ISSN 1870-4069

Non-bio-inspired Metaheuristics in Software Testing

Alfredo Delgado-Santiago[1], Angel J. Sánchez-García[1],
Marcela Quiroz-Castellanos[2]

[1] Universidad Veracruzana,
Facultad de Estadística e Informática,
Mexico

[2] Universidad Veracruzana,
Instituto de Investigaciones en Inteligencia Artificial,
Mexico

dltunasd@gmail.com, {angesanchez, mquiroz}@uv.mx

Abstract. The software testing phase usually consumes a large part of the development of software projects trying to get as many defects as possible in the final product. Different strategies have been approached to optimize this phase of the testing stage; such is the case of metaheuristics, algorithms with the ability to find high-quality solutions in a relatively short time. This research seeks to analyze the current status of the application of metaheuristics that assist in software testing phase activities, only the most representative non-bio-inspired algorithms (NBA) are surveyed, being Hill Climbing and Local Search the most used. The main activities of the software testing stage where NBA was implemented were test case generation, test data generation and test case prioritization (redundancy reduction). It was concluded that NBAs used on their own are only viable in some activities of the software testing phase. As future work, it is proposed to investigate the use of hybrid algorithms and approaches in software testing phase.

Keywords: Metaheuristic, software testing, optimization, systematic literature review.

1 Introduction

The need for software systems to be free of defects is increasing. To ensure the quality of the Software, a transcendental phase is the testing phase. In this revolution called Industry 4.0, Artificial Intelligence seeks to automate processes and provide products with autonomous decision-making, among other benefits.

In the software development process, there are studies on the use of metaheuristics at the software testing stage, however, most of these articles focus on bio-inspired algorithms (i.e., algorithms based on nature), and there are several other alternatives that could contribute to this field.

Table 1. Research questions.

Question	Motivation
RQ1.- What are the non-bio-inspired metaheuristics that have been reported at the software testing stage?	To identify the NBAs reported in the software testing phase.
RQ2.- What are the main activities of the testing stage where non-bio-inspired metaheuristics have been applied?	It is important to identify in which software testing activities the algorithms reported in RQ1 have been used, in order to analyze the contributions.
RQ3.- What are the advantages and disadvantages that have been found with the application of non-bio-inspired metaheuristics at the testing stage?	One of the objectives of this research is to describe the strengths and weaknesses of each approach to know in which activities they will be able to obtain better results.
RQ4.- What types of benchmark problems have been used to test non-bio-inspired metaheuristics?	To know the benchmarks used to test the algorithms found, to identify their characteristics and compare results with the proposals generated in future work.

A heuristic method is a tentative and plausible procedure whose purpose is to discover the solution to a particular problem. The heuristic (or simply heuristic) is a method that helps to discover the solution of a problem by making plausible but fallible guesses about what the best thing is to do next [1].

Building on the above, a metaheuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies for developing heuristic optimization algorithms. Notable examples of metaheuristics include genetic/evolutionary algorithms, tabu search, simulated annealing, and ant colony optimization [2].

As the years go by, the systems developed in software projects become more complex, and, consequently, so do the tests. More and more alternatives are emerging to address the different optimization approaches in the development of software testing, such as the use of NBAs.

The use and research of these alternatives have shown that their use in the field of software testing could represent an optimization to this rigorous stage.

This paper is organized as follows: Section 2 describes background and related work. Section 3 details the method used to execute this Systematic Literature Review. In Section 4, the results obtained from this work are presented. Finally, Section 5 draws the main conclusions and proposes future work.

2 Background and Related Work

Artificial Intelligence is a discipline that has supported each of the phases of software development, such as requirements, design, coding, testing and maintenance.

In the testing phase, especially optimization algorithms have been used to generate test cases or identify defects. However, most of the strategies used are based on evolutionary algorithms or bio-inspired algorithms.

In a manual search of related work, a Systematic Review about Bio-inspired computation in software testing was found [3], which talks about the importance of

Table 2. Keywords and synonyms identified.

