







Application of Genetic Algorithms in Software Engineering: A Systematic Literature Review

Pablo F. Ordoñez-Ordoñez^{1,2}(✉) , Milton Quizhpe¹,
Oscar M. Cumbicus-Pineda^{1,3} , Valeria Herrera Salazar¹ ,
and Roberth Figueroa-Diaz¹ 

¹ Facultad de Energía, CIS, Universidad Nacional de Loja,
Ave. Pío Jaramillo Alvarado, La Argelia, Loja, Ecuador
pfordonez@unl.edu.ec

² ETSI Sistemas Informáticos, Universidad Politécnica de Madrid,
Calle Alan Turing s/n, 28031 Madrid, Spain

³ Departamento de Ciencias de la Computación e Inteligencia Artificial,
Universidad del País Vasco, Leioa, Spain

Abstract. Software engineering was born from the need to establish an adequate and efficient methodology for the development of the software, not using appropriate methods in the software produces a large number of errors, today on Software has evolved drastically and is considered as a discipline that has its own principles and requirements to obtain more structured solutions with planning, development and culmination. The genetic algorithms present an alternative to solve problems of optimization in the software engineering, therefore in this work a systematic literature review (SLR) of the application and technologies was carried out of the genetic algorithms in it. The results are presented based on 127 initial documents which, after passing through a review protocol, were reduced to 20 chords to the research topic, where it was indicated that the greatest application is in the tests of software.

Keywords: Optimization · Software engineering
Genetic algorithms · Genetic programming · Evolutionary algorithms

1 Introduction

Software Engineering is an application of the systematic and disciplined approach to the design, development, operation and maintenance of software [31], the detection of software vulnerabilities is a critical step to ensure the quality and security of the software [28] however, software testing is a time-consuming and costly task, consuming almost 50% of the resources for the development of the system software [27, 29]. Automated software testing is better than manual testing, however, very few test data generation tools are currently available commercially [6].

Genetic algorithms (GA) are formed from the evolutionary algorithms conceived by John Holland in the United States well known during the late sixties [7], these have been widely used in software engineering as a method of optimization [13]. Consequently, this research focuses on the technology and application of genetic algorithms in the problems of software engineering.

This systematic review is based on the protocol proposed by [16,17,22]: According to the information that exists on the application of genetic algorithms in software engineering, it was targeted: “Specify the most recent research about the applications of genetic algorithms in software engineering” as a guide for this review. Taking into account that in the phases of software development a greater optimization and resources are needed, the following research questions (RQ) have been determined:

- **RQ1:** What problem does the genetic algorithm solve in software engineering?
- **RQ2:** What application and technology do genetic algorithms have in Software Engineering?

In Sect. 2, the SLR is executed, the result of which is described in Table 2. On the basis of these results, Sect. 3 presents the most notable details and the synthesis argued and discussed in the 20 primary studies and Sect. 4 concludes as research questions the consequences of the review, and specific lines of research for the future.

2 Review Protocol Development

2.1 Research Identification

The criterion for the choice of search sources was based on web accessibility and the inclusion of search engines that allow to carry out advanced queries, in this way the following were used: ACM [2], IEEE library [1], SCOPUS Library [11] and RRAAE [23].

For the choice of keywords it was considered: research questions and keywords of previously reviewed articles: optimization, software engineering, genetic algorithms, genetic programming, evolutionary algorithms, requirements and testing.

Searches were performed using logical operators: (AND) and (OR) and the following inclusion criteria were considered for the search:

- Take as relevant current publications since 2012.
- Search results in the area of science and computation.
- Documents in Spanish and English language.
- Search the Abstract of the article for keywords.

The Table 1 correspond to the search chains in the different bibliographic sources.

Table 1. Bibliographic sources and search strings.

