ISSN 2007-9737

Parameter Setting in Ant Colony-Based Software Testing

Saúl Domínguez Isidro*, Ángel J. Sánchez García, Saraí Castillo Hernández

Universidad Veracruzana,
Mexico

{sauldominguez, angesanchez}@uv.mx, cashdezsarai@gmail.com

Abstract. Ant Colony Optimization (ACO) has been widely applied in search-based software testing (SBST) for tasks such as test case generation, test suite optimization, and fault detection. However, the effectiveness of ACO in these applications is highly dependent on parameter tuning, which directly influences convergence, efficiency, and reliability. This study investigates the parameter settings used in ACO-based software testing by systematically reviewing existing research. The analysis identifies the most used parameter values and examines their impact on testing performance. Results indicate that parameters such as pheromone evaporation rate, heuristic weight, and the number of ants significantly influence the algorithm's effectiveness, with a balanced trade-off between pheromone influence and heuristic information being a prevalent approach. The findings provide insights into optimizing ACO parameterization for software testing and suggest future research directions for adaptive tuning mechanisms to enhance its efficiency.

Keywords. Software testing, ant colony optimization, parameter settings, search-based software testing, literature review.

1 Introduction

This work extends the study presented by Castillo-Hernández, Domínguez-Isidro, and Sánchez-García (2024), which conducted a systematic literature review (SLR) on the application of Ant Colony Optimization (ACO) in software testing.

That study synthesized existing research on ACO as a search-based approach within the field of software testing, highlighting its adoption, areas of application, and reported benefits.

Search-Based Software Testing (SBST) leverages optimization techniques to automate and improve testing activities, such as test case generation, fault detection, and test suite optimization (McMinn, 2011). Inspired by the foraging behavior of ants, ACO utilizes pheromone-based communication to iteratively refine candidate solutions, making it particularly effective in addressing complex software testing challenges (Dorigo, Birattari, & Stützle, 2006).

Building upon the findings of the previous systematic review, this study further investigates the role of ACO in software testing by addressing the following research questions (Castillo-Hernández, Domínguez-Isidro, & Sánchez-García, 2024):

RQ1: In which software testing activities has ACO been reported?
RQ2: In what types of software systems has ACO been applied for software testing?
RQ3: What characteristics do the software testing problems solved by ACO have?
RQ4: What are the benefits of implementing ACO in software testing activities?
RQ5: What are the challenges of implementing ACO in software testing activities?

In software engineering, ACO has been successfully adapted for test case generation, test suite optimization, and automated test data generation by leveraging its capability to explore large solution spaces efficiently (Castillo-Hernández, Domínguez-Isidro, & Sánchez-García, 2024; Jatana, Suri, & Rani, 2017; Suri & Singhal, 2012).

While ACO has demonstrated its effectiveness as a search-based technique, its performance is highly dependent on parameter tuning, which significantly influences its convergence, efficiency, and reliability (Jatana, Suri, & Rani, 2017; Vats, Mandot, & Gosain, 2014). Therefore, this study expands on prior research by analyzing the parameter configurations used in ACO-based software testing approaches. This analysis is grounded in the studies identified in the SLR, providing insights into the impact of parameter tuning on testing effectiveness and efficiency. In this context, we formulate the following additional research question:

RQ6: What parameter values are used in ACO-based software testing?

The remainder of this paper is structured as follows. Section 2 provides a detailed explanation of the problem statement and the role of parameter tuning in ACO-based software testing. Section 3 describes the research methodology, including the systematic literature review process and selection criteria. The results of the study, including the identified parameter configurations and their impact, are presented in Section 4. Finally, Section 5 concludes the study, summarizing key findings and suggesting future research directions.

2 Problem Statement

The algorithm is particularly useful in software testing for finding optimal test cases that maximize coverage while minimizing redundancy (Castillo-Hernández, Domínguez-Isidro, & Sánchez-García, 2024; Jatana, Suri, & Rani, 2017; Suri & Singhal, 2012; Vats, Mandot, & Gosain, 2014).