Concept	Synonyms and related terms
Metaheuristic	Metaheuristics, Meta-heuristic
Software engineering	-
Software testing	Testing
Benchmark	Benchmarks
Path algorithm	Trajectory algorithm
Local search	Explorative search
Neighborhood search	-
GRASP	Greedy Randomized Adaptive Search Procedure
Simulated annealing	-

software testing and how optimization processes are handled in the software testing stage and objective decision making. However, no Systematic Literature Review on the application of metaheuristics not based on evolutionary algorithms in Software Testing was identified.

Therefore, the purpose of this Systematic Literature Review (SLR) is to complement the identified systematic review with metaheuristics that are not bioinspired. With this, it will be possible to have a balance of both approaches, to describe advantages, disadvantages and possible combinations between them to improve the results obtained in the literature.

In addition, it is intended to identify software testing activities (such as test case generation, branch coverage, defect identification, among others), where different search optimization approaches have been used. Finally, it is expected to identify benchmarks of various software testing activities, in which different optimization and search approaches can be tested to compare future contributions.

3 Research Method

The method proposed by Kitchenham and Charters [4], which was proposed to carry out SLR Software Engineering area, was selected for this work. The planning phase is presented below.

3.1 Research Questions

This section shows the Research Questions (RQs) proposed for this research work. Research Questions are described in Table 1.

3.2 Search Strategy and Data Sources

This section shows the keywords and related terms that were used in the search strings. The decision to include the terms “Path Algorithms”, “Local Search”, “Neighborhood

Table 3. Data source.

Database	**Website**
IEEE Xplore	https://ieeexplore.ieee.org/Xplore/home.jsp
ACM	https://dl.acm.org/
Springer Link	https://link.springer.com/
Science Direct	https://www.sciencedirect.com/

Table 4. Inclusion criteria.

ID	Description
IC1	The study was published between 2017 and 2022.
IC2	Full access to the study.
IC3	The title or abstract of the study contains the search term 'Software testing' and its synonyms with another search term.
IC4	Reading the abstract, the study hints at answering at least one research question.

Search", "GRASP" and "Simulated Annealing" was made because they were considered important search terms in the context of this SLR.

The proposed search string from the key terms is described below.

("Software testing" OR Testing) AND ("Software engineering") AND ("trajectory algorithm" OR "Local search" OR "Explorative search" OR "Neighborhood search" OR "Greedy Randomized Adaptive Search Procedure" OR GRASP OR "Simulated Annealing" OR "Tabu search") AND (benchmark OR benchmarks).

Due to Science Direct operators limit (OR and AND), the string was adapted for use in this research source, so that the string was as similar as possible to the main string. The following search string was used in Science Direct.

"Software testing" AND "Software engineering" AND ("trajectory algorithm" OR "Local search" OR "Explorative search" OR "Neighborhood search" OR GRASP OR "Simulated annealing" OR "Tabu search").

Table 3 shows the databases used as source for this SLR.

3.3 Selection of Primary Studies

In this section, the criteria for the selection of primary studies are presented. The inclusion and exclusion criteria are presented in Table 4 and Table 5, respectively.

3.4 Selection Procedure

The selection procedure was made up of the following stages:

- Stage 1. Primary studies are filtered according to IC1 and IC2.
- Stage 2. The primary studies are removed according to EC1 and EC2.
- Stage 3. Primary studies are filtered according to IC3 and IC4.
- Stage 4. The primary studies are removed according to EC3.

Table 5. Exclusion Criteria.

ID	Description
EC1	Studies that are not written in the English language.
EC2	Studies that are outreach articles, posters, books, chapters, presentations, abstracts, or tutorials.
EC3	Duplicated studies.

Table 6. Application of inclusion and exclusion criteria by stage.

Database	First Results	Stage 1	Stage 2	Stage 3	Stage 4
ACM Digital Library	484	193	126	7	7
IEEE Xplore	38	18	18	4	4
SpringerLink	806	263	261	3	3
Science Direct	300	116	110	5	5
Total	**1, 628**	**590**	**515**	**19**	**19**

4 Results

The search process was carried out according to the SLR planning, executing the final search string in each of the selected sources. As it is shown in Table 6, the greatest reduction of studies occurred during Stage 3 of the selection process.

