Expert Systems With Applications xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use

Thair Mahmoud [a], Bestoun S. Ahmed [b,*]

[a] School of Engineering, Edith Cowan University, 270 Joondalup Drive, Joondalup, WA 6027, Australia
[b] Software Engineering Department, Engineering College, Salahaddin University-Hawler (SUH), 44002 Erbil, Kurdistan

ARTICLE INFO

Keywords:
Covering array
Fuzzy logic
Combinatorial design
Particle swarm optimization
Software testing
Test generation tools
Search-based software engineering

ABSTRACT

Recent research activities have demonstrated the effective application of combinatorial optimization in different areas, especially in software testing. Covering array (CA) has been introduced as a representation of the combinations in one complete set. $CA_{\lambda}(N; t, k, v)$ is an $N \times k$ array in which each t-tuple for an $N \times t$ sub array occurs at least λ times, where t is the combination strength, k is the number of components (factors), and v is the number of symbols for each component (levels). Generating an optimized covering array for a specific number of k and v is difficult because the problem is a non-deterministic polynomial-time hard computational one. To address this issue, many relevant strategies have been developed, including stochastic population-based algorithms. This paper presents a new and effective approach for constructing efficient covering arrays through fuzzy-based, adaptive particle swarm optimization (PSO). With this approach, efficient covering arrays have been constructed and the performance of PSO has been improved for this type of application. To measure the effectiveness of the technique, an empirical study is conducted on a software system. The technique proves its effectiveness through the conducted case study.

1. Introduction

Combinatorial testing (CT) based on combinatorial design has been researched extensively in the software field over the past decade, and studies focus on identifying the applications of this method. Moreover, combinatorial design and optimization have been applied as a sampling technique (e.g., Sahib, Ahmed, & Potrus, 2014; Sulaiman & Ahmed, 2013; Yilmaz et al., 2014) given its effectiveness and usefulness in the software testing field (e.g., Ahmed, Zamli, & Lim, 2012a; Barrett & Dvorak, 2009; Qu, Cohen, & Woolf, 2007). A covering array (CA) is a mathematical representation of the combinations in one complete set. In this set, every t-combination of the input must be covered at least once by the CA (Zhanga, Yan, Zhao, & Zhang, 2014). Hence, the finite set of elements is arranged into patterns (subsets, words, and arrays) according to specific rules (Ahmed, Abdulsamad, & Potrus, 2015; Yilmaz et al., 2014).

A CA is difficult to generate because this issue is a non-deterministic polynomial-time (NP)-hard computational problem. Computation time and problem complexity increase exponentially with an increase in the number of input parameters (Nie & Leung, 2011). In addition, no unique arrangement and size is set for the array. To solve this problem, researchers have adopted artificial intelligent optimization theories.

Strategies based on simulated annealing (SA) (Cohen, 2004), ant colony algorithm (ACA) (Chen, Gu, Li, & Chen, 2009), tabu search (TS) (Nurmela, 2004), genetic algorithm (GA) (Shiba, Tsuchiya, & Kikuno, 2004), and particle swarm optimization (PSO) (Ahmed & Zamli, 2011b; Ahmed et al., 2012a; Chen, Gu, Qi, & Chen, 2010) can effectively generate CAs in optimized sizes. Due to its robustness and simplicity, PSO efficiently produces CAs for different experimental sets; however, our experience and that of other researchers is that PSO is prone to parameter tuning problems (Ahmed et al., 2012a; Lessmann, Caserta, & Arango, 2011). As per an analysis of PSO search performance in optimising the structures for CAs, the stochastic approach in PSO can be enforced further with knowledge-based rules to automatically shift the path of this approach to the correct direction. This hypothesis is developed based on the literature postulating that PSO performance is mainly dependent on the values of search adaptation parameters. In other words, PSO combines the two roles of searching mechanisms, namely, exploration and exploitation. In the former, PSO performs global optimum solution searching; in the latter, PSO seeks accurate optimum solutions by converging the search around a promising candidate. For instance, the determination of appropriate values for these parameters should be based on a compromise between the local and global explorations that facilitate

