ISSN 1870-4069

Cómputo bioinspirado en la prueba de software: una revisión sistemática de la literatura

Jethran E. Gómez San Gabriel, Ángel J. Sánchez García,
Karen Cortés Verdín

Universidad Veracruzana,
Facultad de Estadística e Informática,

jethran18@outlook.com,
{angesanchez, kcortes}@uv.mx

Resumen. Uno de los pilares para que la Industria 4.0 pueda ser una realidad, es el desarrollo de Software de calidad que permita la interconectividad y manejo de dispositivos que utilizamos. Para que un producto de software sea de calidad es necesario que sus requisitos sean validados y verificados, y es en la fase de prueba donde se realizan estos procedimientos. Para esto, es necesario que se haga una completa y correcta ejecución de un Plan de pruebas. Sin embargo, por el tiempo que implica desde su planificación hasta su ejecución la fase de prueba, este no puede ser terminado y en el peor de los casos iniciada. Esta Revisión Sistemática de la Literatura identifica las aportaciones que se han hecho con el Cómputo Bioinspirado y otros tipos de algoritmos de optimización en el proceso de la prueba de Software. Es así como, se identificaron 25 artículos, de cuales se analizaron los resultados de los algoritmos utilizados en un proceso de Prueba de Software. A partir de los resultados obtenidos se identificaron los algoritmos que se han utilizado tanto Evolutivos como de Inteligencia Colectiva, en qué problemas de la Prueba de Software han sido aplicados, así como los beneficios que se han obtenido al utilizarlos.

Palabras clave: Inteligencia Artificial, Prueba de Software, Cómputo Bioinspirado, Revisión Sistemática de la Literatura, Calidad de Software.

Bio-Inspired Computing in Software Testing: A Systematic Review of the Literature

Abstract. One of the pillars for Industry 4.0 to become a reality is the development of quality Software that allows interconnectivity and management of the devices that we use. For a software product to be of quality it is necessary that its requirements be validated and verified, and it is in the testing phase that these procedures are carried out. For this, it is necessary to carry out a complete and correct execution of a Test Plan. However, due to the time that the test phase implies from its planning to its execution, it cannot be finished and in the worst case started. This Systematic Literature Review identifies the contributions that

have been made with Bio-inspired Computation and other types of optimization algorithms in the Software testing process. Thus, 25 articles were identified, of which the results of the algorithms used in a Software Testing process were analyzed. From the results obtained, the algorithms that have been used both Evolutionary and Collective Intelligence were identified, in which Software Testing problems they have been applied, as well as the benefits that have been obtained when using them.

Keywords: Artificial Intelligence, Software Testing, Bio-inspired Computing, Systematic Literature Review, Software Quality.

1. Introducción

El desarrollo de software hoy en día está presente en todas las aplicaciones y la infraestructura de la Industria 4.0. Es por ello que la Ingeniería de Software tiene como una de sus metas ofrecer calidad a sus productos. Un software de calidad es reflejado por el uso de metodologías, estándares y una buena ejecución de pruebas. Para el desarrollo de software existen modelos de desarrollo de software que ayudan a tener una buena ejecución de las fases de desarrollo que son: Análisis, Diseño, Construcción, Prueba y Mantenimiento. En cada una de las fases se espera que haya revisiones para el cumplimiento de los requisitos funcionales y no funcionales.

En la fase de pruebas se espera que se verifiquen y validen los requisitos funcionales. En esta fase se busca establecer un mínimo de casos de prueba que abarquen todos los requisitos. Las pruebas de software son realizadas para que el producto salga con el mínimo de defectos posibles. Los casos de prueba (CP) son usados para verificar el cumplimiento de los requisitos del software [1].

Elaborar un caso de prueba puede ser un trabajo fácil pero tedioso ya que se deben generar múltiples entradas y posibles salidas para una sola funcionalidad. No obstante, al realizar una planeación de las pruebas, se suele no dar el tiempo necesario para que puedan ser aplicadas. Por lo tanto, es posible que al poco tiempo dado a la fase de pruebas no se pueda ejecutar una planeación que abarque cada uno de los erquisitos funcionales. Debido a lo cual, es necesario optimizar la generación y ejecución de pruebas.