The list of references of the 19 primary studies selected for analysis can be found in [5]. The template for data extraction from each primary study can be found in [6].

Nineteen articles were obtained from the application of the search criteria; however, it was detected that only 9 of them contained information relevant to this research, since they answered at least one research question. Fig. 1 shows the proportion of the type of paper found (journal paper or conference paper).

It is worth mentioning that no dominant journal or conference was found. That is, each primary study belongs to a different Journal or conference. Fig. 2 shows the distribution of the years of publication of the selected studies, with 2018 being the most dominant year on this topic.

Next, the report of the results obtained by answering each research question is presented.

4.1 RQ1: What Were the Non-Bio-Inspired Metaheuristics that Have Been Reported at the Software Testing Stage?

Fig. 3 shows the frequency of the algorithms found in this investigation. Several types of NBA were identified: Hill climbing, Random search, Simulated annealing, Greedy Algorithm, LIPS (Linearly Independent Path-based Search), Neighborhood search and Ls-Sampling, being Search based approaches the most studied in the Software testing stage.

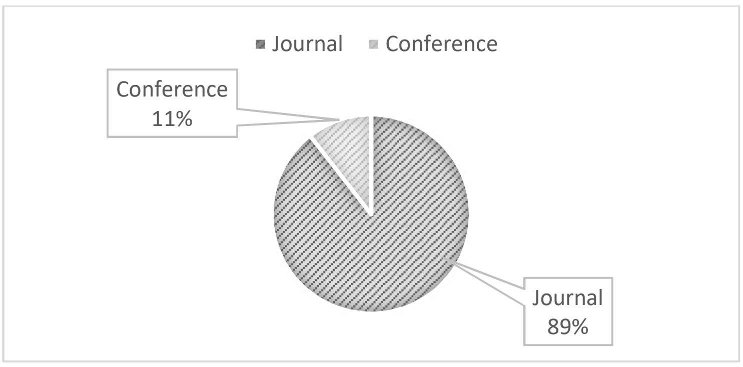
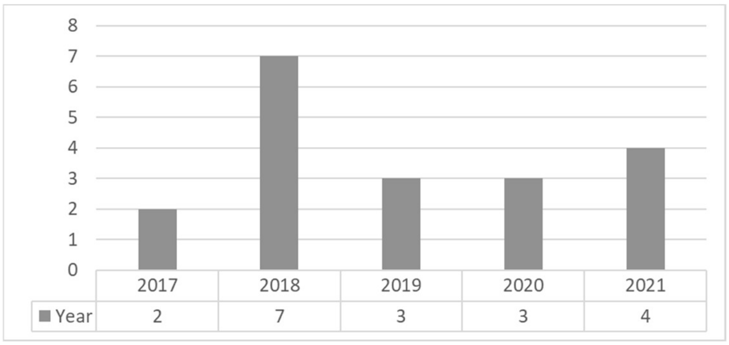


Fig. 1. Distribution by types of publication.



	2017	2018	2019	2020	2021
■ Year	2	7	3	3	4

Fig. 2. Distribution by year in selected databases.

Hill climbing tends to be used for use case prioritization, as it is an optimization algorithm. Due to its variants, the Search-based approaches was used numerous times in the articles, such as Hill climbing, Random search, Neighborhood search and Ls-Sampling.

4.2 RQ2: What are the Main Activities of the Testing Stage where Non-Bio-Inspired Metaheuristics Have Been Applied?

The purpose of this research question is to identify the different activities of the testing stage where the Algorithms detected in **RQ1** were applied. Fig. 4 shows the most addressed testing phase activities.

Four activities were reported for the software testing phase where NBAs assist; most of the algorithms were used for test case prioritization; most of the algorithms detected are optimization algorithms. Hill Climbing [7, 8, 9], Greedy Algorithm [7, 9], Random Search [9], Simulated Annealing [8] and Neighborhood search [8] were used for this activity. Hill Climbing [10] and Simulated Annealing [10] algorithms were used for test data generation.

