International Journal of Innovation in Computer Science and Information Technology
Vol.3, NO.2 , P: 21 - 30
Received : 3 June 2021
Accepted : 29 June 2021

Presenting Comprehensive Comparison between the New Intelligence Software Testing Techniques

Emad Efatinasab[*]
Student at department of electrical and computer engineering, University of birjand,birjand,IRAN
emad.effati@gmail.com
Abolfazl Ajami
student at department of industrial engineering and systems, Tarbiat modares university,Tehran,IRAN
a.ajami@modares.ac.ir

Abstract
in this article, we will take a deeper look at software testing, its methods and, applications. If we want to have a simple definition of software testing, we can say that "software experimenting is the development of running an application with the guidance of finding bugs and subsequently improving its quality."[1] Software testing is a critical process that plays a role in ensuring the quality of software systems. Experimenting is currently considered an industry in the field of software. Software testing success is always determined based on generated test cases and their prioritization [2]. Therefore, it consumes more effort, time, and, cost. Today, a considerable number of soft computing-based approaches are available for better exactness in testing. This paper aims to provide a review of some work that has been done in the software testing area by using soft computing techniques. This paper will be suggesting an approach for data flow testing using PSO and ACO. This paper presents how PSO and ACO algorithm is used for optimizing the issue of data flow testing.

Keywords Software testing technique, Evolutionary Algorithm, Soft Computing

Introduction
Testing mainly comprises of static testing and dynamic testing. The experiment, which does not require any tool, is called Static testing and, it is done without executing the program. Dynamic testing is performed with the execution of code, and it is a part of validation. In this testing, test data is given to the system in the form of input, and results are checked against the expected output while executing the software as the structure of logic is not considered in this testing. [3] On the other hand, white box testing is another essential technique in dynamic testing. It is also known as structural testing as whole structure design, and code is tested, and it aims to test the internal parts and uncover bugs as many as possible in the logic of the program.
Software amplification is the process of imagining, distinguish, proposing, Coding, documenting, Experimenting, and mending bugs in building and keeping applications, frames, or other software parts.
Software amplification is the movement of writing and keeping an original code, still, on a broader concept, it covers everything from the basic idea of the desired software to the Terminal emergence of the software, sometimes in a trace and structured movement. Therefore, software amplification can contain prowl, new process, prototyping, reconciliation, replication, reengineering, maintenance, or any activity that leads to software crop.
Software can be extended for a diversity of object, the three most Usual of which are meeting specific needs for a particular customer / business (specific software),

meeting a diagnosed need for a set of potential users (source software Open and commercial), or for Private Utilization (for example, a scholar can write software to Robotize a typical job). Embedded software development, that is, the development of embedded software, which is used, for example, to control consumer products, requires the integration of the software Extension process with the extension of Controlled physical produce. Software system emphasizes usage and the programming Movement itself and, it is often extended singly. [4]

Today, for a large project, there are large numbers of test cases required. So it becomes difficult for the tester to test large and complex programs. Therefore, there is need to reduce the testing set to generate optimal test data, which further reduces the time and cost involved in testing. This paper focuses on the soft computing techniques, which are guided by data method correlation in the scheme to quest for assay data to fulfill the data method choice criterion. [5] This paper presents an algorithm of PSO and ACO, the soft computing techniques to produce test data, which gives a healthy face of software Shield. This paper will be comparing the path coverages covered by PSO and ACO, which are used for data flow testing.

The border of the Letter is organized as follows. Part2 gives some basilar meaning and definitions, and a survey of various research papers related to dynamic testing. Section 3 describes the data-flow analysis technique. Section 4 shows a PSO and ACO algorithm that is used for optimizing data-flow testing in the proposed approach. Section 5 will show the result of path coverage done by ACO and PSO. Part 6 introduces the conclusion and future work. [6].

LITERATURE REVIEW

Here, this paper discusses some basilar meanings that are used throughout this work.

