eJBio

Electronic Journal of Biology, 2016, Vol.12(2): 145-155

Nature-Inspired Metaheuristic Search Strategies

Mukesh Mann[1,*], Pradeep Tomar[1], Om Praksah Sangwan[2], Shivani Singh[1]

1 School of Information and Communication Technology Gautam Buddha University, Greater Noida, Uttar Pradesh, India;

2 Department of Computer Science and Engineering, Guru Jambheshwar University of Science and Technology, Hisar, Haryana, India.

*Corresponding author. Tel: +91 9873106374; E-mail: mukesh.gbu@gmail.com

Citation: Mann M, Tomar P, Sangwan OmP, Nature-Inspired Metaheuristic Search Strategies. Electronic J Biol, 12:2

Received: February 20, 2016; **Accepted:** March 17, 2016; **Published:** March 24, 2016

Review Article

Abstract

Research on metahurestic for solving optimization problems has been appeared as a great subject of interest during last few years. It involves modeling the natural phenomena of various species foraging for the food and as well as theory of natural evolution of species. In this paper we discuss three major metahurestic approaches for optimization problems appeared in software testing such as path prioritization, automatic test case generation, test case selection etc. First we discuss Ant colony optimization as suggested by Grasse in 1959 and later modeled by Dorigo, Maniezzo, and Colorni in 1996 as one of the optimization algorithm for solving optimization problems. Second focus is on natural inspired phenomena of Honey bee colony suggested by V. Tereshko, based on Reaction–diffusion diffusion model of a honeybee colony's foraging behavior. Finally we end up with genetic algorithm inspired by theory of evolution for solving optimization problems. The survey results the potential use of mentioned metahurestic approaches in software testing.

Keywords: Ant colony optimization (ACO); Artificial bee colony (ABC); Genetic algorithm (GA).

1. Introduction

The process of testing a software system before its actual use is essential for delivering error free software. The system is checked by comparing the expected output with the actual output after executing sample input(s) [1]. If both expected and actual outputs are same then there is no error else the system contains an error. Thus the aim of software testing is to find faults within the system by executing it with sample input data. There are number of software testing techniques such test case optimization and prioritization [2-9] which are proposed in preceding years to reduce various resources such as time, human Intervention in testing phase.

A major area of research is in Structural Testing which takes into account the code, structure of code and internal design. Commonly used techniques for structural testing [8-12] are that include control flow/ coverage testing, basic path testing, loop testing, and data flow testing [13]. Recent studies in testing indicate that a major work is in area of test case generation and prioritization based on experimental proofing. The techniques that are used in such studies are inspired from Artificial intelligence [2-5,14-16]. Therefore a part from the above mentioned techniques a new focus for the researchers is how to map natural intelligence to Artificial intelligence for solving NP hard problems, one such nature inspired intelligence called metahurestic search gives us optimized solutions for a problem in hand. In this paper we discuss various metahurestic techniques on which research work is going on very rapidly.

1.1 The ant colony: Metahurestic

Ant algorithms are one of the most attracting areas of research in optimization of software testing. The natural foraging behaviour of real ant's in wild space was first modelled by Dorigo, Maniezzo, and Colorni [8]. The indirect communication between ants within the colony using pheromone secretion provide a most powerful path for optimization methods. The ant colony algorithms are considered to be a part of Swarm intelligence i.e. multi- agent systems are based on the behavior of natural real world insects acting as swarm and collectively forming optimization behavior while foraging. A number of such swarm based algorithms has been proposed such as artificial bee colony, artificial bee colony optimization and particle swarm optimization. In this section a

details discussion on the idea behind ACO is put forwarded to solve software testing problems in much easy way as compared to the traditional approach.

1.2 Ant in the real world-a wild metaphor

In 1959 Grasse [17] first introduced a term known as Stigmergy to the indirect form of communication among multi-agents evolving as a self-organized single system while modifying their local environment. In 1990 Deneubourg, Aron, Goss, and Pasteels [18] studied the ant colonies Stigmergy nature in which each ant communicate indirectly by depositing a chemical known as pheromone on their path and the ant then tends to follow that path. Goss, Aron, Deneubourg, and Pasteels [18] with their experiment show that ants converge to the trail having higher concentration on pheromone deposit. Let us demonstrate a simple - Shortest bridge experiment – as conducted by (Goss et al.) [17]. Figure 1 shows the diagrammatic view of setup in which two possible paths from source (nest) to destination (food) are shown. Path A is shorter than path B. Initially as the ant move from the nest foraging for food, it follow a random path but as the time passes we observed that all ants following the path A. This is due to the fact that when initially an ant move from her nest following path A and after collecting food she will reach to nest again in less time than an ant which had followed the path B due to the distance

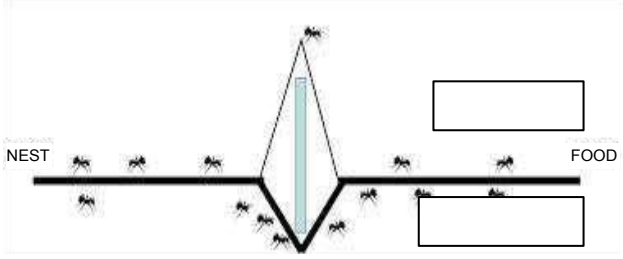

Figure 1. Two possible path for foraging.

parameter and in this way it had deposited a pheromone before the second ant following the path B. Thus a convergence to a shortest path is achieved (Figure 1). The experiment shows the exploration to exploitation due to pheromone deposition tendency of ants in real world.