Por otro lado, uno de los elementos que se maneja en la Industria 4.0 para que podamos tener procesos automatizados y toma de decisiones objetivas, es el uso de Inteligencia Artificial, específicamente los Algoritmos Bioinspirados como una opción para procesos de optimización (numérica o combinatoria). Es por esto que en la presente Revisión Sistemática de Literatura (RSL) se plantean algunas preguntas de investigación que permitan dar un conocimiento sobre el uso de Algoritmos Bioinspirados en las pruebas de software.

Este artículo está organizado como sigue: en la sección 2 se muestra el método de investigación. En la sección 3 se presentan y discuten los principales resultados obtenidos. En la sección 4 y última, se comparten las conclusiones y el trabajo futuro.

Tabla 1. Preguntas de investigación.

Pregunta	Motivación
P1.-¿Qué estrategias de Cómputo Bioinspirado se han usado en la prueba de software?	Conocer las estrategias de Cómputo Bioinspirado que se han aplicado a la fase de Prueba de Software.
P2.- ¿Qué tipos de problemas en la Prueba de Software se han abordado con el Cómputo Bioinspirado?	Conocer el tipo de problemas de Prueba que se han sido abordados con este enfoque.
P3.- ¿Qué beneficios se han obtenido al utilizar Cómputo Bioinspirado en la prueba de Software?	Conocer los beneficios de la aplicación de estas técnicas de optimización en la fase de Prueba.

Tabla 2. Bibliotecas digitales seleccionadas.

Biblioteca	Descripción	Enlace
ScienceDirect	Colección de investigación de Ciencias Físicas e Ingeniería.	https://www.sciencedirect.com/
SpringerLink	Colección de revistas, libros y obras científicas, tecnológicas y médicas.	https://link.springer.com/
IEEE Xplore	Acceso a contenido científico y técnico de libros, revistas, conferencias y cursos.	https://ieeexplore.ieee.org/
ACM Digital Library	Biblioteca digital dirigida principalmente al campo informático.	https://dl.acm.org/

Tabla 3. Criterios de selección.

Criterios de inclusión	Criterios de exclusión
CI1.- Artículos escritos en inglés.	CE1.- Artículos que el año de publicación sea menor al año 2000
CI2. Artículos que en el resumen o introducción toque al menos dos términos de búsqueda empleados en la cadena.	CE2.- Trabajos en presentaciones o reportes técnicos.
CI3.- Artículos que traten sobre el Cómputo Bioinspirado en la prueba de Software.	CE3.- Artículos repetidos

2. Método de investigación

La Revisión Sistemática de la Literatura (RSL) fue basada en la guía provista por Kitchenham y Charters [3]. Esta investigación fue conducida en tres fases: (1) planeación, (2) ejecución de la búsqueda y (3) el reporte.

Tabla 4. Fases de búsqueda para la exclusión.

Fases	Criterios
Fase 1	Artículos que sean menores al año 2000
Fase 2	Artículos que no sean académicos o de investigación
Fase 3	Artículos que no se encuentren escritos en inglés
Fase 4	Artículos que en el resumen o introducción no toquen al menos dos términos de búsqueda
Fase 5	Artículos que no traten sobre el Cómputo Bioinspirado en la prueba de software
Fase 6	Revisión de artículos similares (continuación de trabajos).

Tabla 5. Número de estudios por fase.

Biblioteca digital	Fase 0	Fase 1	Fase 2	Fase 3	Fase 4	Fase 5	Fase 6
SpringerLink	30,095	20,717	8,960	8,947	9	6	1
ScienceDirect	25,295	21,615	16,792	16,792	15	2	0
ACM Digital Library	75,400	43,541	35,076	35,076	5,591	22	12
IEEE Xplore	513	349	74	74	12	12	12

2.1. Planeación

Preguntas de investigación. En esta sección se establecieron las preguntas de investigación (PI) que se desean contestar con esta RSL. A través de ellas se pretende exponer trabajo previo sobre Cómputo Bioinspirado en la Prueba de Software, así como los resultados que se han obtenido al usarlas y encontrar nichos de oportunidad para futuras investigaciones en este campo. Las PI seleccionadas se muestran en la Tabla 1.