– Initialize Parameters – Define the number of ants m, pheromone evaporation rate ρ, pheromone influence α, heuristic influence β, and the number of iterations.
– Generate Ant Solutions – Each ant constructs a feasible solution (i.e., a test suite) by probabilistically selecting test cases based on pheromone intensity and heuristic information.
– Evaluate Solutions – Each solution is assessed using a fitness function that considers test coverage, execution cost, and redundancy.
– Update Pheromones – Pheromone levels are updated based on the quality of the generated solutions.
– Evaporate Pheromones – Pheromones gradually evaporate to prevent premature convergence.
– Iterate Until Convergence – Repeat the process until a stopping criterion is met, such as a maximum number of iterations or convergence to an optimal solution.

2.1 Solution Construction

Each ant incrementally constructs a solution by choosing a test case t_i from a candidate set T based on a probabilistic rule. The probability of selecting test case t_i at step k is given by (Dorigo, Birattari, & Stützle, 2006):

$$\mathrm{P}_{\mathrm{i,j}}^{\mathrm{k}} = \frac{[\tau_{\mathrm{i,j}}]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}{\sum_{j \in T}[\tau_{i,j}]^{\alpha} \cdot [\eta_{i,j}]^{\beta}}, \quad (1)$$

where:

– $P_{i,j}^{k}$ is the probability of selecting test case t_j after t_i.
– $\tau_{i,j}$ is the pheromone intensity associated with the transition from t_i to t_j.
– $\eta_{i,j}$ is a heuristic function that represents the desirability of selecting t_j, often based on coverage and fault detection capability.
– α and β control the influence of pheromone and heuristic information, respectively.
– In software testing, $\eta_{i,j}$ can be defined as:

$$\eta_{i,j} = \frac{Cov(t_j)}{Cost(t_j)}, \quad (2)$$

where $Cov(t_j)$ is the code coverage provided by t_j, and $Cost(t_j)$ is the execution time or resource consumption of the test case.

2.2 Pheromone Update

After all ants construct solutions, pheromone levels are updated to reinforce high-quality solutions. The pheromone update consists of two parts: evaporation and reinforcement (Dorigo, Birattari, & Stützle, 2006).

Evaporation:

$$\tau_{i,j} \leftarrow (1-\rho) \cdot \tau_{i,j}, \quad (3)$$

where ρ is the pheromone evaporation rate ($0 < \rho \leq 1$). This step prevents over-concentration of pheromones, ensuring diversity in the search process.

Reinforcement:

$$\tau_{i,j} \leftarrow \tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k}, \tag{4}$$

where $\Delta\tau_{i,j}^{k}$ is the pheromone deposited by ant k, calculated as:

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{1}{F(S^k)}, & if\ t_j \text{ part of } S^k \\ 0, & otherwise \end{cases}, \tag{5}$$

where t_j is a test case; $F(S^k)$ is the fitness function of the test suite S^k, which can be defined in software testing as:

$$F(S^k) = w_1 \cdot Cov(S^k) - w_2 \cdot Redundancy(S^k) - w_3 \cdot Cost(S^k), \tag{6}$$

where $Cov(S^k)$ is the total code coverage achieved by the test suite. $Redundancy(S^k)$ measures redundant test cases in the suite. $Cost(S^k)$ represents execution cost. w_1, w_2, w_3 are weights balancing these factors.

2.3 Stopping Criteria

The algorithm iterates until one of the following conditions is met:

– A predefined number of iterations is reached.

– The best solution does not improve over successive iterations.

– A coverage threshold (e.g., 95% branch coverage) is met.

The proper configuration of parameters in ACO is crucial for achieving optimal performance in software testing tasks. Parameters such as pheromone influence, heuristic weight, evaporation rate, and the number of ants significantly impact the algorithm's efficiency and effectiveness. Analyzing the most used parameter settings in state-of-the-art proposals provides valuable insights into their impact on solution quality. By identifying trends and limitations in existing implementations, potential areas for improvement can be explored, leading to more efficient and adaptive ACO-based approaches in software testing.

3 Research Method

This study follows the systematic literature review (SLR) methodology proposed by Kitchenham, Budgen, and Brereton (2015). The objective is to analyze the use of ACO in software testing activities, its scope, the characteristics of the problems addressed, as well as the reported benefits and challenges of its implementation.

