SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Analyzing Characteristics of Search Problem and Choosing Appropriate Evolutionary Algorithm

Kabir Umar
Department of Software Engineering,
Faculty of Computing, Bayero University Kano.
Email ukabir.se@buk.edu.ng

Abstract

Choosing appropriate search algorithm for a given task has never been a trivial issue. First, the task needs to be formulated as a search problem, then second, distinguishing characteristics of the search problem need to be analyzed so as to identify which search algorithm is appropriate for the task. This paper presents formulation of SQL injection detection as a grammar reachability search problem, and analyzes the distinguishing characteristics of the search problem, namely: representation of candidate solutions, nature and estimation of search space, variation operations, and fitness evaluation. The paper also present strategy for choosing appropriate search algorithm for the search task. The idea presented in the paper can help in providing further insight into formulation of task as a search problem as well as selecting the right search algorithm to handle the task.

Keywords: *Search Problem, Evolutionary Algorithm, Fitness Evaluation, Variation Operation*

1. Introduction

For several decades, search algorithms have been widely used in addressing NP-hard problems. These are class of problems for which exhaustive enumeration of all possible solutions is not feasible (Hidalgo-Herrero, 2013; Wikipedia, 2024f). To address problem using search algorithm, the first step is to formulate the task as a search problem. Second, is to analyze distinguishing characteristics of the formulated search problem so as to choose an appropriate search algorithm for the task. Choosing appropriate search algorithm for a given task is not a trivial decision because there are several existing search algorithms to choose form. There are random search algorithms, local search algorithms, and global search algorithm. Each existing search algorithm has its strength, weakness, and kinds of problems for which it is applicable depending on some characteristics which include: representation of candidate solution, nature and estimate of corresponding search space, what variation operations are applicable, and how fitness of candidates is measured.

This paper presents formulation of SQL injection vulnerabilities detection as grammar reachability search problem. The paper also presents the distinguishing characteristics of the formulated search problem, and finally present strategy of choosing appropriate search algorithm for the search task. The idea presented in this paper can help researchers to further understand how a task can be formulated as a search problem and how to choose the right algorithm for the task.

The remaining of this paper is organized as follows. Section 2. present review of related literature. The section present existing research works that address SQL injection vulnerabilities detection using search algorithms. This is followed by overview of three main classes of search algorithms, namely random search algorithm, local search algorithm, and global search algorithm. Section 3. present the key formulation of SQLIVs detection as grammar reachability search problem, and then present the key distinguishing characteristics of the search problem. Section

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

4. present strategy for choosing appropriate search algorithm for the task. Discussion is presented in Section 5. And finally, the paper is concluded in section 6.

2. Literature review

This section presents review of related literature. The section begins by presenting research works that were conducted in testing web applications for vulnerabilities using Search Based Software Testing (SBST) approaches. This is followed by discussion on search algorithms and their distinguishing characteristics.

2.1 Search Based Detection of SQL Injection Vulnerabilities

Since the main subject of this paper is on analyzing characteristics of search problem and choosing appropriate search algorithm, therefore, the literature review presented in this section focus more attention on the kind of search algorithms employed by reported research works. Lin & Dong (2024) propose technique for generating test cases based on dynamic feature libraries and attack models by analyzing characteristics of SQL. Genetic algorithm was employed to guide the population's evolution by monitoring state changes of the web application. Furthermore, the selection phase of the genetic algorithm was improved to optimally deform multiple iterations for use cases with low fitness values in order to preserve the population diversity for expanding attack capability of the test cases for more efficient SQL injection detection.

Bahruddin et al. (2023) proposed genetic algorithm based adversary simulation for Structured Query Language (SQL) injection attacks to bypass Web Application Firewalls (WAF). WAF is used to protect web applications from SQL injection, the tool is easy to integrate with the web server, has a minimal cost, and is easy to configure. Baptista et al. (2022a) employed three nature inspired algorithms, namely: Genetic Algorithms (GA), Artificial Bee Colony (ABC), and Ant Colony Optimization (ACO) to develop an approach that detect SQL injection vulnerabilities in source code of web application. The approach was tested empirically using vulnerable web applications-, Bricks, bWAPP, and Twitterlike. Comparison was performed with other tools in the literature. The result obtained verify the effectiveness and robustness of the proposed approach. The same authors also proposed another approach to detect SQL injection vulnerabilities in the source code, using swarm-based algorithms, Artificial Bee Colony (ABC) and Ant Colony Optimization (ACO). The approach was tested empirically using vulnerable web applications - Bricks, bWAPP, and Twitterlike. Experimental results verify the effectiveness of the approach (Baptista et al. 2022b).

The work of Vemulakonda & Venkatesh (2020) proposed approach that combines genetic algorithm and machine learning to develop a novel framework for SQL Injection Attack Detection and Prevention (SQLIADP). The approach uses Java, web technologies, and Special Text Strings to protect Web applications against SQL Injection attacks with zero false reduction and minimal response time. The implemented framework was very efficient and works effectively against SQL Injection. Furthermore, in the work of Arasteh et al., (2024), Gray-Wolf algorithm and machine learning algorithms were combined to develop SQL injection detection method. A training dataset consisting of 13 features was prepared. The most effective features of the dataset were selected using the gray-wolf algorithm, while machine learning algorithms were used to create effective and efficient classifier for detecting SQLi attacks. Experimental results show that the proposed SQL injection detector obtain 99.68% accuracy, 99.40% precision, and 98.72% sensitivity. Moreover, the method increases the efficiency of attack detection by selecting 20% of the most effective features.