Digital library ACM:
(+genetic +algorithms +software +engineering +software +requirements)
(+genetic +algorithms +software +requirements)
(+genetic +algorithms +software +testing)
(+optimization +computer +systems +genetic +algorithms)
Digital library IEEE:
((“Abstract” :software engineering) AND “Abstract” :genetic algorithms))
((“Abstract” :evolutionary algorithms,) AND “Abstract” :software requirements)
((“Abstract” :genetic algorithms) AND “Abstract” :software requirements)
Digital library scopus:
(TITLE-ABS-KEY (optimization software engineering) AND TITLE-ABS-KEY (genetic algorithms)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA, “COMP”))
(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (software requirements) AND TITLE-ABS-KEY (software design)) AND PUBYEAR > 2012 AND PUBYEAR < 2017 AND (LIMIT-TO (SUBJAREA, “COMP”))
(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (web software) OR TITLE-ABS-KEY (software patron)) AND PUBYEAR > 2012 AND (LIMIT-TO (SUBJAREA, “COMP”))
(TITLE-ABS-KEY (genetic algorithms) AND TITLE-ABS-KEY (engineering phase of the software development)) AND PUBYEAR > 2012
Digital library RRAE:
(Todos los Campos:ingenieria de software y Todos los Campos:algoritmos geneticos)
(Todos los Campos:algoritmos geneticos y Todos los Campos:pruebas de software))
((Todos los Campos:diseño de software y Todos los Campos:algoritmos geneticos))

2.2 Selection of Primary Studies

Once the results were obtained with the searched questions, the criterion that will be followed in the execution of the review for the selection and evaluation of primary studies was described. The results of the search that have not been relevant to the stated objective have been discarded taking into account the following exclusion criteria:

- Studies that do not contain information that helps answer RQ1 and/or RQ2 research questions.
- In the summary and content there is no information about the application of the algorithms in software engineering.
- Work that is poorly structured and unclear.
- The conclusion must have relevant information for the investigation.

2.3 Data Extraction

The Table 2 presents the relevant information for each of the selected articles (S01...S20) according to the search by pointing out elements such as: (RQ1) Problem that genetic algorithms solve in software engineering and (RQ2) Application and technology of genetic algorithms in software engineering.

Table 2. Data extraction from the primary studies.

ID-Ref.	Article	Problem-solution	Application-technology
S01-[32]	Minimizing test suites in software product lines using weight-based genetic algorithms	The redundant test cases in the software production lines	Application of genetic algorithms (gas) based on weight to minimize the set of tests
S02-[8]	Cost-priority cognizant regression testing	The redundant cases in the regression tests	Genetic algorithms have been used to optimize the prioritization of test
S03-[25]	UML modeling of load optimization for distributed computer systems based on genetic algorithm	The allocation of resources in distributed systems	Implementation of the genetic algorithms to optimize the waiting time in the allocation of resources you are in distributed systems
S04-[5]	Improved heuristics for solving OCL constraints using search algorithms	Limitations in UML models	Improve the existing heuristic to solve OCL restrictions. Using search algorithms
S05-[15]	Critical components testing using hybrid genetic algorithm	The critical test components	Optimization based on hybrid genetic algorithms
S06-[20]	Development of a framework for test case prioritization using genetic algorithm	The prioritization of test cases in the maintenance phase	Framework for prioritization of test cases using genetic algorithms
S07-[26]	Random or genetic algorithm search for object-oriented test suite generation	Unit test in object-oriented classes	Genetic algorithm in unit tests
S08-[3]	Minimizing feature model inconsistencies in software product lines	Selection of features for the configuration of a product	Optimization of the process of selection of the characteristics by genetic algorithms
S09-[31]	A dynamic approach for retrieval of software components using genetic algorithm	Software reuse	Recovery through the use of genetic algorithms
S10-[19]	Component-based software system test case prioritization with genetic algorithm decoding technique using Java platform	Regression tests	Software test prioritization framework based on component
S11-[4]	Optimization of soft cost estimation using genetic algorithm for NASA software projects	Cost estimation with COCOMO model	Adjustment of the parameters of the COCOMO coefficients using genetic algorithms