3.1 Planning

The search strategy was based on the quasi-gold standard approach proposed by Zhang, Babar, and Tell (2011). Initially, two relevant conferences, GECCO and ICSE, were examined, but no studies on the topic were found. As a result, a manual search was conducted in academic databases. IEEE Xplore was selected to evaluate the search string, as Zhang et al. (2011) identified it as the most commonly used search engine in related studies.

The quasi-gold standard studies revealed two key terms: "Software Testing" and "Ant Colony Optimization". Terms unrelated to the topic, such as "Traveling Salesman Problem" and "Complex Circuits," were excluded to refine the search. Various search strings were generated and evaluated based on sensitivity (relevant studies retrieved ÷ total relevant studies) and precision (relevant studies retrieved ÷ total retrieved studies). A recommended sensitivity range of 70% to 80% was followed (Kitchenham, Budgen, & Brereton, 2015), and the search string that best balanced these factors was selected for the automated search.

The search was carried out in four leading academic databases in software engineering and computational intelligence: IEEE Xplore, ACM Digital Library, ScienceDirect, and SpringerLink. The study selection process was structured into

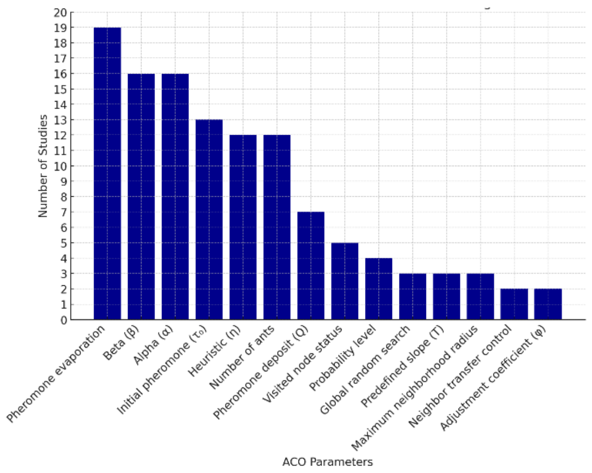

Fig. 1. Distribution of ACO Parameters in SBST

four stages, applying well-defined inclusion and exclusion criteria:

- First stage: Applied search engine filters, including time range (2013– April 2025) and language (English). Studies not published in conferences, congresses, workshops, or journals were excluded, along with abstracts, book chapters, and presentation slides.
- Second stage: Studies without full access and secondary studies were excluded, while those whose titles suggested relevance to at least one research question were included.
- Third stage: Abstracts were reviewed to determine whether the study addressed at least one research question, discarding those that used ACO in domains unrelated to software testing.
- Fourth stage: Primary studies that provided answers to at least one research question were included, while duplicates were removed.

3.2.1 Research Conduction

The initial search retrieved 1,552 primary studies, distributed across the selected databases as follows: 134 from IEEE Xplore, 139 from ACM Digital Library, 1,031 from SpringerLink, and 248 from ScienceDirect. After applying the selection criteria, 167 studies remained, of which 36 were ultimately selected, accounting for 21% of the filtered set.

Throughout the selection process, the number of relevant studies was gradually refined. After the first stage, 592 studies remained, with SpringerLink contributing the largest share. By the second stage, this was further reduced to 58, with IEEE Xplore leading in the number of retained studies. The third stage narrowed the set down to 42, and in the final selection, 36 studies were retained: 20 from IEEE Xplore, 5 from ScienceDirect, and 11 from SpringerLink, with no relevant studies identified from ACM Digital Library.

A notable trend in the selected studies is that 78% come from conference proceedings, while the remaining 22% originate from journal publications. Furthermore, IEEE Xplore contains slightly older studies, whereas SpringerLink includes more recent contributions from the past five years.

In contrast to the original review, this extended study introduces an additional research question (RQ6) focused on parameter settings in ACO-based software testing. To address this, a secondary data extraction process was conducted using the same 36 primary studies previously selected. This approach is consistent with the guidelines of Kitchenham et al. (2015), where the same corpus of primary studies can be analyzed from different analytical perspectives. RQ6 is considered transversal and complementary to the original questions, as it explores how parameter configuration, an often underreported yet critical aspect, influences the implementation and performance of ACO in the identified contexts.