Selección de fuentes y estrategia de búsqueda. Las fuentes que se utilizaron para realizar esta RSL fueron las cuatro bibliotecas digitales mostradas en la Tabla 2, debido a que los artículos, al ser publicados en estas bases de datos, pasan por un proceso de arbitraje por expertos en el área. Además, se optaron por estas cuatro fuentes de información ya que contienen artículos del área de la Informática, Ingeniería y disciplinas afines.

Después de la identificación de las palabras clave descritas: *evolutionary*, *Optimization*, *software* y *test*, se procedió a la elaboración de la cadena de búsqueda mostrada a continuación:

((evolutionary) AND (optimization) AND (software test).

Criterios de selección. Para la selección de los estudios a revisar, se definieron los criterios de inclusión y exclusión que se muestran en la Tabla 3.

Dados los siguientes criterios definidos, se procedió a realizar la selección de los estudios primarios. Para la identificación de los estudios se realizó un proceso por fases de búsqueda como se muestra en la Tabla 4.

Tabla 6. Puntos clave de la revisión de los estudios.

Problemas en la prueba de sotware	Estrategia	Beneficio en la prueba de software	Estudios primarios
Búsqueda de ruta más eficiente	Algoritmo Genético/Genetic algorithm (GA)	Mayor cobertura	[5], [25], [26]
Generación de datos de prueba	Optimización de Enjambre de Partículas/ Particle swarm optimization (PSO)	Mayor cobertura	[5], [8], [17]
Generación de datos de prueba	Algoritmo Genético/Genetic algorithm (GA)	Mayor cobertura	[14], [15], [16]
Búsqueda de ruta más eficiente	Colonia de Hormigas/ Ant Colony (AC)	Priorización de rutas prueba	[9], [22]
Búsqueda de ruta más eficiente	Prueba Estadística	Mayor cobertura	[18], [17]
Generación de datos de prueba	Algoritmo Genético/ Genetic algorithm (GA)	Priorización de rutas prueba	[15], [16]
Generación de datos de prueba	Optimización Combinatoria	Sin especificar	[28]
Búsqueda de ruta más eficiente	Recocido Simulado/ Simulated Annealing (SA)	Mayor cobertura	[26]
Generación de datos de prueba	Evolución diferencial/ Differential Evolution (ED)	Sin especificar	[10]
Generación de datos de prueba	Algoritmo Genético Inmune/ Immune Genetic Algorithm (IGA)	Mayor cobertura	[12]
Generación de datos de prueba	Colonia Artificial de Abejas / Artificial Bee Colony (ABC)	Mayor cobertura	[16]
Generación de datos de prueba	Colonia Artificial de Abejas / Artificial Bee Colony (ABC)	Priorización de rutas prueba	[16]
Generación de datos de prueba	Algoritmo Genético/ Genetic algorithm (GA)	Sin especificar	[11]
Generación de datos de prueba	Optimización de Enjambre de Partículas/ Particle swarm optimization (PSO)	Sin especificar	[11]
Generación de datos de prueba.	Varios	Sin especificar	[13]
Mejorar la cobertura de las pruebas	Algoritmos Evolutivos Multiobjetivo / Multi-Objective Evolutionary Algorithms (MOEA)	Mayor cobertura	[6]
Generación de datos de prueba.	Algoritmos Genéticos y Bosque Aleatorio	Mayor cobertura	[24]
Mejorar la cobertura de las pruebas	Optimización de Enjambre de Partículas/ Particle swarm optimization (PSO)	Priorización de casos de prueba	[7]
Mejorar la cobertura de las pruebas	Colonia de Hormigas/ Ant Colony (AC)	Mayor cobertura	[19]
Mejorar la cobertura de las pruebas	Algoritmo Genético/Genetic algorithm (GA)	Mayor cobertura	[19]
Mejorar la cobertura de las pruebas	Colonia de Hormigas/ Ant Colony (AC)	Priorización de ruta prueba	[20]
Mejorar la cobertura de las pruebas	Algoritmo Genético/ Genetic algorithm (GA)	Priorización de ruta de prueba	[21]
Búsqueda de ruta más eficiente	Varios	Mayor cobertura	[27]
Sin especificar	Varios	Sin especificar	[23]