(continued)

Table 2. (*continued*)

ID-Ref.	Article	Problem-solution	Application-technology
S12-[30]	Software quality assurance for object-oriented systems using meta-heuristic search techniques	Quality of software for object-oriented systems	Use of search techniques for software optimization of fault prediction
S13-[14]	A novel approach for test case generation from UML activity diagram	Tests based on models	Application of genetic algorithms for the generation of test cases
S14-[10]	A novel strategy for automatic test data generation using soft computing technique	Generation of automatic test data	AG-based heuristics for automatic generation of test sets
S15-[33]	Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram	The scenarios of test cases	Prioritization of test scenarios using hybrid genetic algorithm
S16-[21]	Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management	Estimation of times in the planning of software development projects	Parallel evolutionary algorithms to solve the problem term of programming with limitations in the management of projects
S17-[12]	Automatic generation of basis test paths using variable length genetic algorithm	Destination route tests	Genetic algorithm for the generation of test trajectories
S18-[24]	Retrieving sequence diagrams using genetic algorithm	Software reuse	Genetic algorithm for the determination of the similarity of the graphic representations of the sequence diagrams
S19-[9]	Methods for cost estimation in software project management	Bad estimation of costs in the management of software projects	Estimation of costs in the management of software projects using genetic algorithms
S20-[18]	Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks	Estimation of effort in the management of software projects	Estimation models for software projects using neural networks and genetic algorithms

2.4 Data Synthesis

Once the primary studies have been determined, it can be observed that in the present review, 127 have been taken into account for the analysis, of which the

primary ones were considered, 20 of which the following synthesis is illustrated (Table 3):

Table 3. Summary of reviewed studies.

Sources	Studies	Relevants	Selected
ACM	616	27	8
IEEE	245	40	5
SCOPUS	450	60	7
RRAAE	0	0	0
Totals	1311	127	20

In Fig. 1, the incidence of the studies that have been analyzed is shown, in this table it is shown that only one of them S01 [32] has maintained a very significant influence and takes as a direct reference for another 13 studies. S15 and S16 show that the impact that has been obtained has been medium since they have been taken into account more than 4 times and in the rest of the studies the impact they have shown has been low since they have only been considered in less than 4 studies and in three cases S08, S09 and S13 the incidence has been null since they have not been considered for other references.

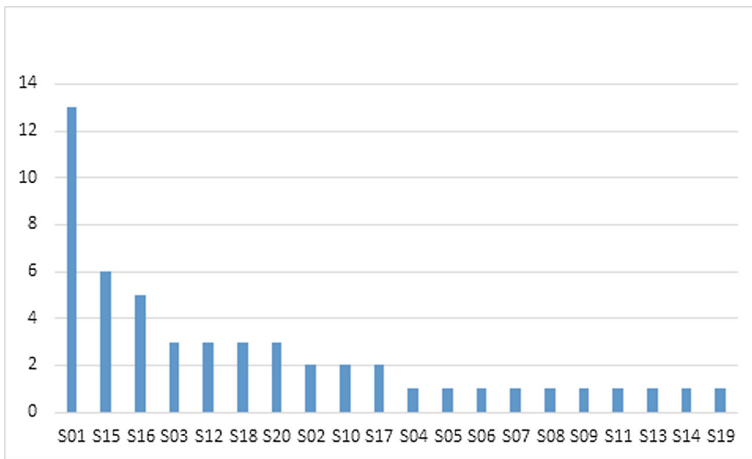


Fig. 1. Synthesis by impact

Figure 2 shows the direct participation of the authors in the different studies that were analyzed, they are the most prominent, as you can see the authors have not collaborated in another study different from the one mentioned in each