4 Results

As described in Section 2, the first step involves configuring the parameters used for Ant Colony Optimization. These parameters are problem-specific and context-dependent, with their assigned values strongly influencing the algorithm's performance.

An analysis of the literature revealed that, since most authors introduce modifications to the algorithm, each adaptation requires specific parameters. The parameters identified in the literature, along with their descriptions and the studies that reference them, are presented in Table 1.

Table 1. ACO parameters in SBST

Parameter	Description	Study
Number of ants (m)	Determines search space diversity, allowing broader exploration. A higher number of ants generally leads to more extensive exploration, though in some cases, it solely dictates search effort rather than variety.	PS-1, PS-5, PS-10, PS-14, PS-15, PS-19, PS-21, PS-22, PS-23, PS-32, PS-35, PS-36
Initial pheromone value (τ_0)	Some studies initialize all nodes and edges with a pheromone value, which evolves throughout the algorithm. This value influences the decision-making process of ants (PS-2).	PS-1, PS-3, PS-4, PS-5, PS-6, PS-8, PS-9, PS-12, PS-21, PS-25, PS-28, PS-30, PS-35
Heuristic value (η)	Represents the weight assigned to a node, depending on the problem. It indicates the quality of the node, or the path formed if the next node is selected. This parameter is part of the probability selection formula for the next node.	PS-3, PS-4, PS-6, PS-8, PS-25, PS-28
Alpha (α)	Determines the relative importance of pheromone levels (PS-1). A higher α value increases the influence of pheromone levels in node selection while reducing the effect of heuristics	PS-1, PS-2, PS-4, PS-5, PS-6, PS-8, PS-10, PS-12, PS-14, PS-15, PS-23, PS-25, PS-28, PS-32, PS-35, PS-36
Beta (β)	Determines the relative importance of heuristic information (PS-1). A higher β value increases the influence of heuristic information during node selection while reducing the impact of pheromone levels.	PS-1, PS-2, PS-4, PS-5, PS-6, PS-8, PS-10, PS-12, PS-14, PS-15, PS-23, PS-25, PS-28, PS-32, PS-35, PS-36
Pheromone deposit factor (Q)	A constant that regulates ant decision-making, increasing or decreasing the likelihood of node selection based on pheromone trails and heuristic values.	PS-5, PS-9, PS-12, PS-14, PS-15, PS-35, PS-36
Pheromone evaporation factor	Controls the rate at which pheromone traces evaporate (i.e., the decay of pheromone values). Higher values result in faster evaporation.	PS-2, PS-5, PS-9, PS-10, PS-12, PS-14, PS-15, PS-16, PS-19, PS-21, PS-22, PS-23, PS-24, PS-25, PS-26, PS-29, PS-32, PS-34, PS-35

Table 1 provides an overview of the key parameters used in ACO-based software testing. Notably, the number of ants and parameters governing pheromone behavior and heuristics (including α and β) are the most frequently referenced in the literature, as they form the fundamental components of ACO.

Furthermore, a trend toward integrating ACO with local and global search techniques has emerged. As noted by Dorigo et al. (2006), local search is optional in ACO and highly problem-dependent; however, its integration has proven beneficial in the context of software testing problems. Figure 1 illustrates the distribution of ACO parameters utilized in software testing studies, ranking them by frequency of occurrence.

The pheromone evaporation factor emerges as the most frequently mentioned parameter, appearing in 19 studies. This suggests its critical role in ACO applications for software testing. A higher evaporation rate leads to faster convergence, reducing the likelihood of exploring diverse solutions, while a lower evaporation rate helps maintain diversity for a longer duration, potentially avoiding premature convergence to suboptimal solutions.

Following pheromone evaporation, alpha (α) and beta (β) are the next most frequently cited parameters, each appearing in 16 studies. These parameters control the balance between pheromone influence and heuristic information, affecting how ants select paths in the search space. Their high occurrence underscores their importance in adjusting the exploration-exploitation trade-off.

The initial pheromone value (τ_0) and heuristic factor (η) are also widely used, each appearing in more than 12 studies. The initial pheromone value influences the starting bias of the search process, while the heuristic factor determines how much problem-specific information guides the ants.