1.3 From nature to artificial computers

In the following section, we briefly explain how the ants' behavior, as well as pheromone evaporation can implement.

1.4 Probabilistic forward ants and solution construction

In ACO, there are two mode of working, the first is called forward move and the other is called backward move. In forward mode the ant's movement is from nest (i.e. the source) to the food (destination) and Vice-versa in backward mode. After reaching to food source the ant reverse its direction .i.e. a shift to backward direction in order to reach to the source. The solution in forward moves is built by taking a probabilistic decision for next node to move to among those in the neighborhood of graph node on which they are located. (Given a graph G = (N,A), two nodes i, j~N if there exists an arc i, j~A). This probabilistic choice is particularly biased on pheromone deposited and Heuristic value between two nodes.

1.5 Deterministic backward ants and pheromone update

The use of an explicit memory allows an ant to repeat the path it had followed while building solutions. While moving backward, PP-ACO ants leave pheromone trail on the arcs they traverse.

1.6 Pheromone updates based on solution quality

In ACO the ants memorize all nodes that are traced through forward move, as well as the associated arc's cost if the graph is weighted. They can therefore evaluate the cost of the solutions they build and use this evaluation to modulate the amount of pheromone and heuristic they deposit while in backward mode. Making pheromone and heuristic update as function of the generated solution quality ensure in directing future ants more strongly toward better solutions.

1.7 Tour building

In Ant systems, the tour in a control flow graph (path) is concurrently builds by m number of ants. An ant start from an initial node and by using random proportional rule the ant decides the next node to be visited upon. The probability with which ant k, at node i, chooses the node j is given as

$$P_{ij}^{k} = \frac{\left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}{\sum l \in N_i^k \left[\tau_{ij}\right]^{\alpha}\left[\eta_{ij}\right]^{\beta}}, \text{ if } j \in N_i^k,$$

Where $N_{ij}=1/d_{ij}$ is called heuristic which is priori available and it indicates the visibility of a path so that an ant can follow a path having higher heuristic. N_i^k is the feasible neighbourhood of ant k at node i i.e. the set of node in CFG which an ant has not visited (P_{ij}^K for choosing a node outside N_i^k is 0). The

effect of pheromone and heuristic on next move is determined by the value of α and β. Setting α=0, will select the most closet node, if β=0 then only deposited pheromone trail influence the next move and it may leads to poor solutions. In situations when a>1, a stagnation situation occurs in which all the ants follow the same path and construct the same tour, which, results in sub-optimal solutions [8]. Thus it is extremely important to choose these parameters very carefully. Good parameter values for the various ant Algorithms are given in Table 1 [19]. In the following section, we briefly explain how the ants' behaviour, as well as pheromone evaporation can implement.

1.8 Pheromone updation: A sequential way

After an ant has constructed its tour, the updation in pheromone is completed by lowering the pheromone value on all arcs visited by the ant by a constant amount (or factor). This is also called as pheromone evaporation. The left pheromone value is then added to the nodes an ant has visited in their tour. Pheromone evaporation is given as

$t_{ij} \leftarrow (1-\rho)^{t}_{ij}$

Where 0<ρ<1, and ρ is the pheromone evaporation rate. ρ controls the indefinite addition of the pheromone thus permitting the algorithm to overlook previously taken bad decisions. The pheromone deposition after evaporation is defined as

$$t_{ij} \leftarrow t_{ij} + \Delta t^{k}_{ij}$$

Where Δt^{k}_{ij} is the pheromone deposited by k^{th} ant on the visited node, which is defined as:

$$\Delta t^{k}_{ij} \begin{cases} \frac{1}{c^{k}_{l}}; \text{arc}(i,j) \in \text{toT}^{k} \\ 0; \text{otherwise}; \end{cases}$$

Where C^{k}_{l} is the tour's length built by the k^{th} ant, which is given by the summation of arc's length which belongs to T^{k}.

Table 1. Parameter setting for various ACO algorithms.

ACO algorithm	α	β	Pheromone Evaporation (ρ)	Number of antsm	Initial Pheromone t_0
Ant systems	1	2 to 5	0.5	n	m/c^{nn}
EAS	1	2 to 5	0.5	N	$(e+m)/ \rho c^{nn}$
ASrank	1	2 to 5	0.1	N	$0.5r(r-1)/ \rho c^{nn}$
MMAS	1	2 to 5	0.02	N	$1/ \rho c^{nn}$
ACS	-	2 to 5	0.1	10	$1/nc^{nn}$

Table 2. Survey on Ant colony approach in automated software testing.