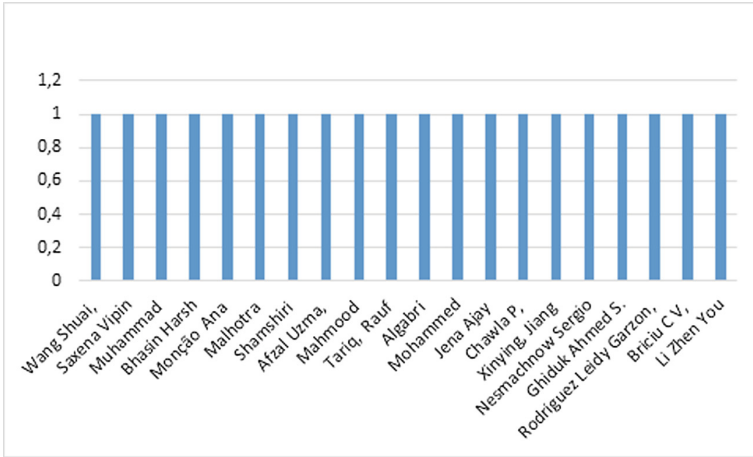


Fig. 2. Synthesis by author.

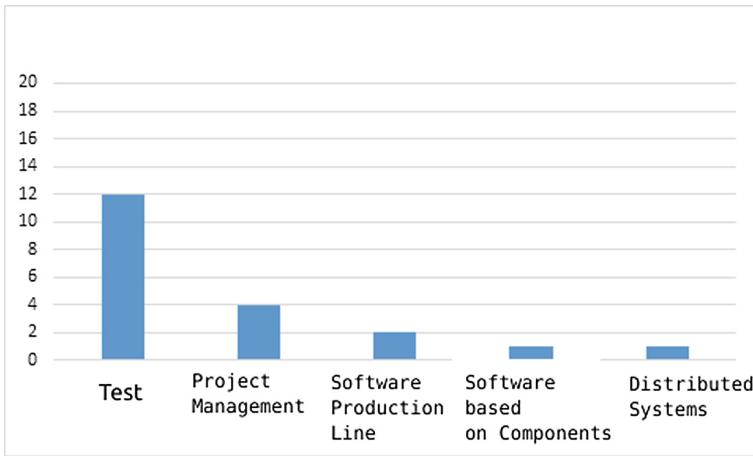


Fig. 3. Synthesis of the AG applications.

one of the sections. Taking into account the results shown, it can be concluded that the topics are related and the results obtained are as desired, so it has not been considered feasible to carry out a second study to corroborate the data obtained in it.

Figure 3 shows clearly that the main application is in the testing phase of a project, and these can be regressive, initial or terminal. It is also shown that these algorithms can be applied in the management of projects, which helps to optimize the overall level of the project as indicated by 4 of the studies, regarding the Production, Distributed Systems and Software Engineering based on components, only the application of these algorithms has been tested in a

single project for each one of them, therefore they are not considered as relevant study points for the application of genetic algorithms.

Figure 4 shows the technology applied by the genetic algorithms, it is denoted that Java is used mostly for development, not only because it is a free programming language, it is also multiplatform and of greater boom. in the development. Another of the technologies is C++, the studies that were analyzed showed that several of them used this language for the programming of these applications. A large percentage of these studies have not implemented any development technology, since they were only applied at some design or initial stage. The Matlab, OLC and Web technologies have been applied only in two of the studies analyzed each, so it can be defined that these are not very relevant or have little boom in the development of applications, the Most development technologies that were used are object oriented and free guidelines.