Other parameters, such as pheromone deposit (Q), visited node status, and probability level, are moderately represented, suggesting that while they are relevant, their impact might be more problem-dependent.

Less frequently used parameters include maximum neighborhood radius, predefined slope (T), adjustment coefficient (φ), global random search control, and neighbor transfer control, each appearing in only a few studies. These parameters are typically associated with ACO variants that integrate local and global search strategies, indicating their application in specialized problem settings.

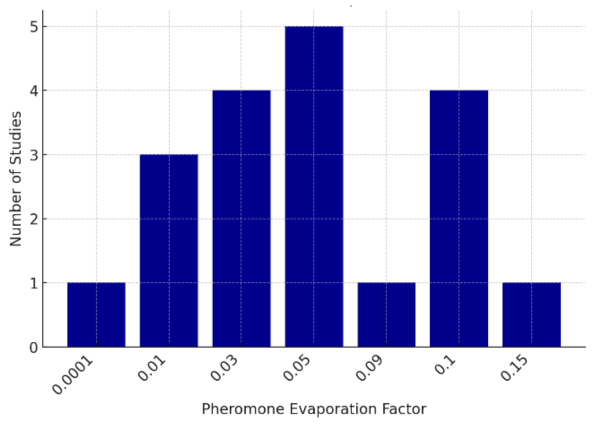

Fig. 2. Distribution of Pheromone Evaporation Factor Values

Less frequently used parameters include maximum neighborhood radius, predefined slope (T), adjustment coefficient (φ), global random search control, and neighbor transfer control, each appearing in only a few studies. These parameters are typically associated with ACO variants that integrate local and global search strategies, indicating their application in specialized problem settings.

4.1ACO Parameter Values

An analysis of the studies identified the specific values assigned to each ACO parameter. The following figures illustrate the distribution of parameter values across various studies.

The number of ants varies across studies, with 10 being the most frequently used value, followed by 30. In two primary studies (PS-8 and PS-19), the number of ants is defined as equal to the number of test cases.

Most authors set the initial pheromone (τ_0) value to 1 for all nodes. This ensures equal selection probability for all nodes in the early iterations. Concerning the heuristic value (η), only two distinct values have been reported for this parameter. The most used value is 2, appearing in four out of six studies that reference this parameter.

Most studies opt for Alpha (α) = 1, meaning pheromone levels are fully considered when selecting the next node. Regarding Beta (β), most studies adopt β = 1, prioritizing heuristic values in node selection. This fact indicates a standardized approach where both pheromone information and heuristics contribute equally to decision-making. This balance ensures steady convergence without excessive bias, making it a widely used and robust configuration across optimization problems.

The most frequently used values for pheromone deposit factor (Q) is 1 and 0.5. Studies that use Q = 0.5 likely aim to maintain more exploratory behavior, while those using Q = 1 prioritize rapid convergence towards promising solutions.

Figure 2 shows the pheromone evaporation factor reported; compared to other parameters, the values assigned to pheromone evaporation are relatively low. This suggests a preference for slow pheromone evaporation, preventing premature convergence to local optima.

For the visited status parameter, a 0-initialization value was reported in four studies (PS-3, PS-6, PS-8, PS-28), allowing ants to access all nodes during solution construction. Similarly, the probability level parameter was initialized to 0 in two studies (PS-6, PS-28), meaning all nodes initially had equal selection probability.

In studies that incorporated ACO with local search (and in some cases, global search), the maximum neighborhood radius was explicitly defined in most cases. However, PS-21, and PS-22, both assigned a value of 10. For the predefined slope parameter, the two studies that used it assigned a value of 1. A similar trend was observed for the adjustment coefficient (φ), where PS-21, used 1 and PS-24, used 0.5.

Finally, for the global random search control limit and neighbor transfer control limit, both studies assigned values of 0.5 and 0.3, respectively. This implies that global search was prioritized over local search, as ants performed global searches half the time while only one in three instances involved movement to a neighboring position.

4.2.1 Types of Software Systems Evaluated

The analysis of the selected primary studies revealed that ACO-based approaches have been applied across various software system types. Identifying the types of systems under evaluation provides context for understanding the relevance and adaptability of parameter configurations.