Reference	Pub. year	Topic	Author	Thrust area	Published in..
[20]	2013	A survey on optimization metahurestic	Ilhem Boussaïd, Julien Lepagnot and Patrick Siarry	Survey of some of the main metahurestic	Elsevier
[21]	2009	Adaptable Learning Pathway Generation with Ant Colony Optimization	Lung-Hsiang Wong and Chee- Kit Looi	Modeling learning pathways that combines rule-based prescriptive planning	International Forum of Educational Technology & Society (IFETS).
[22]	2005	An Ant Colony Optimization Approach to Test Sequence Generation for State based Software Testing	Huaizhong LI and C. Peng LAM	Automatic test Sequence generation for state- based software testing	IEEE
[23]	2006	Artificial Ants as a Computational Intelligence Technique	Marco Dorigo, Mauro Birattari, and Thomas St¨utzle	Computational intelligence	IEEE
[24]	2007	Classification With Ant Colony Optimization	David Martens, Manu De Backer, Raf Haesen	Classification	IEEE
[25]	2009	Automatic Test Data Generation Based On Ant Colony Optimization	Kewen Li, Zilu Zhang and Wenying Liu	Generating test data based	IEEE
[26]	2009	Building Prioritized Pairwise Interaction Test Suites with Ant Colony Optimization	Xiang Chen, Qing Gu, Xin Zhang and Daoxu Chen	Test generation algorithms	IEEE

[27]	2009	An Approach of Optimal Path Generation using Ant Colony Optimization	Praveen Ranjan Srivastava, Km Baby and G Raghurama	Optimal path identification	IEEE
[28]	2006	Automated Software Testing Using Metahurestic Technique Based on An Ant Colony Optimization	Praveen Ranjan Srivastava and Km Baby	Ant colony theory understanding and application	(IRIDIA) Technical Report series Book chapter in Approximation Algorithms and Metaheuristics
[29]	2007	Automatic Mutation Test Input Data Generation via Ant Colony	K. Ayari, S. Bouktif and G. Antoniol	Automatic test input data generation in the context of mutation testing	ACM
[30]	2005	Software Test Data Generation using Ant Colony Optimization	Huaizhong Li and C. Peng Lam	Test data generation for the state based software testing.	World Academy of Science, Engineering and Technology, 2005
[31]	2002	A Review on the Ant Colony Metaheuristics: Basis, Models and New Trends	Oscar Cordon, Francisco Herrera and Thomas Stutzle	Application of ACO to challenging combinatorial problems	Mathware and Soft Computing
[32]	2010	Test Case Prioritization using Ant Colony Optimization	Yogesh Singh, Arvinder Kaur and Bharti Suri	Regression test prioritization technique	ACM
[33]	2005	Comparison among five evolutionary-based optimization algorithms	Emad Elbeltagi, Tarek Hegazy and Donald Grierson	Comparison of five recent evolutionary-based algorithms: genetic algorithms, memetic algorithms, particle swarm, ant-colony systems, and shuffled frog leaping	Elsevier
[34]	2009	A review of ant algorithms	R.J. Mullen, D. Monekosso, S. Barman and P. Remagnino	A critical review of ACO algorithm	Elsevier
[35]	2005	Ant colony optimization theory: A survey	Marco Dorigo and Christian Blum	Discuss ACO algorithm and various open research question	Elsevier

The heuristic updation is given as

Δ nij $\leftarrow$no/C_i^k

In the Table 2 [20-35] we briefly discuss the various works that has been contributed by various researchers in field of software testing using ACO

1.9 Artificial bee colony optimization

Reaction diffusion model of a honeybee colony's foraging behavior [36]. Tereshko was the first who mapped the foraging behavior of honeybee based on reaction diffusion equations which results in collective exploration and exploitation of food source (i.e. candidate solutions) by swarm (honeybees) [37]. The model has three basic components as

- **Food Sources:** In order to select a food source, a forager bee evaluates several properties related with the food source such as its closeness to the hive, richness of the food, taste of its nectar, and the ease or difficulty of extracting this food. For the simplicity, the quality of a food source can be represented by only one quantity although it depends on various parameters.
- **Employed foragers:** An employed forager is the bee who is currently exploiting the food source i.e. hunting the food source. After collecting the nectar (food) from the food source she gives the information about the various parameters such as food quality, distance of food source from hive, direction of food source to the bees in the hives (onlooker).
- **Unemployed foragers:** These are the bees that are continuously looking for the food to exploit it.

They are either the scout bees that are searching the space near to hive randomly or the onlooker bees that are waiting for the employed bee to transfer the information about the food source and once they get information they went out of the hive to go in the direction of food. After leaving the hive they themselves act as an employed bee once they get the food source. On an average there are about 5-10 % of scout bees.

The exchange of information about the food is totally depends on the dance of employed bees in the hive, once the employed bee get the food and return to the hive , she dances on the floor called as waggle dance. The onlookers closely watch this waggle dance on the dancing floor and get employ herself at the most profitable source depending on the dance. In this way a recruitment process has been started i.e. onlookers are recruited to become employed bees.

Let us try to understand the whole process using Figure 2. We assume that there are two food sources A and B. The source A is more rich i.e. higher nectar than source B. Initially assume that two bees act as forger (searching for food) about 5-10% are scouts and the rest are onlooker bees in the hive. So these initial two bees act as a scout bees and let bee (B1) goes to food source A and the other bee (B2) goes to food source B. Once they find the food source and collect the nectar from it they become employed bee E1 and E2 for B1 and B2 respectively. After collecting the food she returns to the hive and start dancing on the dancing floor. As two bee collect different information about the food source so there dance is different. Onlooker's bees carefully watch the dance of these two bees and decide which food source they choose and in this way onlooker bees become employed bees and reached to the food source.

An Artificial Bee Colony Algorithm: A simple ABC algorithm is given in Listing 1.