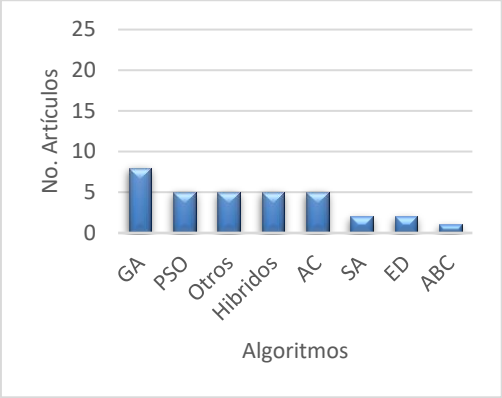

Fig. 1 Algoritmos identificados que se han usado en la Prueba de Software.

Método de síntesis de datos. Para las respuestas las preguntas de investigación, se utilizó el método de síntesis narrativa explicada en la guía de Kitcheman y Charters [3]. Fue posible hacer una extracción de datos mediante una lectura detallada de cada uno de los estudios.

2.2. Ejecución de la búsqueda

Una vez realizada la búsqueda con los criterios definidos en las fuentes seleccionadas, se obtuvieron 53 artículos. Sobre estos artículos obtenidos, se realizó una revisión sobre los títulos, resúmenes y conclusiones, que dieran indicios de las respuestas de alguna pregunta de investigación, obteniendo 35 artículos. Después, se realizó una revisión a profundidad de los artículos que dio como resultado una cantidad de 25 cabe aclarar que se dejaron de lado 7 artículos ya que no se encontraban disponibles. La cantidad de estudios seleccionados por fase de cada una de las bibliotecas se muestra en la Tabla 5.

3. Discusión de resultados

En esta sección se ofrece un resumen de los resultados de la Revisión Sistemática de la Literatura, los resultados de la extracción de datos obtenidas y las respuestas correspondientes de las preguntas de investigación. La Tabla 6 describe estos puntos clave de la extracción de la información.

Para responder a las preguntas, se utilizaron los diferentes hallazgos en los estudios vistos en la Tabla 6.

1. *¿Qué estrategias de Cómputo Bioinspirado se han usado en la prueba de software?*

En la Fig. 1 se pueden observar diferentes tipos de Algoritmos Bioinspirados, de los cuales se mencionan (por sus nombres en inglés) *Genetic Algorithm* (GA), *Particle Swarm Optimization (PSO), Ant Colony (AC), Simulated Annealing (SA), Differential*

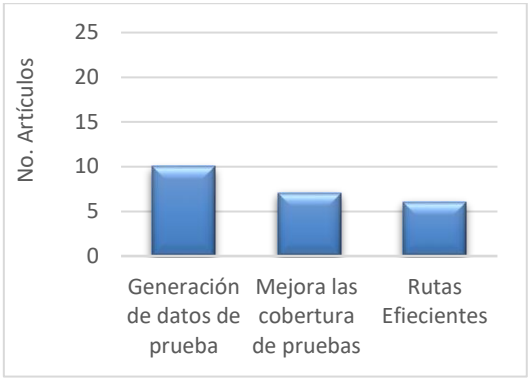

Fig. 2 Problemas de la Prueba de Software abordados con Cómputo Evolutivo.