The findings of this research indicate that, compared with bio-inspired algorithms, NBAs non-bio-inspired algorithms did not perform well doing test cases generation. In this activity, the use of LIPS [11] was detected. According to the findings NBAs were

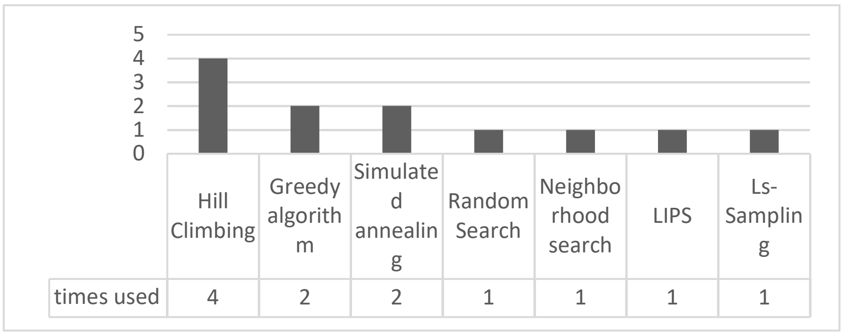


Fig. 3. Frequency of reported algorithms.

used to sort and group test cases into test suites. Only the use of Ls-Sampling [12] was reported for this activity of the software testing stage. LS-Sampling is a Search-based sampling approach. Search-based approaches showed the most versatility for performing software testing stage activities.

4.3 RQ3.- What are the Advantages and Disadvantages Found with the Application of Bio-Inspired Algorithms at the Testing Stage?

The algorithms reported in primary studies were mostly compared against bio-inspired algorithms; the results of these comparisons were that, commonly NBAs did not show a significant improvement compared to bio-inspired algorithms; however, due to their simplicity, i.e., the small number of lines of code required for their execution, these algorithms are optimal to apply to specific or reduced tasks.

In large systems, according to the findings, it is more advisable to use multi-objective approaches due to the number of functionalities they cover. Most of the reported algorithms were used to test case prioritization.

4.4 RQ4.- What Types of Benchmark Problems Have Been Used to Test Non-Bio-Inspired Metaheuristics?

Most of the NBAs were tested with specific problems, i.e., problems proposed by the authors to simulate a real-life scenario [9, 8, 11]. Triangle was used in two studies [9, 12]. One study was tested with the Corpus SF110 benchmark [12]. The results can be seen in Fig. 5.

5 Conclusions and Future Work

Industry 4.0 provides an automation of processes that help the manufacture of many essential products such as software. Nowadays, software systems are becoming more and more complex, therefore, testing those systems is becoming more and more complicated. So, the use of algorithms that help optimize software testing activities is a necessity.

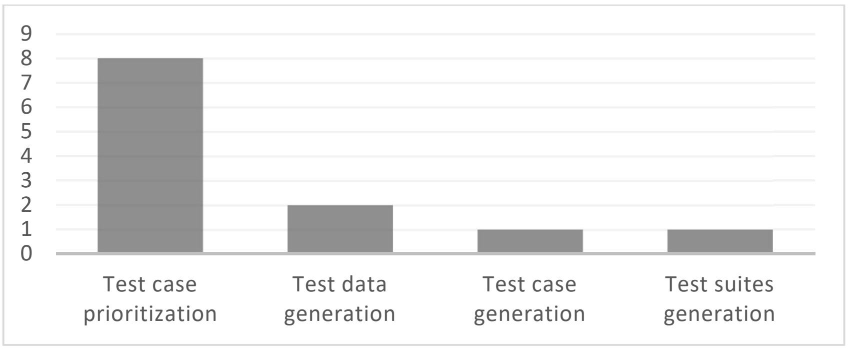


Fig. 4. Activities of the testing phase.

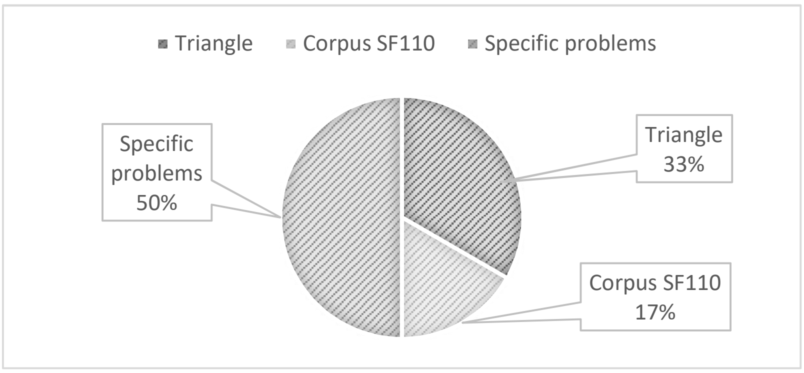


Fig. 5. Reported benchmark.