Thomé et al. (2014) proposed BIOFUZZ technique which uses Genetic Algorithm (GA) to evolve attack input strings that target input parameters. The targeted attacks are carried out, and a database logger is used to

SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

capture dynamic data accesses. Thereafter, the database logger is searched for input strings that match played attacks. As the attack inputs evolves, the technique is able to detect SQLIVs in real world applications. The work of Aziz et al. (2016) proposed a technique based on Genetic Programming (GP). The technique evolves test datasets, which are then used to test applications for SQLIVs. Alshahwan & Harman (2011) proposed Hill Climbing approach that generates test suite which maximizes branch coverage, and expose SQL injection vulnerabilities.

Obviously, both approaches in (Thomé et al., 2014; Alshahwan & Harman, 2011; Aziz et al., 2016) can be seen as search-based test generation strategies for detection of SQLIVs.

Empirical evidences (Arcuri, 2008, 2011; Ackling et al., 2011) have shown that automated software bugs removal is achievable. For instance, Arcuri, (2008, 2011) proposed technique for automatic fixing of software bugs. His technique receives source code of application and set of unit tests as input. The technique represents the application as genetic program, and employs GP (Genetic Programming) strategies to evolve it guided by the unit tests as training set. Through fitness evaluation, mutation, and reproduction, the GP would eventually evolve version of the application that is able to pass all the unit tests. Thus, the bugs in the application are automatically fixed. Ackling et al. (2011) proposed Genetic Algorithm (GA) framework that evolve patches for software repair. The GA evolve patches that, when applied to an application will minimize number of test failures. By evolving a patch that is able to reduce number of test failures to zero, the technique could be able to produce a patch for automatic repair of an application.

2.2 Overview of Search Algorithms

This section presents different classes of search algorithms, and outlines their distinguishing characteristics, including representation of candidate solution, search space, fitness evaluation, and variation operations. Search algorithm is heuristic based method that finds reasonably good solution to a problem whose solutions space is extremely large and cannot easily be exhaustively enumerated. The search process usually starts from a single candidate solution, or population of solutions, and then iteratively moves to better solution(s) by variation of current solution(s). The movement to better solution(s) is performed based on fitness of current solution(s). This process is repeated iteratively until certain criterion is reached. The criterion could be attainment of adequate level of fitness or conducting some number of search iterations (Wikipedia, 2024a, 2024c, 2016h, 2024e). Three classes of search algorithms are presented in this paper, namely: random search algorithms, local search algorithms, and evolutionary (global) search algorithms.

Random Search Algorithm: A typical Random Search (RS) algorithm works by randomly chosen any candidate solution in the search space, and iteratively moving to better solutions through random variations and fitness evaluations, until adequate level of fitness is attained or certain number of iterations are performed (Wikipedia, 2024c; Srinivas & Patnaik, 1994). A simplified RS algorithm that searches for minimal optimal solution is described as shown in Fig. 1.

```
Algorithm random_search( )

begin
 1. Initialize x with random candidate solution
 2, Repeat until termination criterion
 3. {
```

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

```
4.      Randomly choose new candidate solution y (by applying random
        variation to x);
5.      If fitness of y is less than fitness of x, then x = y
6.  }  // of repeat
end
```

Figure 1. Simplified Random Search Algorithm

Local Search Algorithms: A local search algorithm is heuristic based method that usually starts from a candidate solution and then iteratively moves to a better neighbor solution based on fitness information about the solutions in the neighborhood of the current candidate solution. A number of different local search algorithms exist in the literature. They include Hill Climbing (HC), Simulated Annealing (SA), and Tabu Search (TS) (Wikipedia, 2024d; Lenstra, 2003; Mitchell et al., 1993; Ahsan & Anwer, 2024; Alshahwan & Harman, 2011).

Hill Climbing (HC) is one of the most successful and widely used local search algorithms (Mitchell et al., 1993). In this paper we consider HC as typical representation of local search algorithm. In HC, the algorithm moves to the best fitness neighbor solution during each iteration. A simplified HC algorithm that searches for a minimum optimal solution is presented in Fig. 2.

```
Algorithm hc_search( )

begin
1.   // initialize first candidate solution
2.   initialize x_0 with random candidate solution
3.   // perform search iteratively
4.   Repeat until termination criterion
5.   {
6.       choose k neighbors of x_0 as {x_1, x_2, x_3, ..., x_k}
7.       // (by applying local variation to x_0);
8.       for (i = 1 to k)
9.       {
10.          If (fitness of x_0 > fitness of x_i ){
11.              x_0 = x_i
12.          }
13.      }  // of for
14.  }  // of repeat
end
```

Figure 2. Simplified Hill Climbing (HC) Algorithm

Random search and Local search algorithms such as HC support phenotype representation of candidate solutions, however, both random search and local search algorithms are one solution algorithms, therefore, they cannot search solution space for finding multiple optimal solutions. Moreover, in a search space containing multiple optimal solutions, each occurrence of an optimal solution translates to a Minima (or Maxima) point within the

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

search space, resulting in phenomenon called "ridges". Unfortunately, random search and local search algorithms, cannot handle search space containing ridges. In addition, both RS and LS can only support variation through application of mutation but never cross over. This is because cross over requires two parent solutions, unfortunately, RS and LS algorithms evolves only one candidate solution at all times (Wikipedia, 2024d; Wikipedia, 2024b; Lenstra, 2003; Mitchell et al., 1993).

Evolutionary (Global) Search Algorithm: Global search algorithms are heuristic based method that have the capacity of searching for solutions within an entire search space. This is unlike random search and local search algorithms that, respectively, search for solutions at random, and within the neighborhood of current candidate solution. These algorithms are mainly population based. They explore entire search space by generating population of candidate solutions, and exploit neighborhood of individual candidate solutions through variation operations. Several classes of global search algorithms exist, such as evolutionary algorithms, Swarm algorithms, colony algorithms, etc. (Wikipedia, 2024a; Harman et al., 2009; MacNish, 2000; Mitchell et al., 1993; Eiben & Schoenauer, 2002; Ahsan & Anwer, 2024). However, this paper focuses on evolutionary algorithms.