ABC has been used in number of ways in different automated software testing approaches. A detail list of the survey is illustrated in the Table 3 [38-64].

From the literature survey it has been clear that ABC has potential to solve many problems in automated software testing domain. a critical insight from the survey suggest that ABC has been used to make software testing more reliable, time efficient and better resource utilization. It is thus again an open research area in automated software testing.

1.10 Genetic algorithm

Genetic Algorithm was proposed by Holland [2]. GA is heuristic search algorithms that is inspired by natural biological evolution of species and solve a variety of optimization problems to improve the quality of the search. A simple GA algorithm is shown in Listing 2.

The general flow chart for the same is shown in Figure 3. A research survey on the use of genetic algorithm in automated software testing has been conducted as shown in Table 4 [65-85].

A Literature survey study among the various metahurestic techniques used in automated software testing has been discuss above which clearly indicate frequently used metahurestic for automated software testing during the last few years. The survey also indicates the potential of using metahurestic as a search technique in software testing [86-89].

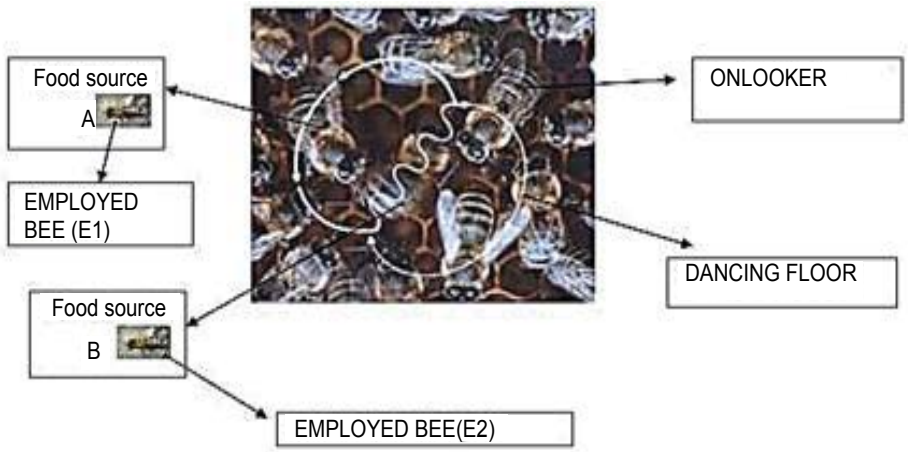

Figure 2. Honey bee wangle dance sharing information.

Table 3. Survey on bee colony approach in automated software testing.

Reference	Pub. year	Topic	Author	Published in
[38]	2012	A Recombination Based Hybridization of Particle Swarm Optimization	Mustafa Servet Kiran and Mesut Gunuz	Elsevier
[39]	2010	A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization	Bahriye Akay and Dervis Karaboga	Elsevier
[40]	2009	A New Design Method Based on Artificial Bee Colony Algorithm for Digital IIR Filters	Nurhan Karaboga	Elsevier
[33]	2005	Comparison Among Five Evolutionary-based Optimization Algorithms	Emad Elbeltagi, Tarek Hegazy and Donald Grierson	Elsevier
[41]	2007	On the Performance of Artificial Bee Colony (ABC) Algorithm	D. Karaboga and B.Basturk	Elsevier
[42]	2008	An Artificial Bee Colony Algorithm for the Leaf-Constrained Minimu Spanning Tree Problem	Alok Singh	Elsevier
[43]	2012	Block Matching Algorithm for Motion Estimation Based on Artificial Bee Colony	Erik Cuevas, Daniel Zaldivar, Marco Perez- Cisneros	Elsevier
[44]	2012	Water Cycle Algorithm- A Novel Metaheuristic Optimization Method for Solving Constrained	Hadi Eskandar, Ali Sadollah, Ardeshir Bahreininejad	Elsevier
[45]	2012	Comparative Performance Analysis of Artificial ABC Algorithm in Automatic Generation Control	Haluk Gozde, M. Cengiz Taplamacioglu and illah Kocaarslan	Elsevier
[46]	2011	Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony	Soma Sekhara Babu Lam and M L Hari Prasad Raju	Elsevier
[47]	2012	The Further Research on the Application of ABC to the Optimization and Control of Project	Cui Qiao and Hengshan Wang	Canadian Center of Science and Education
[48]	2007	Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem	Adil Baykaso? lu, Lale Özbakır and Pınar Tapkan	N/M
[49]	2010	Application of Artificial Bee Colony Algorithm to Software Testing	Surender Singh Dahiya, Jitender Kumar Chhabra and Shakti Kumar	IEEE
[50]	2011	Search-Based Software Testing: Past, Present and Future	Phil McMinn	IEEE
[51]	2012	Overview of Artificial Bee Colony (ABC) Algorithm and Its Applications	Fahad S. Abu-Mouti and Mohamed E. El-Hawary	IEEE
[52]	2012	A Discrete Artificial Bee Colony Algorithm for the Traveling Salesman Problem with Time Windows	Korhan Karabulut and M. Fatih Tasgetiren	IEEE
[53]	2011	A Survey of Combinatorial Testing	Changhai Nie Hareton Leung	ACM
[54]	2012	A Comprehensive Survey: Artificial Bee Colony (ABC) Algorithm and Aplications	Dervis Karaboga, Beyza Gorkemli and Celal Ozturk	Springer
[55]	2007	A Powerful and Efficient Algorithm for Numerical Function Optimization: (ABC) algorithm	Dervis Karaboga and Bahriye Basturk	Springer
[56]	2009	A Comparative Study of Artificial Bee Colony Algorithm	Dervis Karaboga and Bahriye Basturk	Elsevier
[57]	2011	A Survey on the Applications of Bee Colony Optimization Techniques	Dr. Arvinder Kaur and Shivangi Goyal	IJCSE
[58]	1995	Removing the Genetics from the Standard Genetic Algorithm	Shumeet Baluja and Rich Caruana	
[59]	2009	ABC Tester - Artificial Bee Colony Based Software Test Suite Optimization Approach	D. Jeya Mala and V. Mohan	IJCSE
[60]	2012	Hybrid Harmony Search and ArtificialBee Colony Algorithm for Global Optimization Problems	Bin Wu, Cunhua Qian, Weihong Ni and Shuhai Fan	Elsevier