According to the information gathered in this SLR, non-bio-inspired metaheuristics mean a great improvement to the software testing process, however, bio-inspired algorithms are still superior in this aspect [12]. In carrying out this research, the use of hybrid algorithms was detected, showing a significant improvement (according to the benchmarks) in the efficiency when performing activities in the testing process.

It is proposed as future work the research of hybrid algorithms and approaches, since according to studies [13, 14] they represent a significant improvement over the use of individual approaches.

References

1. Feigenbaum, E. A, Feldman, J: Computers and thought. McGraw-Hill (1963)
2. Sörensen, K., Glover, F. W.: Metaheuristics. Encyclopedia of Operations Research and Management Science, pp. 960–970 (2013) doi: 10.1007/978-1-4419-1153-7_1167
3. Gómez-San-Gabriel, J. E., Sánchez-García, Á. J., Cortés-Verdín, K.: Cómputo bio-inspirado en la prueba de software: Una revisión sistemática de la literatura, vol. 149, no. 11, pp. 125-134 (2020)
4. Kitchenham, B. A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Keele University and Durham University Joint Report (2007)

5. Appendix A: Primary studies references (2022) drive.google.com/file/d/1PyDa70JBLd9jmNR_36YkFfdLziIXSJy/view?usp=sharing
6. Appendix B: Template for information analysis (2022) drive.google.com/file/ d/1ehs-f0SNZcDeZmc3PoBaFTY8ypM7FDk/view?usp=sharing
7. Lou, Y., Chen, J., Zhang, L., Hao, D.: A survey on regression test-case prioritization. Advances in Computers, Elsevier, pp. 1–46 (2019) doi: 10.1016/ bs.adcom.2018.10.001
8. Zamli, K.Z., Safieny, N., Din, F.: Hybrid test redundancy reduction strategy based on global neighborhood algorithm and simulated annealing. In: Proceedings of the 7th International Conference on Software and Computer Applications, Association for Computing Machinery, pp. 87–91 (2018) doi: 10.1145/3185089.3185146
9. Prado-Lima, J. A., Vergilio, S. R.: Search-based higher order mutation testing. In: Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing, Association for Computing Machinery, pp. 87–96 (2018) doi: 10.1145/3266003.3266013
10. Nosrati, M., Haghighi, H., Vahidi-Asl, M.: Using likely invariants for test data generation. Journal of Systems and Software, vol. 164, pp. 110549 (2020) doi: 10.1016/j.jss.2020.110549
11. Luo, C., Sun, B., Qiao, B., Chen, J., Zhang, H., Lin, J., Lin, Q., Zhang, D.: LS-sampling: an effective local search based sampling approach for achieving high t-wise coverage. In: Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Association for Computing Machinery, pp. 1081–1092 (2021) doi: 10.1145/3468264.3468622
12. Panichella, A., Kifetew, F. M., Tonella, P.: A large scale empirical comparison of state-of-the-art search-based test case generators. Information and Software Technology, vol. 104, pp. 236–256 (2018) doi: 10.1016/j.infsof.2018.08.009
13. Lu, C., Zhong, J., Xue, Y., Feng, L., Zhang, J.: Ant colony system with sorting-based local search for coverage-based test case prioritization. In: IEEE Transactions on Reliability, vol. 69, no. 3, pp. 1004–1020 (2020) doi: 10.1109/ tr.2019.2930358
14. Monemi-Bidgoli, A., Haghighi, H.: Augmenting ant colony optimization with adaptive random testing to cover prime paths. Journal of Systems and Software, vol. 161, pp. 110495 (2020) doi: 10.1016/j.jss.2019.110495