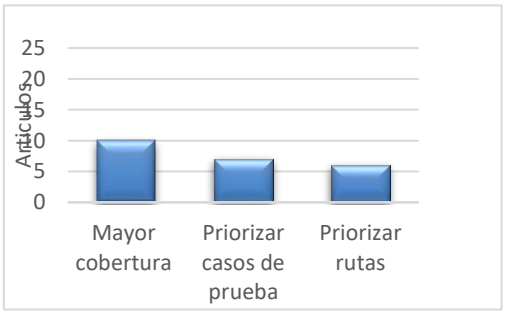

Fig. 3 Beneficios encontrados en los estudios en la fase de prueba.

Evolution (DE) y Artificial Bee Colony (ABC). Además, en la categoría de otros se englobaron otros enfoques de algoritmos como optimización combinatoria y otros algoritmos de optimización [4, 17, 18, 27]. Mientras que los híbridos son una combinación de dos o más algoritmos que su principal función es mejorar las deficiencias de otro algoritmo.

2. *¿Qué tipos de problemas en la prueba de software se han abordado con el Cómputo Bioinspirado?*

En la Fig. 2 se puede observar que existe un mayor número de estudios que hablan sobre la generación de casos de prueba (GCP). Se busca una solución a este problema, ya que cuando se elaboran un mínimo de Casos de Prueba (CP) ocurre que los datos de entrada y salida no abarcan los posibles defectos con los que el sistema cuenta. Algoritmos como AG intentan encontrar las variables de entrada de una manera eficiente y rápida para uno o varios CP.

Esto nos lleva al segundo problema que es la mejora de cobertura de prueba. Cuando se hace un conjunto de CP a un sistema, se espera entrar al mayor número de condiciones posibles [2]. Es por eso por lo que es importante tener un buen conjunto de datos entradas que abarquen el mayor número de rutas.

Por último, se encuentra el problema de *rutas eficientes o priorización de rutas.* En los estudios consultados se encontró una relación con las pruebas de regresión, las cuales están basadas en la idea de probar un sistema que ya fue aplicado en un conjunto de CP, pero al que se le han introducido nuevas funcionalidades o cambios [20]. La pregunta a este problema es ¿Es necesario ejecutar todas las pruebas?, con lo que la optimización busca escoger una ruta específica que tenga una mejor cobertura.

Es importante aclarar que cualquiera de los algoritmos mencionados en la sección anterior puede ofrecer una solución a cualquiera de estos tres problemas, pero hay unos que ofrecen mejor solución a un problema dado que en los estudios se hicieron evaluaciones a diferentes programas con diferentes AE [9, 14].

3. *¿Qué beneficios se han obtenido al utilizar Cómputo Bioinspirado en la prueba de software?*

En la Fig. 3 se puede observar los beneficios encontrados en esta investigación. Se pudo capturar que los beneficios principales son tener una mejor cobertura de los CP ejecutados respecto a otros AE. En los estudios se observó que la cobertura es dada por la función de aptitud aplicado a un producto de software. El individuo óptimo sería el que mayor cobertura encuentre en Grafo de Flujo de Control (GFC) por cada una de sus ramas [14].

En el segundo caso está el priorizar los casos de prueba que garantizan el ejecutar un número mínimo de casos de prueba que abarquen la mayor cobertura, es decir, un CP garantice un criterio dado [15, 21].

En el tercer caso se busca priorizar las rutas con las más se cuenta por diferentes criterios, dos de estos pueden ser basados por costo y tiempo. Dependiendo de cómo sean seleccionados los operadores el algoritmo encontrará la ruta optima.

4. Conclusiones y trabajo futuro

Este trabajo presenta una Revisión Sistemática de la Literatura enfocada a las búsquedas de aplicaciones de Cómputo inspirado en la naturaleza en la Prueba de Software. La motivación principal es la búsqueda de nichos de oportunidad en la Prueba de Software, que ayuden al aseguramiento de la calidad en el proceso y el producto de Software, y por ende, impactar en la calidad de las aplicaciones desarrolladas en la Industria 4.0.