A. *Control Progress Diagram*

A control progress diagram is a control flow diagram that describes a commercial process, or summary. The control flow diagram was introduced in the 1950s ,and has been widely used in several disciplinary engineering. [8] It is one of the classic business processing methodologies, parallel to flowcharts, data flow diagrams, functional flow block diagrams, Gantt charts, Pert diagrams and IDEF.

B. Dominance Tree

In graph theory, the graph tree is connected, without distance. Trees are widely used in computer science and data structure. Such as binary search trees, stacks [9], Hoffmann trees [10] for information compression, and so on. Figure. 2, gives the dominator tree of program 1.A graph without a circle is called a tree. [11]

Enter Program:

```
int main ()
{
int a, b, c;
a=5;
b=6;
c=a+b;
printf (a, b, c);
return 0;
}
```

Fig.1.Program 1[12]

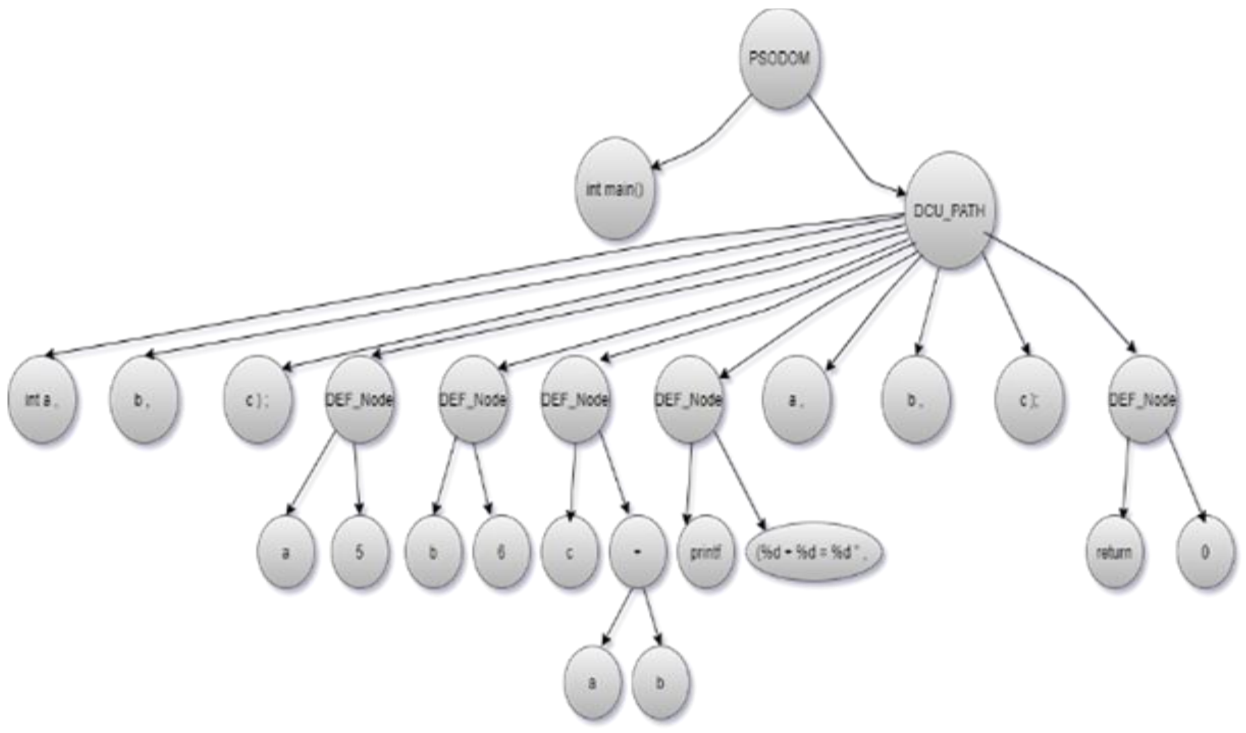


Fig. 2. Dominance Tree [13]

