Information and Software Technology 66 (2015) 13–29

Contents lists available at ScienceDirect

Information and Software Technology

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

Achievement of minimized combinatorial test suite for configuration-aware software functional testing using the Cuckoo Search algorithm

Bestoun S. Ahmed [a,*], Taib Sh. Abdulsamad [b], Moayad Y. Potrus [a]

[a] Software Engineering Department, Engineering College, Salahaddin University-Hawler (SUH), 44002, Erbil – Kurdistan
[b] Statistic & Computer Department, College Of Commerce, University of Sulaimani, SulaimaniNwe Street 27, Zone 209, Sulaimania, Kurdistan

ARTICLE INFO

Article history:
Received 17 January 2015
Received in revised form 16 May 2015
Accepted 16 May 2015
Available online 21 May 2015

Keywords:
Combinatorial testing
Search-based software testing
Cuckoo Search
Covering array
Test generation tools
Mutation testing

ABSTRACT

Context: Software has become an innovative solution nowadays for many applications and methods in science and engineering. Ensuring the quality and correctness of software is challenging because each program has different configurations and input domains. To ensure the quality of software, all possible configurations and input combinations need to be evaluated against their expected outputs. However, this exhaustive test is impractical because of time and resource constraints due to the large domain of input and configurations. Thus, different sampling techniques have been used to sample these input domains and configurations.
Objective: Combinatorial testing can be used to effectively detect faults in software-under-test. This technique uses combinatorial optimization concepts to systematically minimize the number of test cases by considering the combinations of inputs. This paper proposes a new strategy to generate combinatorial test suite by using Cuckoo Search concepts.
Method: Cuckoo Search is used in the design and implementation of a strategy to construct optimized combinatorial sets. The strategy consists of different algorithms for construction. These algorithms are combined to serve the Cuckoo Search.
Results: The efficiency and performance of the new technique were proven through different experiment sets. The effectiveness of the strategy is assessed by applying the generated test suites on a real-world case study for the purpose of functional testing.
Conclusion: Results show that the generated test suites can detect faults effectively. In addition, the strategy also opens a new direction for the application of Cuckoo Search in the context of software engineering.

1. Introduction

Testing is the process of evaluating the functionality of a system to identify any gaps, errors, missing requirements, and other features. This process ensures the sound operation of software [1]. In general, testing is mainly classified as either functional and structural [2,3]. The former method is referred to as "black box testing," and the latter is called "white box testing" [2–4].

In functional testing, the tester ignores the internal structure of the system-under-test and focuses only on the inputs and expected outputs. The technique serves the overall functionality validation of the system, thereby identifying both valid and invalid inputs from the customer's point of view. Structural testing is used to detect logical errors in software [3]. The tester needs to gather information on the internal structure of the system-under-test and to use information with regard to the data structures and algorithms surrounded by the code [5].

Unlike in structural testing, creating a data set (i.e., test data generation) is an important task in functional testing because of the lack of information about the internal design. Previous studies have reported many test data generation methods. In general, these methods use the available information in software requirement specifications, which provide knowledge about input requirements. The tester considers all possible input domains when selecting test cases for the software-under-test. However, considering all inputs is impossible in many practical applications because of time and resource constraints. Hence, the role of test design techniques is highly important.

* Corresponding author.
E-mail addresses: bestoon82@gmail.com (B.S. Ahmed), Taib.shamsadin@yahoo.com (T.Sh. Abdulsamad), moayad_75@yahoo.com (M.Y. Potrus).

http://dx.doi.org/10.1016/j.infsof.2015.05.005
0950-5849/

A test design technique is used to systematically select test cases through a specific sampling mechanism. This procedure optimizes the number of test cases to obtain an optimum test suite, thereby eliminating the time and cost of the testing phase in software development. Different studies proposed various functional test design techniques, such as equivalence class partitioning, boundary value analysis, and cause and effect analysis via decision tables [3,6]. In general, the tester aims to use more than one testing method because different faults may be detected when different testing methods are used. However, with the vast growth and development of software systems and their configurations, the probability of the occurrence of faults has increased because of the combinations of these configurations, particularly for highly configurable software systems. Traditional test design techniques are useful for fault discovery and prevention. However, such techniques cannot detect faults that are caused by the combinations of input components and configurations [7]. Considering all configuration combinations leads to exhaustive testing, which is impossible because of time and resource constraints [2,8,9].

Strategies have been developed in the last 20 years to solve the above problem. Among these strategies, combinatorial testing strategies are the most effective in designing test cases for this problem. These strategies facilitate search and generate a set of tests, thereby forming a complete test suite that covers the required combinations in accordance with the strength or degree of combination. This degree starts from two (i.e., $d = 2$, where d is the degree of combinations).