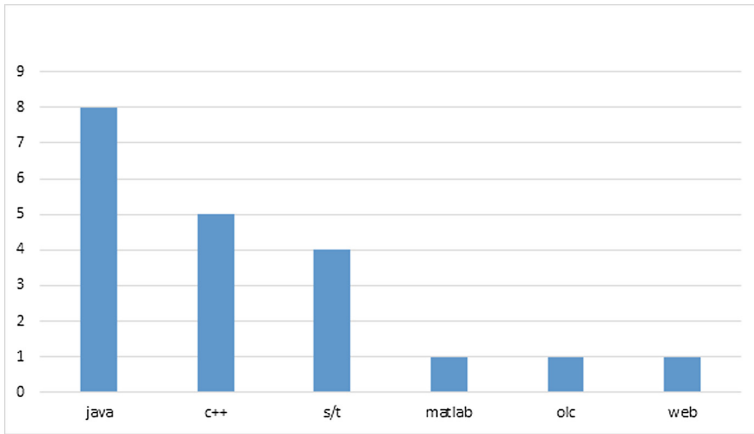


Fig. 4. Synthesis of technology.

3 Discussion

S01 and S08 agree that the test in the Software Product Lines can be minimized appropriately using genetic algorithms since the total number of test cases is reduced, improving its efficiency.

S02, S14 and S17 propose to the Genealogical Algorithms as a solution for the automatic generation of tests, since the software development process invests at least 50% of the total cost in the testing process. software.

S03, S04, S13, S15 and S18 agree that the benefits of software reuse multiply if carried out in the early stages of software development. Sequence diagrams are commonly used to model the functionality of software systems in the early stages of the software development life cycle.

In S05, S06 and S10 they emphasize that the cases of prioritization tests is an essential task that reduces the test effort in the maintenance phase to a

considerable degree. These articles propose a framework for the prioritization of test cases using a tool based on a genetic algorithm, developed in the Java language.

S07 and S12 agree that the identification of faults in the very early phase of life cycle software development is very necessary. This helps software developers focus more on quality assurance, use the workforce in the right perspective, and reduce the cost of debugging software development in particular.

S11, S16, S19 and S20 agree that the deadline, limited programming in project management, is a problem of optimization with greater relevance in software engineering and other real-life situations. They are responsible for the planning of the activities that must be completed before the specified dates to resolve the programming period with limitations in the management of projects. The genetic algorithms have been designed to calculate precise solutions in the times of reduced execution.

4 Conclusion and Future Work

Regarding the technology used, it is noted that the highest percentage of studies carried out are based on Java, a programming language for free and object-oriented guidelines, which is used at all levels of programming. On the other hand a large number of the works: S04, S13, S115, S18, S19, were still in stages of study and many of these have been made in their development process in UML or in its initial phases.

There are primary studies S02, S03, S06, S07, S10, S12 that are specific and offer a clear answer on the problems that the application of General Algorithms has solved. Likewise, the application of these algorithms allowed the optimization of several of the stages in the development of the Software Engineering, in addition to the technologies in which a software can be executed. Genetic algorithm are not limited or do not show restrictions on those that currently exist. Based on the primary studies S02, S04, S09, S11, S15 analyzed there are several scopes in the application of genetic algorithms in the Software development, most of these are shown in the design and testing stages, which mainly streamlines the control of programming errors and optimizes the time and costs that are generally the stages in the which concentrates most of the effort and budget of a project.

Finally, the applications of genetic algorithms are of greater height according to their timeline, denoting a great use in initial stages or tests, since the results that have been obtained in each of these have clearly shown that in all they have achieved the optimization of processes, which demonstrates the improvement of the cost-time-effort ratio. However, the need for applications in the software life cycle is present, when this process is caused by frameworks and agile methodologies that involve great interaction with the user.