The definition of a dominant node was first proposed by Reiss Prosser in an article entitled Applications of the Boolean Matrix for Flow Chart Analysis [14]. In this article, he did not provide an algorithm for finding dominating nodes and only defined it. The first algorithm for this problem was proposed ten years later by Edward Lowry and Medlock [15]. Its applications include program optimization, code generation, and circuit testing. Compilers also make extensive use of information from node dominance. For example, one application in the compiler is to find loops in optimizing code with the help of base blocks [16].In directional graphs, we say that node w overcomes node n .For every node n all paths from n to v pass- through node w. As a result, it can be said that each node overcomes itself. According to the definition, each node can be considered a set of dominant nodes that overcome this node. It is also possible to define a dominant tree for each graph, each node in the graph, the ancestors of this node in the tree are the nodes that dominate this node in the graph. [17].

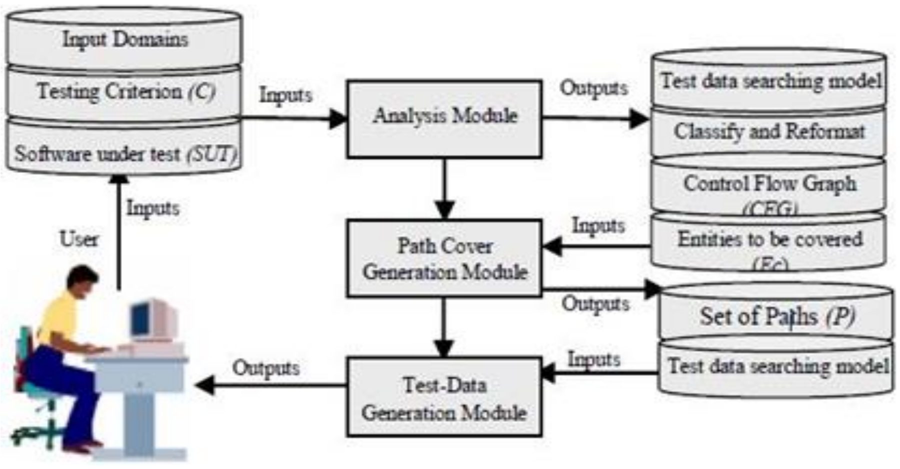


Fig.3. Block Diagram of Data Flow Testing Technique [18]

Node v overcomes node w when node w and v is vv.

Instant Dominant Nodes V-node is a set of nodes that strictly overcomes only node v and does not precisely overcome any other node [19].

Node-Dominant Nodes, V-nodes are a set of nodes whose node V overcomes their direct parents but does not strictly overcome those nodes. [20]

DATA FLOW INVESTIGATION METHOD

In system definitions, organizations can be defined as a system that uses human resources and materials to work for a particular purpose [21] These organizations can be divided into smaller systems (departments, departments and groups). Each of these smaller systems works for a specific purpose, such as accounting, sales, production, information processing, and management [22] all systems receive a set of inputs and convert them to outputs after processing. Of course, an ideal system is one that can reorganize itself without the need for human decision-making. For example, a clothing manufacturer in Italy uses such a system to sell its products. The company produces all its T-shirts in white and keeps them in stock[24] Using an intelligent information system, the trend of demand for goods in the market is analyzed, in the last stage of sending goods to the market, the coatings become the colors that have received the most demand from customers. It should be noted that the subsystems of a more extensive system are interconnected and, in many cases, interdependent. The figure below shows this dependence. In this system, the outlet of the production subsystem is used as the input of the marketing system and the outlet of the same marketing system is used as the input of the production system. A system or subsystem in an organization can be represented graphically by a set of methods. These methods want to identify system boundaries and show the information flowing in a system. [33] Data Flow Diagram is one of these methods. Data Flow Chart (DFD) focuses on the inputs and outputs of information to a system as well as the processes performed on them. This diagram is made using four main components: a square with a shadow, an arrow, a rectangle with rounded corners, and a rectangle with an open head. These shapes are shown in the figure below. Shaded squares are used to display external inputs and data (another department, an individual, or a machine) that can send or receive data to or from the system. Each data must be named by name. To prevent data flow lines from being interrupted, data can be repeated several times in a chart. [35]