Although not all studies explicitly stated the nature of the systems being tested, a substantial portion offered enough information to infer the software domain or structural model involved.

Many studies focused on object-oriented systems, where ACO was primarily applied to support unit test generation. In these contexts, the structure and behavior of software components were modeled through class diagrams, object interactions, and method invocations. Studies such as PS-10 and PS-14 exemplify this trend, where the goal was to automate the generation of test sequences that cover class methods or paths through object dependencies. The object-oriented paradigm offered a modular test structure, allowing ACO to be tailored to navigate and prioritize test paths effectively.

Another group of studies applied ACO in the context of embedded systems, which are characterized by strict resource constraints, real-time requirements, and specialized control mechanisms. In these systems, test optimization becomes critical to ensure that limited test budgets are effectively utilized. Study PS-23, for example, employed ACO to generate test cases that meet specific execution and resource criteria. The application of ACO in embedded systems highlights its potential in scenarios where functional correctness and performance constraints must be satisfied.

Several studies abstracted the system under test using graph-based models, typically derived from UML diagrams such as activity or state machine diagrams. These models represent the system's dynamic behavior or control flow, facilitating model-based testing strategies. Studies PS-4 and PS-5 used ACO to traverse these diagrams and generate paths that satisfy coverage criteria. This approach is particularly practical in the early stages of software development, where behavioral specifications exist, but implementation details may still be evolving.

Some studies addressed component-based software systems, emphasizing the need for interface-level testing. In these cases, ACO was utilized to prioritize or select test cases that verify the interaction among software components. For instance, PS-25 applied ACO to ensure quality assurance in systems composed of independently developed and integrated modules. These scenarios often require precise control over parameter configurations to balance test effectiveness and efficiency.

Finally, a subset of studies focused on general-purpose or unspecified software systems, where the primary concern was the testing technique rather than the system domain. In such studies, ACO was used as a general search-based method for structural testing, regression testing, or test suite minimization. Despite lacking specific domain characteristics, these studies contributed valuable insights into how ACO can be parameterized and adapted for broad testing goals.

This classification provides a clearer understanding of the contexts in which ACO-based testing has been applied. It also emphasizes the potential influence that system type may exert on selecting and tuning ACO parameters, underlining the importance of considering the testing domain when designing and evaluating ACO configurations.

5 Conclusions and Future Work

This study analyzed the role of parameter settings in Ant Colony Optimization (ACO)-based software testing by systematically reviewing the existing literature. The investigation identified the most frequently used parameter values and their impact on test case generation, test suite optimization, and overall testing efficiency. The results indicate that key parameters, such as pheromone evaporation rate, heuristic influence, and the number of ants, play a crucial role in determining the algorithm’s effectiveness.

A significant finding is that most studies adopt a balanced approach between pheromone influence and heuristic information, ensuring steady convergence without excessive bias. Furthermore, parameter tuning strategies are highly problem-dependent, with variations in the assigned values reflecting the specific needs of different software testing tasks.

Pheromone evaporation is the most configurable parameter because it controls convergence speed and solution diversity, making it fundamental to optimizing ACO-based search processes. Alpha (α) and beta (β) are essential for adjusting search behavior, influencing how much

ISSN 2007-9737

pheromone or heuristic information guides decisions. The frequency of initial pheromone value (τ_0) and heuristic factor (η) suggests that initialization strategies are critical in ACO applications. Lesser-used parameters are more problem-specific, often appearing in studies that employ ACO in combination with local and global search techniques.

The insights gained from this study contribute to a deeper understanding of ACO parameterization in software testing. Future work may focus on developing adaptive parameter tuning mechanisms to further enhance the robustness and efficiency of ACO-based testing approaches.