[61]	2012	Artificial Bee Colony Programming for Symbolic Regression	Dervis Karaboga, Celal Ozturk, Nurhan Karaboga and Beyza Gorkemli	Elsevier
[62]	2011	Collaborative Artificial Bee Colony Optimization Clustering Using SPNN	D.Shanthi and R.Amalraj	Elsevier
[63]	2011	Nature Inspired Cooperative Strategies for Optimization (NICSO 2011)	David Alejandro Pelta,Natalio Krasnogor, Dan Dumitrescu	Springer
[64]	2012	Artificial Bee Colony Algorithm with Improved Explorations for Numerical Function Optimization	Mohammad Shafiul Alam, Md. Monirul Islam and Kazuyuki Murase	Springer

Figure 3: Flowchart of a simple genetic algorithm.

Table 4. Survey on genetic algorithm approach in automated software testing.

Ref No	Pub. year	Topic	Author	Published in.
[65]	2011	Diversity oriented test data generation using Metaheuristics search techniques	Paulo M.S. Bueno , Mario Jino, W. Eric Wong	Elsevier
[66]	2006	Automatic test data generation using genetic algorithm and program dependence graphs	James Miller, Marek Reformat, Howard Zhang	Elsevier
[67]	2009	Application of Genetic Algorithm in Software Testing	Praveen Ranjan Srivastava and Tai- hoon Kim	International Journal of Software Engineering and Its Applications
[68]	2010	Using Genetic Algorithms and Dominance Concepts for Generating Reduced Test Data	Ahmed S. Ghiduk Moheb R. Girgis	Informatica
[69]	2009	Eccentric Test Data Generation for Path Testing Using Genetic Algorithm	Punam Mishra, Bhabani Shankar Prasad Mishra	International Conference on Computer Engineering and Applications
[70]	2004	Search-based software test data generation: a survey	Phil McMinn	Software Testing, verification and reliability
[71]	2007	Automatic software test data generation for spanning sets coverage using genetic algorithms	Abdelaziz M. Khamis,Moheb R. Girgis,Ahmed S. Ghiduk	Computing and Informatics,
[72]	1996	Automatic structural testing using genetic algorithms	B. F. Jones, H.-H. Sthamer and D. E. Eyres	Software Engineering Journal
[73]	1997	Genetic Algorithms for Dynamic Test Data Generation	Christoph C. Michael, Gary E. McGraw, Michael A. Schatz ,Curtis C. Walton	IEEE
[74]		Automated Software Test Data Generation for Complex Programs	Christoph Michael & Gary McGraw	Not mentioned
[75]	2000	Using Genetic Algorithms for Test Case Generation in Path Testing	Jin-Cherng Lin and Pu-Lin Yeh	IEEE
[76]	2001	Generating Software Test Data by Evolution	Christoph C. Michael, Gary McGraw,Michael A. Schatz	IEEE Transactions on Software Engineering
[77]	2002	Breeding Software Test Cases with Genetic Algorithms	D. Berndt, J. Fisher, L. Johnson, J. Pinglikar, and A. Watkins	Proceedings of the 36th Hawaii International Conference on System Sciences (IEEE)
[78]	2004	Investigating the Performance of Genetic Algorithm-Based Software Test Case Generation	Donald J. Berndt and Alison Watkins	Proceedings of the Eighth IEEE International Symposium on High Assurance Systems Engineering
[79]	2003	Genetic Algorithm Based Test Data Generator	Irman Hermadi AND Moataz A. Ahmed	IEEE
[80]	2004	Using a Genetic Algorithm and Formal Concept Analysis to Generate Branch Coverage Test Data Automatically	Susan Khor and Peter Grogono	Proceedings of the 19th International Conference on Automated Software Engineering
[71]	2008	Automatic Path-oriented Test Data Generation Using a Multi-population Genetic Algorithm	Yong Chen and Yong Zhong	Fourth International Conference on Natural Computation (IEEE)
	2012	A Concept of Out Degree in CFG for Optimal Test Data Using Genetic Algorithm	Shadab Irfan and Prabhat Ranjan	1st Int'l Conf. on Recent Advances in Information Technology (IEEE)