References

1. IEEE Xplore Digital Library. <https://ieeexplore.ieee.org/Xplore/home.jsp>
2. ACM: The ACM Digital Library. <https://www.acm.org/>
3. Afzal, U., Mahmood, T., Rauf, I., Shaikh, Z.A.: Minimizing feature model inconsistencies in software product lines. In: Proceedings of the 17th IEEE International Multi-Topic Conference Collaborative and Sustainable Development of Technologies, IEEE INMIC 2014, pp. 137–142 (2015). <https://doi.org/10.1109/INMIC.2014.7097326>
4. Algabri, M., Saeed, F., Mathkour, H., Tagoug, N.: Optimization of soft cost estimation using genetic algorithm for NASA software projects. In: 2015 5th National Symposium on Information Technology: Towards New Smart World, pp. 1–4 (2015). <https://doi.org/10.1109/NSITNSW.2015.7176416>, <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7176416>
5. Ali, S., Iqbal, M.Z., Arcuri, A.: Improved heuristics for solving OCL constraints using search algorithms. In: 16th Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 1231–1238 (2014). <https://doi.org/10.1145/2576768.2598308>
6. Alzabidi, M., Kumar, A.: Automatic software structural testing by using evolutionary algorithms for test data generations. *J. Comput. Sci.* **9**(4), 390–395 (2009). http://paper.ijcsns.org/07_book/200904/20090453.pdf
7. Baccichetti, F., Bordin, F., Carlassare, F.: λ -Prophage induction by furocoumarin photosensitization. *Experientia* **35**(2), 183–184 (1979). <https://doi.org/10.1007/BF01920603>. <http://www.gbv.de/dms/ilmenau/toc/01600020X.PDF>
8. Bhasin, H.: Cost-priority cognizant regression testing. *ACM SIGSOFT Softw. Eng. Notes* **39**(3), 1–7 (2014). <https://doi.org/10.1145/2597716.2597722>
9. Briciu, C.V., Filip, I., Indries, I.I.: Methods for cost estimation in software project management. In: IOP Conference Series: Materials Science and Engineering, vol. 106, no. 1 (2016). <https://doi.org/10.1088/1757-899X/106/1/012008>, <http://www.scopus.com/inward/record.url?eid=2-s2.0-84960154391&partnerID=tZOtx3y1>
10. Chawla, P., Chana, I., Rana, A.: A novel strategy for automatic test data generation using soft computing technique. *Front. Comput. Sci.* **9**(3), 346–363 (2015). <https://doi.org/10.1007/s11704-014-3496-9>. <http://www.scopus.com/inward/record.url?eid=2-s2.0-84938208965&partnerID=40&md5=6b7065f7903d0a046c17613f79b6ecd1>
11. Elsevier B.V.: Scopus. <https://www.scopus.com/home.uri>
12. Ghiduk, A.S.: Automatic generation of basis test paths using variable length genetic algorithm. *Inf. Process. Lett.* **114**(6), 304–316 (2014). <https://doi.org/10.1016/j.ipl.2014.01.009>
13. Hsinyi, J.: Can the genetic algorithm be a good tool for software engineering searching problems? In: Proceedings of the International Conference on Computer Software and Applications, vol. 2, pp. 362–364 (2006). <https://doi.org/10.1109/COMPSAC.2006.123>
14. Jena, A.K., Swain, S.K., Mohapatra, D.P.: A novel approach for test case generation from UML activity diagram. In: 2014 International Conference on Issues Challenges in Intelligent Computing Techniques, pp. 621–629 (2014). <https://doi.org/10.1109/ICICT.2014.6781352>, <http://www.scopus.com/inward/record.url?eid=2-s2.0-84899098078&partnerID=tZOtx3y1>