References

1. **Castillo-Hernández, S., Domínguez-Isidro, S., Sánchez-García, Á. J. (2024).** Ant Colony Optimization in Software Testing: A Systematic Literature Review. Abstraction and Application (accepted).
2. **Dorigo, M., Birattari, M., Stützle, T. (2006).** Ant colony optimization. IEEE Computational Intelligence Magazine, Vol. 1, No. 4, pp. 28–39. Doi: 10.1109/MCI.2006.329691.
3. **ISO/IEC/IEEE. (2022).** International Standard - Software and systems engineering -- Software testing -- Part 1: General concepts (ISO/IEC/IEEE 29119-1:2022E).
4. **Jatana, N., Suri, B., Rani, S. (2017).** Systematic literature review on search based mutation testing. DOAJ (Directory of Open Access Journals).
5. **Kitchenham, B. A., Budgen, D., Brereton, P. (2015).** Evidence-based software engineering and systematic reviews. CRC Press, Vol. 4.
6. **McMinn, P. (2011).** Search-based software testing: Past, present and future. Fourth International Conference on Software Testing, Verification and Validation Workshops. Doi: 10.1109/ICSTW.2011.100.
7. **Suri, B., Singhal, S. (2012).** Literature survey of ant colony optimization in software testing. CSI Sixth International Conference on Software Engineering (CONSEG), Doi: 10.1109/CONSEG. 2012.6349506.
8. **Vats, P., Mandot, M., Gosain, A. (2014).** A comparative analysis of ant colony optimization for its applications into software testing. Innovative Applications of Computational Intelligence on Power, Energy and Controls with Their Impact on Humanity (CIPECH). Doi: 10.1109/CIPECH.2014.7018499.
9. **Zhang, H., Babar, M. A., Tell, P. (2011).** Identifying relevant studies in software engineering. Information and Software Technology, Vol. 53, No. 6, pp. 625–637. Doi: 10.1016/j.infsof.2011.01.004.

References of Primary Studies

PS-1. Totla A. M., Renwa, T., Y., Subbiah, R., C., Dharmaraj, T. B., Prakash, S. O. (2022). Comparative analysis of ant colony optimization and particle swarm optimization for test case prioritization. International Conference on Innovation and Intelligence for Informatics, Computing, And Technologies (3ICT).

PS-2. Agrawal, A. P., Kaur, A. (2017). A comprehensive comparison of ant colony and hybrid particle swarm optimization algorithms through test case selection. Advances in Intelligent Systems and Computing, pp. 397–405.

PS-3. Ahmad, S. F., Singh, D. K., Suman, P. (2018). Prioritization for regression testing using ant colony optimization based on test factors. Advances in Intelligent Systems and Computing, pp. 1353–1360.

PS-4. Arifiani, S., Rochimah, S. (2019). Generating test data using ant colony optimization (ACO) algorithm and UML state machine diagram in gray box testing approach. IEEE Transactions on Emerging Topics in Computational Intelligence.

PS-5. Arora, V., Bhatia, R., Singh, M. (2017). Synthesizing test scenarios in UML activity diagram using a bio-inspired approach. Computer Languages, Systems & Structures, Vol. 50, pp. 1–19.

PS-6. Bian, Y., Li, Z., Zhao, R., Gong, D. (2017). Epistasis based ACO for regression test case prioritization. IEEE Transactions on Emerging Topics in Computational Intelligence, Vol. 1, No. 3, pp. 213–223.

PS-7. **Bidgoli, A. M., Haghighi, H. (2020).** Augmenting ant colony optimization with adaptive random testing to cover prime paths. Journal of Systems and Software, Vol. 161, 110495.

PS-8. **Bidgoli, A. M., Haghighi, H., Nasab, T. Z., Sabouri, H. (2017).** Using swarm intelligence to generate test data for covering prime paths. Lecture Notes in Computer Science, pp. 132– 147.

PS-9. **Biswas, S., Kaiser, M. S., Mamun, S. A. (2015).** Applying ant colony optimization in software testing to generate prioritized optimal path and test data. International Conference on Electrical Engineering and Information Communication Technology (ICEEICT).

PS-10. **Bruce, D., Menéndez, H. D., Barr, E. T., Clark, D. (2020).** Ant colony optimization for object-oriented unit test generation. In Lecture Notes in Computer Science, pp. 29–41.

PS-11. **Carino, S., Andrews, J. H. (2015).** Dynamically testing GUIs using ant colony optimization (T). 30th IEEE/ACM International Conference on Automated Software Engineering (ASE).