Evolutionary algorithms are class of global search algorithms that stem from the concept of biological theory of natural evolution of animal species. They are mainly generational and consist of four main different algorithms namely, Genetic Algorithm (GA), Genetic Programming (GP), Evolutionary Programming (EP), and Evolutionary Strategies (ES). Although these four algorithms share some common characteristics, however, each of them possesses some key features that distinguishes it from the rest, and makes it suitable for some classes of problems (Harman et al., 2009; MacNish, 2000; Mitchell et al., 1993; Eiben & Schoenauer, 2002). Table 1. shows the distinguishing features of the evolutionary algorithms.

Table 1. Distinguishing features of Evolutionary Algorithms

Feature	**GA**	**GP**	**ES**	**EP**
Phenotype representation	No	Yes	Yes	Yes
Fitness evaluation	Yes	Yes	Yes	Yes
Mutation	Yes	Yes	Yes	Yes
Cross over	Yes	Yes	Yes	No
Single parent reproduction	No	No	No	Yes
Multiple optimal solutions	Yes	Yes	Yes	Yes

Notes, Yes means feature is supported. No means feature is not supported.

It can be seen from row one of the table that the evolutionary algorithms GP, ES, and EP supports phenotype representation of candidate solution whereas only GA does not support this feature. GA supports genotype representation of candidate solution. Rows two and three of the table show that all the evolutionary algorithms support fitness evaluations, and variation operations (sometimes call mutation operations) respectively. Row four reveals that only EP does not support cross over operation. This is because, as indicated by row five, only EP uses single parent reproduction scheme that provides only mutation and no cross over. The last row reveals the fact that all the evolutionary algorithms are capable of searching a space containing multiple optimal solutions. A simplified evolutionary search algorithm is presented in Fig. 3.

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

```
Algorithm Evolutionary_search( )

begin
   1.  Generate initial population of individuals at random;
   2.  Compute fitness of each individual;
   3.  While (not termination condition) do
   4.  {
   5.      select parents from individuals;
   6.      produce off-springs;
   7.      form population of next generation;
   8.      compute fitness for each individual;
   9.  } // of while loop
end
```

Figure 3. Simplified Evolutionary Algorithm.

3. Characteristics of Grammar Reachability Search Problem.

This section presents our strategies of formulating SQL injection vulnerabilities detection as grammar reachability search problem. In addition, characteristics of the grammar reachability search problem are analyzed in order to identify appropriate search algorithm for the task.

3.1 Formulation of SQL Injection Vulnerabilities Detection as Grammar Reachability Search Problem.

This section presents overview of the reformulation of SQLIVs detection as grammar reachability search problem. The section begins with Fig. 4. that shows source code of an example webpage which performs basic user authentication in a Java web application.

```
1    protected void doPost(HttpServletRequest request,
2    HttpServletResponse response) {
3
4    String N ="";
5
6    String Q = null;
7    String R = null;
8    java.sql.Statement stat = null;
9    stmt = conn.createStatement();
10   java.sql.ResultSet S = null;
11   N = request.getParameter("username");
12   String P = request.getParameter("userpass");
13   R = toSQL(N); // validation for N (i.e. Username)
14   Q = "select * from userstbl where uname='" + R + "'
15   AND passwd ='" + P + "'";
16
17   S = Stmt.executeQuery(Q);
18    // exec qry at sensitive sink
19   }
```

Figure 4. Source code of Example Vulnerable Webpage

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

The source code in Fig. 4. is used as a running example for illustration purpose in the remaining sections of this paper. The following terminologies should be noted:

Application's Entry Point (AEP): Is a program statement at which input data gets into web application. In the source code, this is identified by presence of AEP Function such as request.getParameter("username") of line 11.

Sensitive Sink (SS): Is a program statement at which dynamic query is executed inside web application (Medeiros et al., 2014; Halfond & Orso, 2005; Yan et al., 2013). In a source code, an SS is identified by presence of query executing commands such as Stmt.executeQuery(Q) of line 17.

Data Validation Statement: This refers to any statement that performs validation of input data. For example, the statement R = toSQL(N) of line 13 is a data validation statement which uses the data validation function toSQL(N) to secure the username input data.

From the source code, it can be seen that data input field "username" is validated at line 13 using the validation function toSQL(N), whereas data input field "Password" is not validated. Thus "Password" field is vulnerable to SQL injection. The key idea behind reformulation of SQL injection vulnerabilities detection as a grammar reachability search problem in a given source code of web page such as Fig. 4. Is as follows:

Given the source code of a web page (such as Fig. 4.) which receives input data via AEP functions, uses these input data in generation of query strings, and execute the query dynamically using SS functions, then it follows that:

Where data flow paths can be established which begins from a given AEP (such as "password" in line 12 of Fig. 4.) and ends in an SS (such as line 17 of Fig. 4.), if data validation is performed along the path, then the associated AEP (such as "username" in lines 11 of Fig. 4.) is said to be validated and secure, otherwise, if data validation is NOT performed along the path, then the corresponding AEP (such as "password" in line 12 of Fig. 4.) is vulnerable to SQL injection.

The AEP-to-SS path can be represented as a sequence of source code line numbers that shows corresponding data flow across program statements for example L11, L13, L14, L17 depicts data flow from AEP parameter "username" to dynamic query execution at SS in Line 17. The strategy of our grammar-based vulnerabilities detection is to extract grammar production rules from declaration and assignment statements, such that the nonterminal symbol of the LHS (Left Hand Side) of each grammar production rule represents the variable of the LHS of the corresponding statement from which the grammar rule is extracted.

