (i.e. improvement, diversification and intensification) that are integrated into the Tabu Search HLH. As the name suggests, the improvement operator checks for improvements in the objective function. The diversification operator measures how diverse the current and the previously generated solution are against the population of potential candidate solutions. Finally, the intensification operator evaluates how close the current and the previously generated solution are against the population of solutions.

The rest of the paper is organized as follows. Section 2 presents an overview of hyper-heuristics and the mathematical and theoretical foundation for *t-way* testing. Section 3 reviews the state-of-the-art for *t-way* test case generation strategies. Section 4 presents the design and implementation of HHH. Section 5 describes the calibration of HHH. Section 6 evaluates HHH against existing strategies and Section 7 debates the usefulness of HHH. Section 8 elaborates on threats to validity. Finally, Section 9 presents our conclusion.

2. Theoretical framework

2.1. Overview of hyper-heuristics

To put hyper-heuristics into perspective, consider the different possible options for utilizing and combining meta-heuristic algorithms (see Fig. 1). The first option, shown for completeness, is a standard meta-heuristic algorithm and can be ignored from our discussion as we want to focus on hybridization methodologies. These are shown in remaining figures, that is a hybrid meta-heuristic and a hyper-heuristic.

Hybrid meta-heuristics can be low-level or high level hybridizations [13]. Low-level hybridization combines two or more algorithms. High-level hybridization retains the original meta-heuristic, which can run independently (e.g. either in sequence or parallel) without any connection amongst the meta-heuristics involved (i.e. it operates as a black box).

Hyper-heuristics could be seen as a hybrid meta-heuristic owing to the integration of more than one meta-heuristic algorithm (refer to Fig. 1). However, unlike a typical hybrid meta-heuristic, hyper-heuristics (or *(meta)-heuristic to choose (meta)-heuristics* [18–20]) adopts a high level meta-heuristic (HLH) to adaptively select from a set of low level meta-heuristics (LLHs), which are applied to the problem at hand. The LLHs communicate with the HLH through a domain barrier to relay the feedback of the quality of the current solution. Only the LLHs have domain knowledge, meaning that the HLH is a general algorithm that can be utilized for different problems without any algorithmic changes. It is only required to supply a different set of LLHs. In fact, the LLHs can also be formed from (low-level or high-level) hybrid meta-heuristics themselves.

Recent developments have introduced hyper-heuristics that are able to automatically generate the LLHs, whereby the end user does not have to implement a set of LLHs for each problem domain. Moreover, the HLH is also able to evolve its own selection and acceptance criteria [21,22].

2.2. The t-way test generation problem

Consider a hypothetical example of a mobile phone product configuration. The product configuration has four features (or parameters): Call Options, Message Types, Media, and Screen. Each parameter takes three possible values (e.g. Call Options = {Voice Calls, Video Calls, Both Voice and Video Calls}, Message Types = {Text, Video, Image}, Media = {Camera, Radio, Video Player}, and Screen = {Basic Colors, High Resolution, Black and White}). The pairwise (2-way) test generation for the Mobile Product Configuration can be seen in Fig. 2 with nine test cases. The mapping of the corresponding tests can be achieved (row-wise) from the 2-way representation based on the defined parameter values (column-wise) as depicted in Table 1. It should be noted that all the 2-way interaction tuples between parameters are covered at-least once.

Mathematically, the *t-way* test generation problem can be expressed by Eq. (1).

$$f(Z) = |\left\{I \text{ in VIL:} Z \text{ covers} I\right\}| \quad (1)$$

$$\text{Subject to} Z = Z_1, Z_2, \ldots, Z_i, \text{in} P_1, P_2, \ldots\ldots, P_i; i = 1, 2, \ldots, N$$

where, $f(Z)$ is an objective functions (or the fitness evaluation), Z (i.e., the test case candidate) is the set of decision variables Z_i, VIL is the set of non-covered interaction tuples (I), the vertical bars $|\cdot|$ represent the cardinality of the set and the objective value is the number of non-covered interaction tuples covered by Z, P_i is the set of possible range of values for each decision variable, that is, P_i = discrete decision variables $(Z_i(1) < Z_i(2) < \ldots\ldots < Z_i(K))$; N is the number of decision variables (i.e. parameters); and K is the number of possible values for the discrete variables.

2.3. The covering array notation

In general, *t-way* testing has strong associations with the mathematical concept of covering arrays (CA). For this reason, *t-way* testing often adopts CA notation for representing *t-way* tests [23]. The notation $CA_\lambda(N; t, k, v)$ represents an array of size N with v values, such that every $N \times t$ sub-array contains all ordered subsets from the v values of size t at least λ times [24,25], and k is the number of components. To cover all t-interactions of the components, it is normally sufficient for each component to occur once in the CA. Therefore, with $\lambda = 1$, the notation becomes $CA(N; t, k, v)$. When the CA contains a minimum number of rows (N), it can be considered an optimal CA according to the definition in Eq. (2) [26].

$$CAN(t, k, v) = \min\{N : \exists CA_\lambda(N; t, k, v)\} \quad (2)$$

To improve readability, it is customary to represent the covering array as CA (N; t, k, v) or simply CA (N; t, v^k). Using our earlier example of the mobile phone product configuration in Fig. 2, the test suite can be represented as CA (9; 2, 3^4). In the case when the number of component values varies, this can be handled by Mixed Covering Array (MCA) (N; $t, k, (v_1, v_2, \ldots v_k)$) [27]. Similar to covering array, the notation can also be represented by MCA (N; t, k, v^k). For example, MCA (9; 2, 3^2 2^2) represents a test suite of size nine for a system with four components (two components having three values and two components having two values) covering two-way interactions. Fig. 3 illustrates the two aforementioned CA and MCA arrangements respectively.

Having described the theoretical framework, the following section surveys the existing studies on *t-way* strategies in order to reflect the current progress and achievements in the scientific literature.

3. Existing literature on *t-way* strategies

Generally, *t-way* strategies can be classified as algebraic or computational approaches [28,29]. Algebraic approaches are often based on the extensions of the mathematical methods for constructing Orthogonal Arrays (OAs) [30,31]. Examples of strategies