El Cómputo Bioinspirado en el área de la Prueba de Software ha tenido un gran interés en los últimos 15-20 años. Se encontraron trabajos recientes que buscan optimizar en 3 diferentes problemas. También, se encontraron trabajos donde se comparaban resultados de algunos Algoritmos Evolutivos e Inteligencia Colectiva para dependiendo del problema en cuestión.

Con base en los artículos estudiados se observa una gran oportunidad en los algoritmos Bioinspirados Híbridos. Estos presentan una gran ventaja sobre los algoritmos bioinspirados per sé que cuentan con algunas deficiencias. En el uso de otros algoritmos que no necesariamente fueran bioinspirados, como pueden ser los

algoritmos estadísticos o procesos gaussianos, se encuentra una oportunidad de mejorar para el cambio de los operadores que puedan dar mejores resultados.

Se puedo observar que los Algoritmos bioinspirados han tenido una gran aportación en la fase de Prueba de Software, resolviendo tres problemas principales que pueden hacer que los costos y tiempos se reduzcan drásticamente, especialmente en la búsqueda de mayor cobertura de pruebas. Esto debido a que se tiene una buena exploración de la mayoría del espacio de búsqueda, siendo el Algoritmo Genético el más utilizado.

También se pudo observar que varios de los experimentos realizados eran hacia sistemas pequeños, dejando para trabajo futuro el uso de algoritmos bioinspirados para sistemas grandes. Además, con el uso de algoritmos híbridos, se ofrece una nueva mejora en los resultados. Por lo tanto, se encuentra un nicho de oportunidad en el uso del Cómputo Bioinspirado en el área de fase de Pruebas combinado con otros enfoques de optimización e incluso de Aprendizaje Máquina.

Referencias

1. Sánchez, J.: Pruebas de Software. Fundamentos y Técnicas, 1, http://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf (2015)
2. Nidhra, S.: Black box and white box testing techniques - a literature review. International Journal of Embedded Systems and Applications, 2(2), pp. 29−50 (2012)
3. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Department of Computer Science University of Durham, Durham, UK, Tech. Rep. (2007)
4. Sagarna, R., Lozano, J.: Variable search space for software testing. In: International Conference on Neural Networks and Signal Processing, Nanjing, China: IEEE (2003)
5. Koleejan, C., Xue, B., Zhang, M.: Code coverage optimisation in genetic algorithms and particle swarm optimisation for automatic software test data generation. In: IEEE Congress on Evolutionary Computation (CEC). Wellington, New Zelanda: IEEE (2015)
6. do Nascimento Ferreira, T., Kuk, J., Pozo, A., Vergilio, S.: Product selection based on upper confidence bound MOEA/DDRA for testing software product lines. In: IEEE Congress on Evolutionary Computation (CEC), Curitiba, Brazil: IEEE (2016)
7. Hla, K., Choi, Y., Sou-Park, J.: Applying particle swarm optimization to prioritizing test cases for embedded real time software retesting. In: IEEE 8th International Conference on Computer and Information Technology Workshops, Korea: IEEE (2008)
8. Abdul, R., Ejaz, N., Abbas, Q., Rehman, S.U., Shahid, A.A.: PSO based test coverage analysis for event driven software. In: 2nd International Conference on Software Engineering and Data Mining, SEDM´10, pp. 219–224 (2010)
9. Sayyari, F., Emadi, S.: Automated generation of software testing path based on ant colony. In: 2nd International Congress on Technology, Communication and Knowledge, ICTCK´15, pp. 435–440, Institute of Electrical and Electronics Engineers Inc. (2016)
10. Becerra, R.L., Sagarna, R., Yao, X.: An evaluation of differential evolution in software test data generation. In: IEEE Congress on Evolutionary Computation, CEC, pp. 2850–2857 (2009)
11. Laţiu, G.I., Creţ, O.A., Văcariu, L.: Automatic test data generation for software path testing using evolutionary algorithms. In: Proceedings 3rd International Conference on Emerging Intelligent Data and Web Technologies, EIDWT, pp. 1–8 (2012)