The extracted grammar rules are converted to context free grammar (CFG) production rules. Then, the CFG rules are used to test reachability from nonterminal symbol representing AEP statement to nonterminal symbol representing SS statement. However, in the extracted grammar, the rule extracted from a SS statement is considered as the start rule for tracking reachability. The Format of Extracted Grammar Production Rules is shown in table 2. Furthermore, corresponding CFG production rules extracted from source code of Fig.4. are shown in Fig.5.

SLU JST
Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Table 2. Format of Extracted Grammar Production Rules

Example	Description
$X_p \rightarrow X_q$	Unit rule, nonterminal produce single nonterminal
$X_p \rightarrow \alpha X_q \beta \quad where \ \alpha, \beta \in (\Sigma \cup N)^*$	Nonterminal produce combinations of terminals and nonterminals
$Y_{pa} \rightarrow \gamma \ \ where \ \gamma \in \Sigma^*$	Nonterminal produce sequence of terminal symbols
$X_{pb} \rightarrow \alpha$ **fun_name**$(X_q)\beta$	Nonterminal symbol produce function with one argument, e.g. **executeQuery**(X_p), **toSQL**(X_p)

Notes that, in the table, X_{ab} and Y_{ac} denotes nonterminal symbols, the subscript $a = \mathrm{p, q, r}$ are line numbers of statements from which grammar rule is extracted, subscript $b =$ letter S or A or V and subscript $c = letter\ a$. Moreover, **fun_name** represents function names, denoting sensitive sink functions, AEP functions, and data validation functions.

$X_{17S} \rightarrow$ **executeQuery**(X_{14})
$X_{14} \rightarrow Y_{14a}\, X_{13}\, Y_{14b}\, X_{12A}\, Y_{14c}$
$Y_{14a} \rightarrow$ select * from userstbl where uname=' +
$Y_{14b} \rightarrow$ + "' AND passwd = '" +
$Y_{14c} \rightarrow$ + "'"
$X_{13V} \rightarrow$ **toSQL** (X_{11A})
$X_{11A} \rightarrow$ **request.getParameter**("username")
$X_{12A} \rightarrow$ **request.getParameter**("userpass")

Figure 5. CFG rules extracted from the running example.

The extracted CFG rules, such as Fig. 5, are then analyzed for establishing possible flow paths from AEPs to SSs by testing for reachability of nonterminal symbols. Obviously, reachability in this context means existence of an AEP-to-SS data flow path. Where reachability is found, we analyze the corresponding productions sequence (i.e., derivations sequence) for presence of data validation along productions sequence. Where data validation is found along productions sequence, then the associated AEP parameter is secured and not vulnerable. However, absence of data validation along such productions sequence means that the associated AEP parameter is not secured, and thus, SQLIV is found.

By reformulating detection of SQL injection vulnerabilities as grammar reachability search problem, individual candidate solutions are formed as sequence of CFG production rules. Therefore, the key challenge is "finding" the reachability "productions sequence" from pool of CFG production rules extracted from source code of web page been analyzed. Finding "productions sequence" from pool of grammar rules can be done by repeated sequential search through all possible combinations of grammar rules in the pool. Although, the procedure of repeated sequential search sounds pretty satisfactory and can deliver. Unfortunately, given a web application with several large web pages containing multiple SSs, this approach could face repeated sequential search through very large number of production rules, thereby resulting in very heavy amount of computation. For instance, consider

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

just a single moderate size web page with 500 lines of code, for which 225 CFG production rules were extracted. Selecting set of "8-rules productions sequence" that establishes required reachability for a given SS can be done in about $5.791672 * 10^{18}$ ways (i.e., 225 Permutation 8 ways). This is an extremely large number of searches.

In view of the above example, it can easily be argued that as the size of web page grows in lines of codes and multiple SSs, the size of search space increases geometrically such that exhaustive evaluation of all possible combinations of grammar rules to establish SS-to-AEP reachability become infeasible. Consequently, the task of "selecting" or "finding" appropriate "productions sequences" that establishes SS-to-AEP reachability can be modelled as a Search Problem (SP), and be solved using heuristics-based search algorithm. Unfortunately, the literature offers several search algorithms for different kinds of SPs. In general, search algorithms are classified into "global" search algorithms such as Evolutionary Programming (EP), "local" search algorithms such as Hill Climbing (HC), and "random" search algorithms. Each of these algorithms have specific strengths, weaknesses, and problem scenarios for which it is suitable and performs very well. Thus, a challenging question is which search algorithm is suitable for the grammar reachability search problem? In order to properly address this important question, there is need to deeply investigate some key characteristics of the problem at hand. These are characteristics which directly affect different aspects of the search process, and they include, how to specify representation for candidate solutions, nature and estimate of the search space, how to evaluate fitness of candidate solutions, how to perform variation over candidate solutions, and whether cross-over is feasible and constructive, among other things. These characteristics are discussed in the following subsections.

3.2 Representation of Candidate Solution

An important requirement of applying search algorithm to solve a given problem is specifying representation of candidate solutions (Eiben & Schoenauer, 2002). Our search process seeks to find CFG "productions sequences" that establishes SS-to-AEP reachability. We represent candidate solution as a comma-separated "chain of grammar production rules" corresponding to "productions sequence". In the chain of grammar production rules that defines candidate solution, a grammar rule is represented by its LHS nonterminal symbol. Therefore, the representation for candidate solution can be expressed as shown in Fig. 6 below.