[81]	2012	Automatic Program Instrumentation in Generation of Test Data using Genetic Algorithm for Multiple Paths Coverage	P .Maragathavalli, S.Kanmani	IEEE-International Conference On Advances In Engineering, Science And Management
[82]	2012	Genetic algorithm based software testing specifically structural testing for software reliability enhancement	NIRPAL P.B. AND KALE K.V.	International Journal of Computational Intelligence Techniques
[83]	2000	Automated test-data generation for exception conditions	N. Tracey, J. Clark, K. Mander§ and J. McDermid	SOFTWARE—PRACTICE AND EXPERIENCE
	2012	Program test data generation for branch coverage with genetic algorithm	AnkurPachauriand Gursaran	CS & IT-CSCP 2012
[84]	2005	Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm	Moheb R. Girgis	Journal of Universal Computer Science

2. Conclusion

In this paper we analyze scope of research that can be carried out in computer science especially in field of software testing using nature inspired algorithms such as ACO, ABC AND GA. From the survey we conclude that during the last few decades there has been a significant change in testing approaches. Metahurestic approaches are slowly replacing the Manual and traditional testing activities. We have a strong observation that these nature inspired algorithms are not only useful on computer science but also in other related filed such as electronics (making optimized control flow, load balancing, circuit optimization and related optimization problems) and biology such as protein folding prediction. Thus metahurestic is growing area for research and is open to carry out new researches. We have observed that researchers are moving from traditional algorithms to natural inspired algorithms in solving NP hard problems.

Acknowledgements

This Work was supported by university grant commission (UGC), Government of India for Doctoral Research Study under Grant No. F./201415/ NFO201415OBCDEL16123. The authors are very grateful to the anonymous reviewers for providing valuable and detailed comments that have greatly improved the paper.

References

[1] Myers GJ, Badgett T, Thomas TM, et al. (2004). Inc ebrary. The Art of Software Testing.

[2] Holland JH. (1975), Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, U Michigan Press.

[3] Korel B, Laski J. (1991). Algorithmic software fault localization. *System Sciences, 1991.* Proceedings of the Twenty-Fourth Annual Hawaii International Conference on, IEEE. 246-252.

[4] Xu YC, Xiao RB. (2006). Hybrid particle swarm algorithm for packing of unequal circles in a larger containing circle. Intelligent Control and Automation, WCICA 2006. The Sixth World Congress on, IEEE. 3381-3385.

[5] Zha W, Venayagamoorthy GK. (2005). Comparison of non-uniform optimal quantizer designs for speech coding with adaptive critics and particle swarm. Industry Applications Conference, 2005. Fourtieth IAS Annual Meeting. Conference Record of the 2005, IEEE. 674-679.

[6] Nenortaite J, Simutis R. (2005). Adapting particle swarm optimization to stock markets. Intelligent Systems Design and Applications, 2005. ISDA'05. Proceedings. 5th International Conference on, IEEE. 520-525.

[7] Do H,Rothermel G. (2006). On the use of mutation faults in empirical assessments of test case prioritization techniques. Softw Eng IEEE Trans. **32:** 733-752.

[8] Dorigo M., Maniezzo V, and Colorni A. (1996). Ant system: optimization by a colony of cooperating agents. Syst Man, Cybern Part B Cybern IEEE Trans. **26:** 29-41.

[9] Do H, Rothermel G., and Kinneer A. (2006). Prioritizing JUnit test cases: An empirical assessment and cost-benefits analysis. Empir Softw Eng. **11:** 33-70.

[10] Rothermel G, Harrold MJ. (1996). Analyzing regression test selection techniques. Softw Eng IEEE Trans. **22:** 529–551.

[11] Liu S, Chen Y. (2008). A relation-based method combining functional and structural testing for test case generation. J Syst Softw. **81:** 234-248.

[12] Frankl PG, Weyuker EJ. (2000). Testing software to detect and reduce risk. J Syst Softw. 53: 275-286.

[13] Do H, Rothermel G, Kinneer A. (2004). Empirical studies of test case prioritization in a JUnit testing environment. Software Reliability Engineering, 2004. ISSRE 2004. 15th International Symposium on, IEEE. 113-124.

[14] Mann M. (2014). Test case prioritization using Cuscutta search. Netw Biol. **4:**179.

[15] Rothermel G, Harrold MJ, Von Ronne J, et al. (2002). Empirical studies of test - suite reduction. Softw Testing Verif Reliab. **12:** 219-249.

[16] Kim JM, Porter A. (2002). A history- based test prioritization technique for regression testing in resource constrained environments. Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, IEEE, 119-129,

[17] Grassé PP. (1959). The reconstruction of the nest and coordination inter chez Bellicositermes natalensis etCubitermes sp. theory stigmergy: Interpretive Essay behavior of termites manufacturers. Soc insects. **6:** 41-80.

[18] Deneubourg JL, Aron S, Goss S, et al. (1990). The self-organizing exploratory pattern of the argentine ant. J Insect Behav. **3:** 59-168.

[19] Dorigo M, Birattari M, Blum C, et al. (2008). Ant Colony Optimization and Swarm Intelligence: 6th International Conference, ANTS 2008, Brussels, Belgium, Proceedings, Springer.

[20] Boussaïd I, Lepagnot J, Siarry P. (2013). A survey on optimization metaheuristics. Inf Sci (Ny). **237:** 82-117.

[21] Wong LH, Looi CK. (2009). Adaptable Learning Pathway Generation with Ant Colony Optimization. Educ Technol Soc. **12:** 309-326.