Arrows indicate the transfer of information from one point to another. Because information is about a person, place, or thing, the flow of information must be described by a name. Rectangles with rounded corners are used to represent information processing. Information processing causes changes or transformations in the input information, so the information output of a process must be named with a new title.

The last functional element in DFD diagrams is a rectangle with an open head that is used to display the data storage center. Like other DFD elements, this element must be named by a specific name. Data centers are numbered by a particular, coder such as D1, D2, D3, etc. In drawing DFD diagrams, it should be noted that each process must have at least one input and one output. Another point is that an external input cannot be connected directly to a data storage (file) center, and for this, the data must be transferred through a process. Each DFD model should not have more than 9 processes. If a model has more than nine functions, the same function can be put in pro and transferred to a lower level as a subsystem. The following figure shows an example of a DFD model with real names.

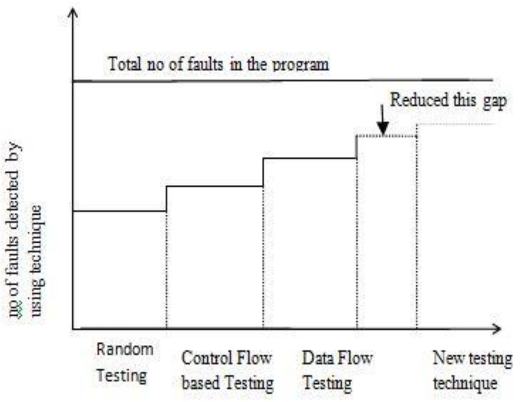


Fig.4.Comparisonof Testing Techniques

SUGGESTED METHOD

Ace heap Optimization (PSO) is one of the most severe Sagacious optimization algorithms in the field of heap sense. The algorithm was present by James Kennedy and Russell C. Eberhart in 1995 and is inspired by the Civic behavior of animals such as fish and birds that live together in Little and Enormous groups [36] In the PSO algorithm, the organ of the answer habitancy communicate directly with each other and solve the problem by exchanging information with each other and recalling good memories of the past. The PSO algorithm is appropriate for a diversity of continuous and discrete problems and gives excellent, answers to various optimization difficulties [37]

PSO can perform much better in achieving more def-use coverage as compared to other existing search-based optimization techniques like ACO, GA, etc. as it has the advantage of memory so it can keep information of reasonable solutions of all particles. The algorithm of PSO consists of these steps given below: [38]

```
1.  Initiate Swarm
2.  Repeat
3.  For p=1 to number of Ace do
4.  Appraise (p)
5.  Update Experiment (P)
6.  Update proximity best (p, k)
7.  For d=1 to number of dimensions' do
8.  Stroke (p, d)
9.  End for
10. End for
11. Until criterion
```

The consecutive PSO exemplar uses a real-valued multi-dimensional region as opinion region, and subsume the situation of each ace in that region using the following equations:

$$v_{id}^{t+1} = w.v_{id}^{t} + c_{1.}\varphi_{1}.(p_{id}^{t} - x_{id}^{t}) + c_{2.\varphi 2.}.(p_{gd}^{t} - x_{id}^{t})$$