$X_i , X_j , X_k , \ldots , X_q$

Figure 6. Representation of Candidate Solution

From Fig. 6., it is apparent that the number of nonterminal in the chain varies from one candidate to another, depending on how many productions are so far applied in the sequence. The chain of grammar rules in Fig. 6. represents a candidate solution, and corresponds to a productions sequence. Figure 7. shows three example candidate solutions for the running example.

i. X_{14} , X_{12A}
ii. X_{14}
iii. $X_{17S} , X_{14} , X_{13V}, X_{11A}$

Figure 7. Example of Candidates for the running example web application

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

The representation is chosen for simplicity of analysis during the search process. The representation would make it easier to check for presence of data validation along productions sequence which establish reachability. The above candidate solution is of phenotype representation, since it depicts the actual "productions sequence" which makes the desired solution. The candidate solution is not abstract representation of the desired solutions, as in genotype. Hence, variation operations and fitness evaluations will apply directly to the candidate solutions during the search process. In addition, the above candidate's representation suggests that the proposed search process requires a search algorithm that supports phenotype representation of candidate solutions. A clear question that comes to mind is "What is the estimate total number of different possible candidates that can be defined?" This question is answered in the next subsection.

3.3 Estimation of Search Space

A number of techniques for search space estimation exist (Arcuri, 2011). These techniques are reviewed so as to define appropriate search space estimation for the proposed grammar reachability search process. As mentioned earlier in preceding subsection, the proposed search process analyzes single webpage at a time, therefore, the applicable search space consists of all possible "productions sequences" that can be formed from the CFG production rules extracted from the webpage. If **n** grammar production rules are extracted from a webpage, then for each SS in the webpage, the key goal of our approach is to search pool of **n** production rules and find sets of $\boldsymbol{r}_i$ appropriate rules, where each set of rules form a "productions sequence" that establishes SS-to-AEPs reachability.

Therefore, the search space consists of all different possible selections of $\boldsymbol{r}_i$ rules from total of **n** distinct rules. This is because each possible selections of $\boldsymbol{r}_i$ rules correspond to a given "productions sequence". In the task at hand, the **n** rules in the pool are distinct, ordered and the position of each rule is important. Hence, the first of the $\boldsymbol{r}_i$ rules can be chosen in **n** different ways, the second can be chosen in (**n – 1**) different ways, and so on, while the last of the $\boldsymbol{r}_i$ rules can be chosen in ($\boldsymbol{n - r_i + 1}$) different ways. Thus, the search space of candidate solutions corresponds to "**n** Permutations of $\boldsymbol{r}_i$". Therefore, the size of the search space can be estimated using the following definition.

Definition 1: Search Space of Candidates for a given Sensitive Sink

Let $S(X_{iS})$ denotes size of search space for X_{iS} (Sensitive Sink at ith statement) in a given webpage (WP) having k number of sensitive sinks. If reachability from X_{iS} to arbitrary AEPs is established using r_i rules from pool of **n** distinct rules, then $S(X_{iS})$ is defined as the total number of different selections of r_i rules from pool of **n** rules, estimated as **n** permutation of r_i. This is mathematically expressed as follows.

$$S(X_{iS}) = n(n-1)(n-2) \ldots (n - r_i + 1) = n!/(n - r_i)! \qquad (1)$$

Definition 2: Search Space of Candidates for a Webpage

The size of search space for webpage WP containing k multiple SSs, denoted as S(WP), is estimated as the summation of search spaces of individual SSs in the page. This is expressed as follows.

$$S(WP) = \sum_{i \in k} S(X_{iS}) = \sum_{i \in k} n!/(n - r_i)! \qquad (2)$$

where $k = \{\text{SS statements numbers}\}$

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Obviously, Eq. 2. gives a mere estimation of the maximum search space possible for WP. This is because in some web applications, the scope in which variables are defined and used can significantly restrict the size of applicable search space, thus, suggesting that a search within neighborhood of SS can be sufficient enough to find all associated productions sequences. Unfortunately, this is not always the case. In building large real world web applications, for example, it is common practice to declare some variables at global scope, and access them from several locations in different statements, including AEP and dynamic query statements. Hence, we can be comfortable using Eq. 2, and ignore any restrictions that might be induced by variables scope constraints.

Moreover, the equation shows that as the sizes of **n** and/or r_i increases, the size of the search space increases geometrically. The implication is that, for real world large web application with large pages and multiple SSs, a slight increase in any of the two parameters can significantly increase the size of the search space. Invariably, this search space can be extremely large, even for a moderate web page consisting of few hundred LOCs and few SSs.

For the grammar reachability search problem at hand, each occurrence of SQLIV in a webpage corresponds to a given optimal solution and would translate to a **Minima** point in the search space described above. This is because in a large web application, it is often possible to have multiple SQLIVs in a single webpage. Hence, with several SQLIVs in a webpage, the search space would contain several Minima points resulting in phenomenal problem called "ridges" (Wikipedia, 2024b, 2016h). This is an important characteristic of the above search space for the proposed search problem. Consequently, this suggest that the search task at hand requires a global search algorithm which can handle search space with ridges.

As an illustrative example, let us estimate the size of search space for the running example. Recall that the total number of CFG production rules **n** extracted from source code of the running example is eight (see Fig. 5. CFG extracted from the running example). Using Eq. 1, the size of the search space can be estimated as follows.

$$S(X_{17S}) = 8!/(8-4)!$$

$$= 8 * 7 * 6 * 5 * 4!/4!$$

$$= 8 * 7 * 6 * 5 * 1$$

$$= 1680 \text{ candidates}$$

Hence, the estimated search space for the sensitive sink X_{17S} is 1680 candidates. Moreover, in this example, the estimated search space for the sensitive sink X_{17S} is the same as the search space for the entire webpage WP, because there is only one sensitive sink (X_{17S}) in the Running example. Therefore,

$$S(\text{WP}) = S(X_{17S}) = 1680 \text{ candidates}$$

The above search space estimation means that about 1680 different candidate solutions can be defined using the eight CFG production rules extracted from source code of the running example. In order words, an estimate of 1680 different "productions sequences" are possible. Obviously, not all of these candidate solutions will establish an SS-to-AEP reachability. Therefore, in the proposed search process, it is necessary to have means of assessing the "reachability measure" of each candidate solution. Fitness evaluation strategy that measures degree of reachability attained by each candidate solution in the proposed search process is highlighted in the following subsection.