* Corresponding author. Tel.: +964 750 1725998.
E-mail addresses: t.mahmoud@ecu.edu.au (T. Mahmoud), bestoon82@gmail.com, bestoon82@yahoo.com (B.S. Ahmed).

http://dx.doi.org/10.1016/j.eswa.2015.07.029
0957-4174/

accelerated convergence. Evidence shows that depending on problem complexity, different parameter values are required to obtain the optimum solution (Huimin, Qiang, & Zhaowei, 2014; Xu, 2013). This issue can be solved by supporting the PSO algorithm with a mechanism that adapts the parameters to process scenarios and thus controls optimization performance. This strategy utilizes a Mamdani-type fuzzy inference system (FIS), and the FIS parameters can be designed to suit the optimization problem to be solved. In the current work, three FIS designs are proposed to tune the three main PSO parameters. The proposed FISs are built to monitor PSO performance and adjust the parameters to overcome optimization process problems, thus improving efficiency. This approach is applied to CA generation.

Thus, the contributions of this research are threefold. First, the proposed methodology is intended to overcome the drawback of parameter tuning in the conventional PSO algorithm when generating CA structures by employing a set of rules to monitor PSO performance. As a result, CAs with improved size can be generated. Second, the monitoring mechanism proposed in this work is applied via fuzzy logic. A specific rule-based system is established and a membership function design is customized to improve CA generation efficiency. This mechanism can be applied to detect faults within an artifact effectively for software mutation testing. Third, the implementation of this methodology establishes a unified strategy for CA generation. This methodology includes adding the fuzzy logic-based adaptive PSO to a set of other algorithms that can automatically generate arrays and configure the relevant PSO covering structure accordingly.

On this basis, the paper is organized as follows: Section 2 presents the theoretical backgrounds, mathematical notations, and definitions of the CA. Section 3 reviews relevant literature and highlights the most important findings. Section 4 introduces and provides an overview of PSO. Section 5 introduces in detail and justifies the proposed strategy with the design and implementation, including the appropriate algorithms. Section 6 presents the evaluation results. Section 7 lists the threats to the validity of the conducted experiments. Finally, Section 8 provides the concluding remarks.

2. CA mathematical preliminaries and notations

CAs first appeared as a generalization of orthogonal arrays. An orthogonal array OAλ (t, k, g) is an array with index λ that has strength t, k factors, and g levels. For a set of columns $B = \{b0, \ldots , b\ t{-}1\} \subseteq \{0, \ldots , k-1\}$, we say that B is λ-covered if the N $\times$ s sub array over the columns of B has each t-tuple over v as a row at least λ times. The λ parameter is often omitted when $\lambda = 1$ (Cheng, 1980). Orthogonal Arrays have been used in the literature in the design of experiments by taking each row in the array as a test case. The main drawback of the OA is its limited usefulness in this application since it requires the factors and levels to be uniform (Beizer, 1990; Ronneseth & Colbourn, 2009). To address this limitation, the Covering Array (CA) has been introduced to complement OA.

A Covering Array $CA_\lambda(N; t, k, v)$ is an $N \times k$ array over $\{0, \ldots , v-1\}$ such that every $B \in \binom{\{0,\ \ldots\ ,\ k-1\}}{t}$ is λ-covered such that every $N \times t$ sub-array contains all ordered subsets from v values of size t at least λ times (Nie et al., 2015; Yilmaz et al., 2014). For optimality, we normally want t-tuples to occur at least once. As such, we consider the value of $\lambda = 1$, which is often omitted. Hence the notation becomes $CA(N;t,k,v)$ (Hartman & Raskin, 2004). We say that the array has size N, strength t, k factors, v levels, and index λ. Given t, k, v, and λ, the smallest N for which a $CA_\lambda(N; t, k, v)$ exists is denoted $CAN_\lambda(t, k, g)$. A $CA_\lambda(N; t, k, v)$ with $N = CAN_\lambda(t, k, v)$ is said to be optimal.