PS-12. **Carneiro, S. M., Da Silva, T. A. R., Rabelo, R. De A. L., Silveira, F. R. V., De Campos, G. A. L. (2015).** Artificial immune systems in intelligent agent's test. IEEE Congress on Evolutionary Computation (CEC).

PS-13. **Elanthiraiyan, N., Arumugam, C. (2014).** Parallelized ACO algorithm for regression testing prioritization in Hadoop framework. IEEE International Conference on Advanced Communications, Control and Computing Technologies.

PS-14. **Ferrer, J., Kruse, P. M., Chicano, F., Alba, E. (2015).** Search based algorithms for test sequence generation in functional testing. Information and Software Technology, Vol. 58, pp. 419–432.

PS-15. **Kumar, G., Chopra, V. (2018).** Hybrid approach for automated test data generation. Journal of ICT Standardisation.

PS-16. **Kumar, S., Ranjan, P., Rajesh, R. (2016).** Modified ACO to maintain diversity in regression test optimization. 3rd International Conference on Recent Advances in Information Technology (RAIT).

PS-17. **Liu, W., Xiong, J. (2020).** Research on simplified method of combination test case set for basic software system. IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS).

PS-18. **Mao, C., Xiao, L., Yu, X., Chen, J. (2015).** Adapting ant colony optimization to generate test data for software structural testing. Swarm and Evolutionary Computation, Vol. 20, pp. 23–36.

PS-19. **Mehboob, F., Jilani, A., Abbas, M. (2013).** State based testing using swarm intelligence. Proceedings of Science and Information Conference.

PS-20. **Mirhosseini, S. M., Haghighi, H. (2020).** A search-based test data generation method for concurrent programs. International Journal of Computational Intelligence Systems, Vol. 13, No. 1, pp. 1161.

PS-21. **Mohanty, S., Mohapatra, S. K., Meko, S. F. (2020).** Ant colony optimization (ACO-Min) algorithm for test suite minimization. Advances in Intelligent Systems and Computing, pp. 55–63.

PS-22. **Noguchi, T., Washizaki, H., Fukazawa, Y., Sato, A., Ota, K. (2015).** History-based test case prioritization for black box testing using ant colony optimization. IEEE 8th International Conference on Software Testing, Verification and Validation (ICST).

PS-23. **Panthi, V., Mohapatra, D. P. (2016).** ACO based embedded system testing using UML activity diagram. IEEE Region 10 Conference (TENCON).

PS-24. **Panwar, D., Tomar, P., Harsh, H., Siddique, M. H. (2017).** Improved meta-heuristic technique for test case

ISSN 2007-9737

prioritization. Advances in Intelligent Systems and Computing, pp. 647–664.

PS-25. Prajapati, N., Kumar, N. (2016). Data flow-based quality testing approach using ACO for component-based software development. International Conference on Computing, Communication and Automation (ICCCA).

PS-26. Sayyari, F., Emadi, S. (2015). Automated generation of software testing path based on ant colony. International Congress on Technology, Communication and Knowledge (ICTCK).

PS-27. Sharifipour, H., Shakeri, M., Haghighi, H. (2018). Structural test data generation using a memetic ant colony optimization based on evolution strategies. Swarm and Evolutionary Computation, Vol. 40, pp. 76–91.

PS-28. Silva, D., Rabelo, R., Neto, P. S., Britto, R., Oliveira, P. A. (2019). A test case prioritization approach based on software component metrics.

PS-29. Silva, D., Rabelo, R., Campanha, M., Neto, P. S., Oliveira, P. A., Britto, R. (2016). A hybrid approach for test case prioritization and selection. IEEE Congress on Evolutionary Computation (CEC).

PS-30. Singhal, S., Gupta, S., Suri, B., Panda, S. (2016). Multi-deterministic prioritization of regression test suite compared: ACO and BCO. Advances in Intelligent Systems and Computing, pp. 187–194.

PS-31. Solanki, K., Singh, Y., & Dalal, S. (2015). Test case prioritization: An approach based on modified ant colony optimization (m-ACO). International Conference on Computer, Communication and Control (IC4).

Article received on 02/06/2025; accepted on 07/10/2025.
**Corresponding author is Saúl Domínguez Isidro.*