SLUJST
Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

3.4 Fitness Evaluation

The proposed search process begins by seeding population of first generation with candidate solutions (i.e., productions sequences), each consisting of a single randomly selected production rule. The candidates grow along the search process through application of variation operations and fitness evaluation. In the literature, different fitness evaluation approaches have been defined, depending on the problem at hand (Arcuri, 2011; Thomé et al., 2014; Dominguez-Jimenez et al., 2011; Aziz et al., 2016; Baresel et al., 2002; Arcuri & Yao, 2014; Ahsan & Anwer (2024).

These existing approaches were reviewed so as to define appropriate fitness evaluation for the proposed search process. In the proposed search process, the fitness of candidate solution is evaluated in three dimensions that measures how much reachability is established so far. These three dimensions are: Rules production correctness, Productions sequence completeness, Data validation, and they are defined using equ. 3., equ. 4., equ. 5., below respectively. Details of formulation of these fitness equations is presented in our earlier work (Umar, K. et al., 2018b) The production correctness fitness of a candidate is evaluated as follows:

$$f_{prod}(C) = \sum_{i=1}^{k} f_{prod}(X_i) \quad i = 1,2,..,k \qquad (3)$$

The productions sequence completeness fitness of candidate is evaluated as follows:

$$f_{com}(C) = f_{com}(X_1) + f_{com}(X_k) \qquad (4)$$

The data validation fitness of candidate is evaluated as follows:

$$f_{val}(C) = \begin{cases} 0 & X_i \overset{+}{\Rightarrow} \textbf{val\_fun}(X_i)\ \beta \\ 5 & \text{otherwise} \end{cases} \qquad (5)$$

where X_{ihe}, X_i heP_{seq}, $j > 0, i > j,$ *and* $i = 2, \ldots =$

Finally, the overall fitness of candidate is evaluated as summation of the three fitness components as follows.

$$f(C) = f_{val}(C) + f_{prod}(C) + f_{com}(C) \qquad (6)$$

The overall fitness gives reachability measure for the candidate solution, and is employed by the proposed search process to find optimal solutions that establish an SS-to-AEP reachability with no data validation, and consequently reveals presence of SQLIVs. Therefore, the proposed grammar reachability search process requires search algorithm which provides for fitness evaluation of candidates as described above. In the course of the search process, the candidates grow and improve in fitness through application of variation operations that produce offspring. The strategy for variation of candidates is highlighted in the following subsection.

3.5 Variation Operations

The proposed search process begins by seeding population of first generation with candidates, each consisting of a single randomly selected production rule. The candidates grow along the search process through applications of four variation operations and fitness evaluation. In our research work, we defined novel variation operator for the proposed search process. The novel variation operator aims at improving elitism for the search process while

SLU JST
Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

maintaining balance between exploration and exploitation of the search space (Wikipedia, 2024a; Ahsan & Anwer, (2024). Thus, the novel variation operator is defined as three tuples using Equ. 7.

$$S_{opr} = \{Opr, C, tournamentOf(q)\} \quad (7)$$

Where Opr is the randomly chosen variation operation, C is the candidate solution, and $tournamentOf(q)$ is a function that applies tournament selection on q rules and returns the best neighbor rule accordingly. The novel variation operator applies four variation operations to bias the search process to neighborhood of candidate solutions as well as to eligible variations. The way in which the novel variation operator uses the fitness information to bias application of eligible variation operations is summarized in Table 3.

Table 3. Biasing of Variation Operations

Variation Operation	Eligibility Condition	Interpretation
Append Head	$f_{com}(X_1) = 5$ and $f_{prod}(X_2) = 0$ if $k > 1$ $f_{com}(X_1) = 5$, if $k = 0$	Append head if first rule is not X_{iS}, but it is connected to second rule
Replace Head	$f_{prod}(X_2) = 1$	Replace head if first rule is not connected to second rule
Append Tail	$f_{com}(X_k) = 5$ and $f_{prod}(X_k) = 0$, if $k > 1$ $f_{com}(X_k) = 5$, if $k = 0$	Append tail if last rule does not contain "AEP fun", but is appropriate
Replace Tail	$f_{prod}(X_k) = 1$	Append tail if last rule is not appropriate

In Table 3, the first column shows four variation operations that are applied to candidate. The second column shows the values of candidate's fitness for which the corresponding variation operation is eligible. The last column provides explanation of what is done based on fitness values in column two. Consequently, the novel variation operator applies the above four variation operations to candidate solution through mutation process. Therefore, the proposed grammar reachability search process requires search algorithm that support mutation operation. Furthermore, it should be observed that cross-over operation is not required in the variation operations defined above.

4. Selection of appropriate Search Algorithm

Having discussed the characteristics of the grammar reachability search problem in the preceding subsections, several search algorithms are analyzed to choose the most appropriate algorithm for handling the task. The characteristics of the proposed search problem that were analyzed are: representation of candidates, search space, fitness evaluation, and variation operations. First, the feasibility of applying Random Search algorithm to handle the proposed search task is investigated. Second, the feasibility of applying local search algorithm, particularly, Hill Climbing (HC) to handle the proposed search task is also investigated. Last, global search algorithms, particularly, Evolutionary algorithms are also investigated for possible consideration. Results of investigations is presented in

Sule Lamido University Journal of Science & Technology Vol. 8 No.1 [June, 2024], pp. 108-124
https://doi.org/10.56471/slujst.v8i.482
Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Table 4. In Table 4, column one shows the features (i.e. characteristics of the search problem) of interest that are used as basis of comparison. The search algorithms analyzed are represented by column two through column seven in the order RS (Random Search), HC (Hill Climbing), GA (Genetic Algorithm), GP (Genetic Programming), ES (Evolutionary strategies), and EP (Evolutionary Programming) respectively. An algorithm is given a score of 1 if the algorithm supports the corresponding feature, and score of 0 if the algorithm does not support the corresponding feature. The last column of the table, which represents the proposed grammar reachability search problem, is given a score of 1 if the corresponding feature is an essential requirement of the proposed search task, and a score of 0 if the corresponding feature is not required by the proposed search task. The last row of the table shows total score for each algorithm.