[22] Li H, Lam CP. (2005). An ant colony optimization approach to test sequence generation for state based software testing. Quality Software, 2005 (QSIC 2005), Fifth International Conference on, IEEE. 255-262.

[23] Dorigo M, Birattari M, Stutzle T. (2006). Artificial ants as a computational intelligence technique. IEEE Comput Intell Mag.**1:** 28-39.

[24] Martens D, De Backer M, Haesen R, et al. (2007). Classification with ant colony optimization. Evol Comput IEEE Trans.**11:** 651-665.

[25] Li K, Zhang Z, Liu W. (2009). Automatic test data generation based on ant colony optimization. Fifth International Conference on Natural Computation, IEEE. 216-220.

[26] Chen X, Gu Q, Zhang X, et al. (2009). Building prioritized pairwise interaction test suites with ant colony optimization. Quality Software, 2009. QSIC'09. 9th International Conference on, IEEE. 347-352.

[27] Srivastava PR, Baby KM, Raghurama G. (2009). An approach of optimal path generation using ant colony optimization. TENCON 2009-2009 IEEE Region 10 Conference, IEEE. 1-6.

[28] Srivastava PR, Baby K. (2010). Automated software testing using metahurestic technique based on an Ant Colony Optimization. Electronic System Design (ISED), 2010 International Symposium on, IEEE. 235-240.

[29] Ayari K, Bouktif S, Antoniol G. (2007). Automatic mutation test input data generation via ant colony. Proceedings of the 9th annual conference on Genetic and evolutionary computation, ACM. 1074-1081.

[30] Li H, Lam CP. (2004). Software Test Data Generation using Ant Colony Optimization. International Conference on Computational Intelligence. 1-4.

[31] García OC, Triguero FH, Stützle T. (2002). A review on the ant colony optimization metaheuristic: Basis, models and new trends. Mathw Soft Comput. **9:** 141-175.

[32] Singh Y, Kaur A, Suri B. (2010). Test case prioritization using ant colony optimization. ACM SIGSOFT Softw Eng Notes. **35:** 1-7.

[33] Elbeltagi E, Hegazy T, Grierson D. (2005). Comparison among five evolutionary-based optimization algorithms. Adv Eng informatics. 19: 43-53.

[34] Mullen RJ, Monekosso D, Barman S, et al. (2009). A review of ant algorithms. Expert Syst Appl, **36:** 9608-9617.

[35] Dorigo M, Blum C. (2005). Ant colony optimization theory: A survey. Theor Comput Sci, **344:** 243-278.

[36] Tereshko V. (2000). Reaction-diffusion model of a honeybee colony's foraging behaviour. Parallel Problem Solving from Nature PPSN VI, Springer. 807-816.

[37] Tereshko V. and Loengarov A. (2005). Collective decision making in honey-bee foraging dynamics. Comput Inf Syst, **9:** 1.

[38] Kıran MS, Gündüz M. (2013). A recombination-based hybridization of particle swarm optimization and artificial bee colony algorithm for continuous optimization problems. Appl Soft Comput, **13:** 2188-2203.

[39] Akay B, Karaboga D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci (Ny). **192:** 120-142.

[40] Karaboga N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. J Franklin Inst, **346:** 328-348.

[41] Karaboga D. and Basturk B. (2008). On the performance of artificial bee colony (ABC) algorithm. Appl Soft Comput. **8:** 687-697.

[42] Singh A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. Appl Soft Comput. **9:** 625-631.

[43] Cuevas E, Zaldívar D, Pérez-Cisneros M, et al. (2013). Block matching algorithm for motion estimation based on Artificial Bee Colony (ABC). Appl Soft Comput. **13:** 3047-3059.

[44] Eskandar H, Sadollah A, Bahreininejad A, et al. (2012). Water cycle algorithm-A novel metaheuristic optimization method for solving constrained engineering optimization problems. Comput Struct. **110:** 151-166.

[45] Gozde H, Taplamacioglu MC, Kocaarslan I. (2012). Comparative perfor, IEEE. 153-163.

[46] Abu-Mouti FS, El-Hawary ME. (2012). Overview of Artificial Bee Colony (ABC) algorithm and its applications. Systems Conference (SysCon), 2012 IEEE International, IEEE. 1-6.

[47] Karabulut K, Tasgetiren MF. (2012). A discrete artificial bee colony algorithm for the traveling salesman problem with time windows. Evolutionary Computation (CEC), 2012 IEEE Congress on, IEEE. 1-7.

[48] Nie C, Leung H. (2011). A survey of combinatorial testing. ACM Comput Surv. 43: 11.

[49] Karaboga D, Gorkemli B, Ozturk C, et al. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev. **42:** 21-57.

[50] Karaboga D, Basturk B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim. **39:** 459-471.

[51] Karaboga D, Akay B. (2009). A comparative study of artificial bee colony algorithm. Appl Math Comput. **214:** 108-132.

[52] Kaur A, Goyal S. (2011). A survey on the applications of bee colony optimization techniques. Int J Comput Sci Eng. **3:** 3037.

[53] Baluja S, Caruana R. (1995). Removing the genetics from the standard genetic algorithm. Machine Learning: Proceedings of the Twelfth International Conference. 38-46.