15. Jeya Mala, D., Sabari Nathan, K., Balamurugan, S.: Critical components testing using hybrid genetic algorithm. *ACM SIGSOFT Softw. Eng. Notes* **38**(5), 1 (2013). <https://doi.org/10.1145/2507288.2507309>. <http://dl.acm.org/citation.cfm?doid=2507288.2507309>
16. Kitchenham, B.: Procedures for performing systematic reviews. Keele University, Keele, UK 33(TR/SE-0401), 28 (2004). <https://doi.org/10.1109/METRIC.2004.1357885>
17. Kitchenham, B., et al.: Systematic literature reviews in software engineering: a tertiary study. *Inf. Softw. Technol.* **52**(8), 792–805 (2010). <https://doi.org/10.1016/j.infsof.2010.03.006>
18. Li, Z.Y.: Predicting project effort intelligently in early stages by applying genetic algorithms with neural networks. *Appl. Mech. Mater.* **513–517**, 2035–2040 (2014). <https://doi.org/10.4028/www.scientific.net/AMM.513-517.2035>
19. Mahajan, S., Joshi, S.D., Khanaa, V.: Component-based software system test case prioritization with genetic algorithm decoding technique using Java platform. In: 2015 International Conference on Computing Communication Control and Automation, pp. 847–851 (2015). <https://doi.org/10.1109/ICCUBEA.2015.169>, <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7155967>
20. Malhotra, R., Tiwari, D.: Development of a framework for test case prioritization using genetic algorithm. *ACM SIGSOFT Softw. Eng. Notes* **38**(3), 1 (2013). <https://doi.org/10.1145/2464526.2464536>. <http://dl.acm.org/citation.cfm?doid=2464526.2464536>
21. Nesmachnow, S.: Efficient parallel evolutionary algorithms for deadline-constrained scheduling in project management. *Int. J. Innov. Comput. Appl.* **7**(1), 34–49 (2016). <https://doi.org/10.1504/IJICA.2016.075468>
22. Pino, F., García, F., Piattini, M.: Revisión sistemática de mejora de procesos software en micro, pequeñas y medianas empresas. *Rev. Española Innovación Calid. e Ing. del Softw.* REICIS **2**(1), 6–23 (2006). <http://redalyc.uaemex.mx/pdf/922/92220103.pdf>
23. RRAAE: Red de Repositorio de Acceso Abierto del Ecuador. <http://www.rraae.org.ec/>
24. Salami, H.O., Ahmed, M.: Retrieving sequence diagrams using genetic algorithm, pp. 324–330. IEEE Computer Society (2014). <https://doi.org/10.1109/JCSSE.2014.6841889>
25. Saxena, V., Arora, D., Mishra, N.: UML modeling of load optimization for distributed computer systems based on genetic algorithm. *SIGSOFT Softw. Eng. Notes* **38**(1), 1–7 (2013). <https://doi.org/10.1145/2413038.2413043>
26. Shamshiri, S., Rojas, J.M., Fraser, G., Mcminn, P., Court, R.: Random or genetic algorithm search for object-oriented test suite generation? In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, pp. 1367–1374 (2015). <https://doi.org/10.1145/2739480.2754696>, <http://dl.acm.org/citation.cfm?id=2754696>
27. Sharma, C., Sabharwal, S., Sibal, R.: A survey on software testing techniques using genetic algorithm. *Int. J. Comput. Sci. Issues* **10**(1), 381–393 (2013). <https://arxiv.org/ftp/arxiv/papers/1411/1411.1154.pdf>
28. Shuai, B., Li, M., Li, H., Zhang, Q., Tang, C.: Software vulnerability detection using genetic algorithm and dynamic taint analysis. In: 2013 Proceedings of the 3rd International Conference on Consumer Electronics, Communications and Networks, CECNet 2013, pp. 589–593 (2013). <https://doi.org/10.1109/CECNet.2013.6703400>, <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6703400>

29. Sommerville, I.: Software engineering (2010). <https://doi.org/10.1111/j.1365-2362.2005.01463.x>
30. Suresh, Y.: Software quality assurance for object-oriented systems using meta-heuristic search techniques, pp. 441–448 (2015)
31. Vodithala, S.: A dynamic approach for retrieval of software components using genetic algorithm. <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7339085>
32. Wang, S., Ali, S., Gotlieb, A.: Minimizing test suites in software product lines using weight-based genetic algorithms. In: Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation, pp. 1493–1500 (2013). <https://doi.org/10.1145/2463372.2463545>
33. Wang, X., Jiang, X., Shi, H.: Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram, pp. 854–857. IEEE Computer Society, November 2015. <https://doi.org/10.1109/ICSESS.2015.7339189>