Table 4. Results of Search Algorithms Analyzed

Feature	RS	HC	GA	GP	ES	EP	Proposed grammar reachability search problem
Phenotype representation	1	1	0	1	1	1	1
Fitness evaluation	1	1	1	1	1	1	1
Mutation operation	1	1	1	1	1	1	1
No cross over	1	1	0	0	0	1	1
Single parent reproduction	1	1	0	0	0	1	1
Multiple optimal solutions	0	0	1	1	1	1	1
Total Score	5	5	3	4	4	6	6

Using the results of Table 4, a bar chart which depicts total scores obtained by the algorithms in comparison with the proposed search task is plotted as shown in fig. 8.

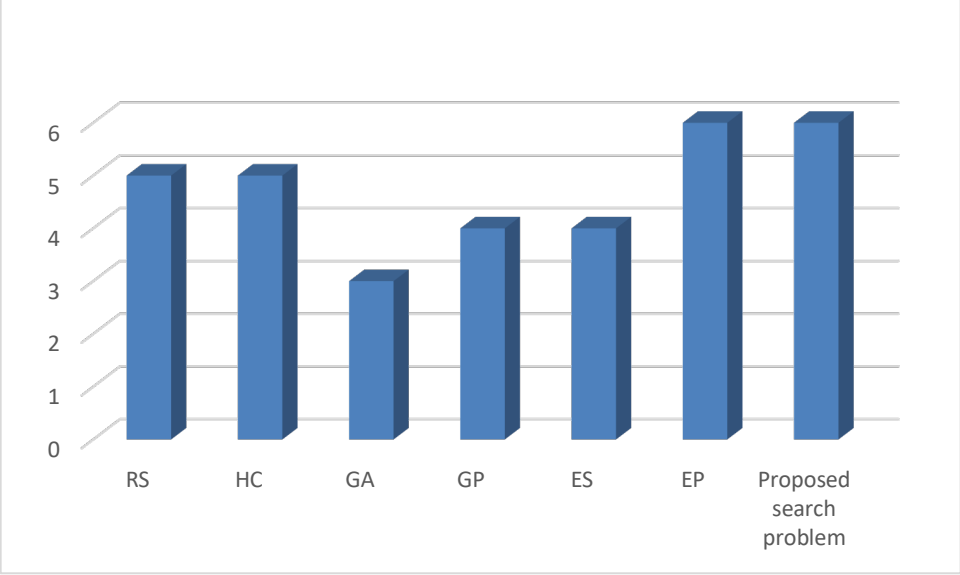

Figure 8. Results of search algorithm analyzed in comparison to proposed search task

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

5. Discussion

From the results shown in Table 4. of the preceding section, it can be seen that the proposed search task has total score of 6. This means that it requires an algorithm which supports all the six features listed in the first column of the table. It should be noted that for algorithm to meet this requirement, the algorithm must have obtained score of 6 as well. RS and HC scored 5, therefore, they can satisfy most of the feature requirements of the proposed search task such as fitness evaluation, mutation and single parent reproduction, however, they are both unable to satisfy the requirement to support multiple optimal solutions. Unfortunately, this is a very important requirement because the applicable search space (discussed in preceding subsection) is characterized by occasional presence of multiple optimal solutions (i.e. ridges) corresponding to multiple SQLIVs in a web page. Consequently, it is not feasible to employ either an RS or HC algorithm to handle the proposed search task.

It can also be seen that all the evolutionary algorithms can satisfy most of the feature's requirement of the proposed search task, such as fitness evaluation, mutation, and support for multiple optimal solutions. Nevertheless, some of the evolutionary algorithms could not meet some requirements of the proposed search task. For instance, GA requires genotype candidate representation, and cross over operation for variation to produce offspring. However, the proposed search task requires phenotype representation of candidate solutions, and does not provide for cross over operation. In addition, GA does not support single parent reproduction as demanded by the proposed search task. Consequently, GA cannot be employed to handle the proposed search task.

Although, both GP and ES support phenotype representation of candidate solutions as required by the proposed search task, unfortunately, they both require cross over operation and do not support single parent reproduction. Hence, it is not feasible to employ either GP or ES to handle the proposed search task. Fortunately, with a score of 6, EP satisfies all feature requirements of the proposed search task, namely, phenotype representation of candidate solutions, fitness evaluation, use of variation/mutation and no cross over, single parent reproduction, and support for multiple optimal solutions. This is depicted in the bar chart of Fig. 8. Where EP has the same score as the proposed search problem. Therefore, EP algorithm is selected to handle the proposed grammar reachability search problem. This is implemented in our work that reported a tool for detection and removal of SQL injection vulnerabilities, named EPSQLiFix (Umar, K. et al., 2018a).

6. Conclusion

Application of search algorithm for addressing wide range of NP-hard problems is still very active in the research community. Since, there are several existing search algorithms, it is therefore very essential to understand the mechanism of choosing an appropriate search algorithm for a given task at hand. The idea presented in this paper suggest how a given task is formulated as a search task problem as well as how an appropriate search algorithm is selected for the task. It is helpful that, the research community would find the ideas very useful.