One serious problem in CA is that the levels for each input factor are considered to be uniform. In other words, each input factor must have equal numbers of levels. However, most of the time, the input-factors have different levels in practice. For this case, mixed level covering array (MCA) is notated. A mixed level covering array, MCA $(N; d, k, (v_1, v_2, \ldots v_k))$, is an $N \times k$ array on v levels, where the rows of each $N \times d$ sub-array cover and all d-tuples of values from the d columns occur at least once (Xiao, Cohen, & Woolf, 2007). For more flexibility in the notation, the array can be presented by MCA $(N; d, v^k)$) and can be used for a fixed-level CA, such as CA $(N ;d, v^k)$ (Lei et al., 2008).

3. Review of literature and related work

CA generation is an NP-hard problem; hence, methods for addressing this issue effectively have been sought. Two main methods are employed to generate CAs, namely, horizontal and vertical generation methods (Nie & Leung, 2011). In the vertical method, an input factor is generated each time, that is, one-factor-one-time (OFOT). This approach is sometimes called one-parameter-at-a-time. At the end of the generation process, whole rows form the final CA. By contrast, the horizontal method is known as one-test-at-a-time (OTAT) (Bryce, Colbourn, & Cohen, 2005; Nie & Leung, 2011).

The OFAT method begins with an initial array that consists of several selected factors (Othman, Zamli, & Mohamad, 2013). To certify combination coverage, the array is extended horizontally by adding one factor at a time. The CA is extended vertically with the introduction of new test cases. This method was first implemented in the in-parameter-order (IPO) algorithm (Yu & Tai, 1998), and this strategy was further developed to generate variations of the IPO algorithm, such as IPOG (Lei, Kacker, Kuhn, Okun, & Lawrence, 2007), IPOG-D (Lei, Kacker, Kuhn, Okun, & Lawrence, 2008), IPOG-F (Forbes, Lawrence, Lei, Kacker, & Kuhn, 2008), and IPO-s (Calvagna & Gargantini, 2009).

The OTAT method normally iterates through all elements of the combinations, and an entire test case is generated per iteration. Most methods begin by generating numerous solutions and then selecting the best solution that covers the majority of the t-tuples; this process requires an optimization mechanism. According to Nie and Leung (2011), these techniques can be classified into four main groups of algorithms: random, greedy, heuristic search, and metaheuristic algorithms. In random optimization algorithms, test cases are selected at random from a complete set of such cases based on input distributions (Nie & Leung, 2011). The selection process is based on the coverage of the t-tuples and simply works by approaching favorable positions in the search space iteratively. These positions are sampled around the current position. Greedy algorithms generally construct a set of objects from the smallest possible elements recursively. Problems are solved through recursion, in which the solution to a particular problem depends on solutions to smaller instances of the same problem. Greedy algorithms are used with the OTAT method to cover many uncovered combinations in each row of the final combinatorial test suite (Wang, Xu, & Nie, 2008). A considerable amount of research has been conducted to develop different algorithms and tools, such as the algorithm applied to pair-wise generation in the CATS tool (Sherwood, 1994), the greedy algorithms used in the PICT tool (Czerwonka, 2006), and the density-based greedy algorithm (Bryce & Colbourn, 2007).

Heuristic search- and artificial intelligence (AI)-based techniques have been employed effectively in CA construction. These techniques generally start with a random set of solutions. Then, a transformation mechanism is applied to this set to transfer it to a new set in which the solutions are particularly efficient for t-tuple coverage. The transformation equations must generate a more efficient set for each iteration. Despite the detailed variations in heuristic search techniques, the essential difference lies in the transformation functions and mechanisms. In the current study, techniques such as SA (Cohen, Dwyer, & Shi, 2007), TS (Nurmela, 2004), GA (Shiba, et al., 2004), ACA (Chen et al., 2009; Shiba et al., 2004), and PSO (Ahmed, Sahib, & Potrus, 2014; Ahmed, Zamli, & Lim, 2012b) are effectively used for CA construction.