[54] Mala D.J. and Mohan V. (2009). ABC Tester-Artificial bee colony based software test suite optimization approach. Int J Softw Eng. **2:** 15-43.

[55] Wu B, Qian C, Ni W, et al. (2012). Hybrid harmony search and artificial bee colony algorithm for global optimization problems. Comput Math with Appl. **64:** 2621-2634.

[56] Karaboga D, Ozturk C, Karaboga N, et al. (2012). Artificial bee colony programming for symbolic regression. Inf Sci (Ny). **209:** 1-15.

[57] Shanthi D, Amalraj R. (2012). Collaborative artificial bee colony optimization clustering using SPNN. Procedia Eng. **30:** 989-996.

[58] Pelta DA, Krasnogor N, Dumitrescu D, et al. (2011). Nature Inspired Cooperative Strategies for Optimization (NICSO 2011). Springer Science and Business Media.

[59] Alam MS, Islam MM, Murase K. (2012). Artificial bee colony algorithm with improved explorations for numerical function optimization. Intelligent Data Engineering and Automated Learning-IDEAL 2012, Springer. 1-8.

[60] Bueno PMS, Jino M, Wong WE. (2014). Diversity oriented test data generation using metaheuristic search techniques. Inf Sci (Ny). **259:** 490-509.

[61] Miller J, Reformat M, Zhang H. (2006). Automatic test data generation using genetic algorithm and program dependence graphs. Inf Softw Technol. **48:** 586-605.

[62] Srivastava PR, Kim T. (2009). Application of genetic algorithm in software testing. Int J Softw Eng its Appl. **3:** 87-96.

[63] Ghiduk AS, Girgis MR. (2010). Using genetic algorithms and dominance concepts for generating reduced test data. Informatica. 34.

[64] Mishra P, Mishra BSP. (2009). Eccentric test data generation for path testing using genetic algorithm. Int Proc Comput Sci Inf Technol. **2:** 536.

[65] McMinn P. (2004). Search-based software test data generation: A survey. Softw Test Verif Reliab. **14:** 105-156.

[66] Chen Y, Zhong Y. (2008). Automatic path-oriented test data generation using a multi-population genetic algorithm. Natural Computation, 2008. ICNC'08. Fourt International Conference on, IEEE. 566-570.

[67] Jones BF, Sthamer HH, Eyres DE. (1996). Automatic structural testing using genetic algorithms. Softw Eng J. **11:** 299-306.

[68] Michael CC, McGraw GE, Schatz MA, et al. (1997). Genetic algorithms for dynamic test data generation. Automated Software Engineering, 1997. Proceedings., 12th IEEE International Conference, IEEE. 307-308.

[69] Michael C, McGraw G. (1998). Automated software test data generation for complex programs. Automated Software Engineering, 1998. Proceedings. 13th IEEE International Conference on, IEEE. 136-146.

[70] Lin JC, Yeh PL. (2000). Using genetic algorithms for test case generation in path testing. Test Symposium, 2000. (ATS 2000). Proceedings of the Ninth Asian, IEEE. 241-246.

[71] Michael CC, McGraw G, Schatz MA. (2001). Generating software test data by evolution. Softw Eng IEEE Trans. **27:** 1085-1110.

[72] Berndt D, Fisher J, Johnson L, et al. (2003). Breeding software test cases with genetic algorithms. System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on, IEEE. 10-pp.

[73] Berndt DJ, Watkins A. (2004). Investigating the performance of genetic algorithm-based software test case generation. High Assurance Systems Engineering, 2004. Proceedings. Eighth IEEE International Symposium on, IEEE. 261-262.

[74] Hermadi I, Ahmed MA. (2003). Genetic algorithm based test data generator. Evolutionary Computation, 2003. CEC'03. The 2003 Congress on, IEEE. 85-91.

[75] Khor S, Grogono P. (2004). Using a genetic algorithm and formal concept analysis to generate branch coverage test data automatically. Automated Software Engineering, 2004. Proceedings. 19th International Conference on, IEEE. 346-349.

[76] Maragathavalli P, Kanmani S, Kirubakar JS, et al. (2012). Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths coverage. Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on, IEEE. 349-353.

[77] Nirpal PB, Kale KV (2012). Genetic Algorithm Based Software Testing Specifically Structural Testing for Software Reliability Enhancement. Int J Comput Intell Tech ISSN. 466-976.

[78] Tracey N, Clark J, Mander K, et al. (2000). Automated test-data generation for exception conditions. Software-Practice Exp. **30:** 61-79.

[79] Girgis MR. (2005). Automatic test datageneration for data flow testing using a genetic algorithm. JUCS. **11:** 898-915.

[80] Mohapatra D. (2011). GA Based Test Case Generation Approach for Formation of Efficient Set of Dynamic Slices. Int J Comput Sci Eng. 3.

[81] Sharma C, Sabharwal S, Sibal R. (2014). Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams. arXiv Prepr arXiv14104838.

[82] Maragathavalli P, Anusha M, Geethamalini P, et al. (2011). Automatic test-data generation for modified condition/decision coverage using genetic algorithm.

[83] Singhal A, Chandna S, Bansal A. (2012). Optimization of Test Cases Using Genetic Algorithm 1.

[84] Mahajan M, Kumar S, Porwal R. (2012). Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach. ACM SIGSOFT Softw Eng Notes. 37.