Considering all combinations in a minimized test suite is a hard computational optimization problem [2,10–12], because searching for the optimal set is an NP-hard problem [2,11–15]. Hence, searching for an optimum set of test cases can be a difficult task, and finding a unified strategy that generates optimum results is challenging. Three approaches, namely, computational algorithms, mathematical construction, and nature-inspired metaheuristic algorithms, can be used to solve this problem efficiently and find a near-optimal solution [16].

Using nature-inspired metaheuristic algorithms can generate more efficient results than other approaches [17,18]. This approach is more flexible than others because it can construct combinatorial sets with different input factors and levels. Hence, its outcome is more applicable because most real-world systems have different input factors and levels. Techniques that have been used to construct combinatorial sets include simulated annealing (SA) [7], tabu search (TS) [19], genetic algorithm (GA) [20], ant colony algorithm (ACA) [20,21], and particle swarm optimization (PSO) [22,23].

SA generates promising results in cases with small parameters and values as well as a small combination degree. However, it could not exceed certain parameters and values, and is unable to obtain results for combination degrees greater than three [20,24]. PSO can compete with other strategies in most cases even when the combination degree exceeds three [25,26]. However, PSO suffers from the effect of parameter tuning on its performance and from problems with local minima. Recent studies have discovered new nature-inspired metaheuristic algorithms that can produce better results than the traditional PSO algorithm for different applications.

Cuckoo Search (CS) [27] is one of the novel nature-inspired algorithms that have been proposed recently to solve complex optimization problems. CS can be used to efficiently solve global optimization problems [28] as well as NP-hard problems that cannot be solved by exact solution methods [29]. The most powerful feature of CS is its use of Lévy flights to update the search space for generating new candidate solutions. This mechanism allows the candidate solutions to be modified by applying many small changes during the iteration of the algorithm. This in turn makes a compromised relationship between exploration and exploitation which enhance the search capability [30]. To this end, recent studies proved that CS is potentially far more efficient than GA and PSO [31]. Such feature have motivated the use of CS to solve different kinds engineering problems such as scheduling problems [32], distribution networks [33], thermodynamics [34], and steel frame design [35].

The current paper presents the design and implementation of a strategy to construct optimized combinatorial sets using CS. Besides the Lévy flights, another advantage of CS over other counterpart nature-inspired algorithms such as PSO and GA, is that it does not have many parameters for tuning. Evidences showed that the generated results were independent of the value of the tuning parameters [27,31].

The rest of the paper is organized as follows: Section 2 presents the mathematical notations, definitions, and theories behind the combinatorial testing. Section 3 illustrates a practical model of the problem using a real-world case study. Section 4 summarizes recent related works and reviews in the existing literature. Section 5 discusses the methodology of the research and implementation. The section reviews CS in detail and discusses the design and implementation of the strategy. In addition, it shows how the combinations are generated and describes in detail the algorithms that are used within the proposed strategy. Section 6 contains the evaluation results on the efficiency, performance, and effectiveness of CS. Section 7 presents threats to validity for the experiments and the case study. Finally, Section 8 concludes the paper.

2. Covering array mathematical preliminaries and notations

One future move toward combinatorial testing involves the use of a sampling strategy derived from a mathematical object called covering array (CA) [36]. In combinatorial testing, CA can be simply demonstrated by a table with rows and columns that contain the designed test cases; each row is a test case, and each column is an input factor for the software-under-test.

This mathematical object originates essentially from another object called orthogonal array (OA) [12]. An orthogonal array $OA_\lambda(N; d, k, v)$ is an $N \times k$ array in which for every $N \times d$ sub-array, each d-tuple occurs exactly λ times, where $\lambda = N/v^d$. In this equation, d is the combination strength; k is the number of factors ($k \geqslant d$), and v is the number of symbols or levels associated with each factor. To consider all combinations, each d-tuple must occur at least once in the final test suite [37]. When each d-tuple occurs exactly one time, then $\lambda = 1$, and it can be excluded from the mathematical notation, i.e., $OA(N; d, k, v)$. As an example, the orthogonal array OA(9; 2, 4, 3) that contains three levels of value (v), with a combination degree (d) of two, and four factors (k) can be generated by nine rows. Fig. 1(a) illustrates the arrangement of this array.

OA (9; 2, 4, 3)

k_1	k_2	k_3	k_4
1	1	1	1
2	2	2	1
3	3	3	1
1	2	3	2
2	3	1	2
3	1	2	2
1	3	2	3
2	1	3	3
3	2	1	3

(a)

CA (9; 2, 4, 3)

k_1	k_2	k_3	k_4
1	3	3	3
3	2	3	1
1	1	2	1
1	2	1	2
3	1	1	3
2	1	3	2
3	3	2	2
2	3	1	1
2	2	2	3

(b)

MCA (9; 2, 4, $3^2 2^2$)

k_1	k_2	k_3	k_4
2	1	1	2
2	2	2	1
3	3	2	2
1	3	1	1
1	1	2	1
1	2	1	2
3	2	1	1
3	1	1	1
2	3	1	2

(c)

Fig. 1. Examples illustrating OA, CA, and MCA.