12. Bouchachia, A.: An immune genetic algorithm for software test data generation. pp. 84–89, Institute of Electrical and Electronics Engineers (IEEE) (2008)
13. Feldt, R., Poulding, S.: Broadening the search in search-based software testing: it need not be evolutionary. In: Proceedings 8th International Workshop on Search-Based Software Testing, SBST´15, pp. 1–7, Institute of Electrical and Electronics Engineers Inc. (2015)
14. Zhu, Z., Xu, X., Jiao, L.: Improved evolutionary generation of test data for multiple paths in search-based software testing. In: IEEE Congress on Evolutionary Computation, CEC´17, pp. 612–620, Institute of Electrical and Electronics Engineers Inc. (2017).
15. Xu, X., Jiao, L., Zhu, Z.: Boosting search based software testing by using ensemble methods. In: IEEE Congress on Evolutionary Computation, CEC´18, Proceedings, Institute of Electrical and Electronics Engineers Inc. (2018)
16. Mann, M., Tomar, P., Sangwan, O.P.: Bio-inspired metaheuristics: evolving and prioritizing software test data. Applied Intelligence, 48(3), pp. 687–702 (2018)
17. Poulding, S., Alexander, R., Clark, J. A., Hadley, M. J.: The optimisation of stochastic grammars to enable cost-effective probabilistic structural testing. In: GECCO 2013, Proceedings of the 2013 Genetic and Evolutionary Computation Conference, pp. 1477–1484. https://doi.org/10.1145/2463372.2463550 (2013)
18. Deshmukh, J., Horvat, M., Jin, X., Majumdar, R., Prabhu, V. S.: Testing cyber-physical systems through Bayesian optimization. ACM Transactions on Embedded Computing Systems, Vol. 16, Association for Computing Machinery (2017)
19. Chivilikhin, D., Ulyantsev, V., Tsarev, F.: Test-based extended finite-state machines induction with evolutionary algorithms and Ant Colony Optimization. In: GECCO'12 Proceedings of the 14th International Conference on Genetic and Evolutionary Computation Companion, pp. 603–606 (2012)
20. Singh, Y., Kaur, A., Suri, B.: Test case prioritization using ant colony optimization. ACM SIGSOFT Software Engineering Notes, 35(4), 1–7 (2010)
21. Mayan, J. A., Ravi, T.: Test case optimization using hybrid search technique. In: ACM International Conference Proceeding Series, Association for Computing Machinery (2014)
22. Srivastava, P. R.: Structured testing using ant colony optimization. In: ACM International Conference Proceeding Series, pp. 203–207 (2010)
23. Ali, S.: Search-based test optimization for softwere system. In: GECCO 2018 Companion, Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion, pp. 328–348, Association for Computing Machinery, Inc. (2018)
24. Satya, S. P. K., Prasad, S. U. M. D., Gopi, K. D.: Recommendation and regression test suite optimization using heuristic algorithms. In: ACM International Conference Proceeding Series, pp. 202–203, Association for Computing Machinery (2015)
25. Rao, K. K., Raju, G., Nagaraj, S.: Optimizing the software testing efficiency by using a genetic algorithm. ACM SIGSOFT Software Engineering Notes, 38(3), 1–5 (2013)
26. Yan, L., Hu, W., Han, L.: Optimize SPL test cases with adaptive simulated annealing genetic algorithm. In: ACM International Conference Proceeding Series. Association for Computing Machinery (2019)
27. Din, F., Zamli, K. Z.: Fuzzy adaptive teaching learning-based optimization strategy for GUI functional test cases generation. In: ACM International Conference Proceeding Series, pp. 92–96, Association for Computing Machinery (2018)
28. Shir, O. M., Doerr, C., Bäck, T.: Compiling a benchmarking test-suite for combinatorial black-box optimization: A position paper. In: GECCO 2018 Companion, Proceedings of the 2018 Genetic and Evolutionary Computation Conference Companion, pp. 1753–1760, Association for Computing Machinery, Inc (2018)