References

Ackling, T., Alexander, B. & Grunert, I. (2011, July 12–16). Evolving patches for software repair. In *Proceedings of the 13th Annual Conference on GECCO.* Dublin, Ireland: ACM.

Ahsan, F., & Anwer, F. (2024). A systematic literature review on software security testing using metaheuristics. *Automated Software Engineering*, *31*(2), 1-73.

Alshahwan, N. & Harman, M. (2011). Automated web application testing using search based software engineering. In Proceedings of the 26th International Conference on Automated Software Engineering. (pp. 3-12). Washington DC: IEEE/ACM. DOI: 10.1109/ASE.2011.6100082.

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Arasteh, B., Aghaei, B., Farzad, B., Arasteh, K., Kiani, F., & Torkamanian-Afshar, M. (2024). Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms. *Neural Computing and Applications*, 1-22.

Arcuri, A. (2008, May 10 - 18). On the automation of fixing software bugs. *In Proceedings of the 30th International Conference on Software Engineering*. (pp. 1003-1006). Leipzig, Germany: ACM. DOI: 10.1145/1370175.1370223

Arcuri, A. (2011, June). Evolutionary repair of faulty software. *Journal of Applied Soft Computing*, *11* (4), 3494–3514. DOI: 10.1016/j.asoc.2011.01.023.

Aziz, B., Bader, v. & Hippolyte, C. (2016, March 30 - April 1). Search-based sql injection attacks testing using genetic programming. In *Proceedings of the 19th European Conference on EuroGP.* (pp 183-198). Porto, Portugal: Springer. doi: 10.1007/978-3-319-30668-1_1.

Bahruddin, H., Suryani, V., & Wardana, A. A. (2023, September). Adversary Simulation of Structured Query Language (SQL) Injection Attack Using Genetic Algorithm for Web Application Firewalls (WAF) Bypass. In *Proceedings of SAI Intelligent Systems Conference* (pp. 656-669). Cham: Springer Nature Switzerland.

Baptista, K., Bernardino, A. M., & Bernardino, E. M. (2022a, June). Detecting SQL Injection Vulnerabilities Using Nature-inspired Algorithms. In *International Conference on Computational Science* (pp. 451-457). Cham: Springer International Publishing.

Baptista, K., Bernardino, E. M., & Bernardino, A. M. (2022b, April). Detecting SQL injection vulnerabilities using artificial bee colony and ant colony optimization. In *World Conference on Information Systems and Technologies* (pp. 273-283). Cham: Springer International Publishing.

Eiben, A. E., & Schoenauer, M. (2002). *Evolutionary computing*. Information Processing Letters, 82(1), 1-6.

Harman, M., Mansouri, S. A., & Zhang, Y. (2009). Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, King's College London, Tech. Rep. TR-09-03*.

Hidalgo-Herrero, M., Rabanal, P., Rodriguez, I., & Rubio, F. (2013). Comparing problem solving strategies for NP-hard optimization problems. *Fundamenta Informaticae*, *124*(1-2), 1-25.

Lenstra, J. K. (2003). *Local search in combinatorial optimization*. Princeton University Press.

Lin, J., & Dong, W. (2024, January). SPAFuzz: Web Security Fuzz Testing Tool Based on Fuzz Testing and Genetic Algorithm. In *2024 4th International Conference on Consumer Electronics and Computer Engineering (ICCECE)* (pp. 190-195). IEEE.

MacNish, C. (2000, December). Evolutionary programming techniques for testing students' code. In *Proceedings of the Australasian conference on Computing education* (pp. 170-173). ACM.

Mitchell, M., Holland, J. H., & Forrest, S. (1993). When will a genetic algorithm outperform hill climbing?. *Ann Arbor*, *1001*, 48109

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*,*27*(6), 17-26.

Thomé, J., Gorla, A. & Zeller, A. (2014). Search-based security testing of web applications. In *Proceedings of the 7th International Workshop on SBST.* (pp. 5-14). doi: 10.1145/2593833.2593835.

Umar, K., Sultan, A. B., Zulzalil, H., Admodisastro, N., & Abdullah, M. T. (2018a). Comparing Web Vulnerability Scanners with a New Method for SQL Injection Vulnerabilities Detection and Removal EPSQLiFix. *International Journal of Engineering & Technology*, *7*(4.31), 40-45.

Umar, K., Sultan, A. B., Zulzalil, H., Admodisastro, N., & Abdullah, M. T. (2018b, July). Formulation of SQL injection vulnerability detection as grammar reachability problem. In *2018 International*

Online ISSN: 2736-0903
Print ISSN: 2736-089X
www.slujst.edu.ng

Conference on Information and Communication Technology for the Muslim World (ICT4M) (pp. 179-184). IEEE.

Vemulakonda, R., & Venkatesh, K. (2020). SQLIADP: A Novel Framework to Detect and Prevent SQL Injection Attacks. In *Smart Intelligent Computing and Applications: Proceedings of the Third International Conference on Smart Computing and Informatics, Volume 2* (pp. 41-50). Springer Singapore.

Wikipedia. (2024a, January 10th). Global Optimization. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Global_optimization.

Wikipedia. (2024b, January 10th). Hill Climbing. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Hill_climbing.

Wikipedia. (2024c, January 11th). Random Search. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Random_search.

Wikipedia. (2024d, January 15th). Local Search. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Local_search_(optimization),

Wikipedia. (2024e, January 18th). Context-Free Grammar. [Online]. Retrieved from: https://en.wikipedia.org/wiki/Context-free_grammar.

Wikipedia. (2024f, June 18th). NP-hardness. [Online]. Retrieved from: https:// https://en.wikipedia.org/wiki/NP-hardness.