$$x_{id}^{t+1} = x_{id}^{t} + v_{id}^{t+1}$$

Equation 1

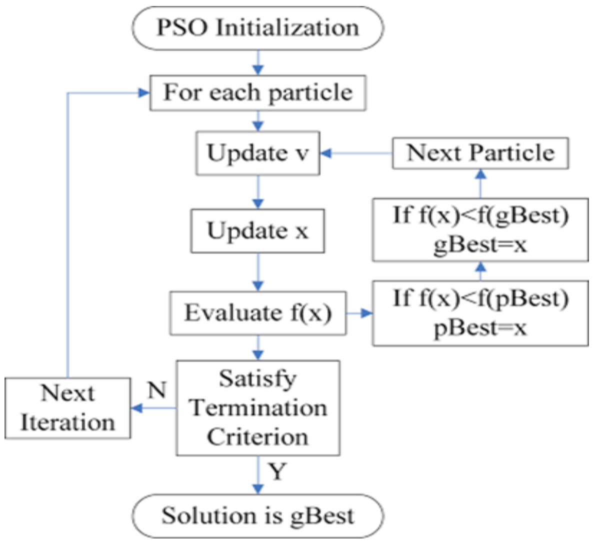


Fig.5. Flow Graph [39]

The ant colony algorithm, or "Ant Colony Optimization" as its name implies, is based on the natural behavior of the ant colonies and the worker ants working in them. The process of finding food sources in an ant colony is very efficient. When ants start exploring for food sources, they naturally find a "reasonable" and "optimal" path from their nest to food sources. [40] In other words, the ant population is somehow always able to find an optimal path to provide the required food. Simulation of such optimal behavior forms the basis of ant colony optimization. In this article, the ant colony algorithm is fully described. It should be noted that the exact name of this algorithm is ant colony optimization, which is often called the ant colony algorithm. Suppose two ants are moving from the nest to the food source through two completely different paths. As the ants move toward the food source, they emit a trace of "pheromone" into the environment, which disintegrates naturally over time. The ant that (randomly) selects the shortest path to the food source begins the return journey to the nest earlier than the other ants. In this case, on the way back to the nest, the ant starts to release the pheromone back into the environment, thereby reinforcing the pheromone trail left in the shortest path.

1. Build the searching model
2. Chose one Complainant-clear way from the way shield and mark it.
3. Put ants at the start nook of the quest model.
4. Ant moves and enters the number of nook.
5. If (ant does not get to the end node) go to step four.
6. Entry the way
7. Produce the corresponding data.
8. Perform the plan under test using the produced data and entry the execution way.
9. Compute the similarity between the execution way and the complainant-clear way.
10. Update pheromone.
11. If (execution path does not shield the complainant's clear path) go to step 3.
12. Entrer the test data.
13. If (there is an unmarked Complainant-clear way in the way cover) go to step 2
14. Outlet the set of test data and the set of covered complainant clear way.
15. End

The algorithm will automatically stop if there are no unmarked complainant-clear way. In the way shield.

Model of original ant density:

$$\Delta\tau_{ij}^{k}(t,t+1) = \int_{0}^{Q} \begin{matrix} if\ ant\ passes\ ij \\ otherwise \end{matrix}$$

Equation 2

In the initiatory afflux sample for any ant k, Q is a constant; that is, the pheromone is incremented by fix value. The number of common nodes between the executed path and the complainant clear path of the current complainant -use pair is defined as Q. [41]
The updating pheromone formula is

$$\tau_{ij}(t+n) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}$$

Equation 3

Other ants instinctively follow the strongest pheromone pathway in the environment and reinforce the pheromone trail in that pathway. After a certain period, not only does the rejection of the pheromone in the shortest path not disintegrate, but, as the rejection of the pheromone of other ants accumulates, it becomes more and more amplified. The path in which the strongest pheromone trace is left becomes the default path for the ants to move from the colony to the food original and vice versa.
The ant colony optimization method provides a model for implementing optimization methods. So far, various successful implementations of this optimization method have been proposed. Algorithms such as "Ant System", "Ant Colony System" and "Min-Max Ant System" are among the most important and successful implementations of this optimization method.

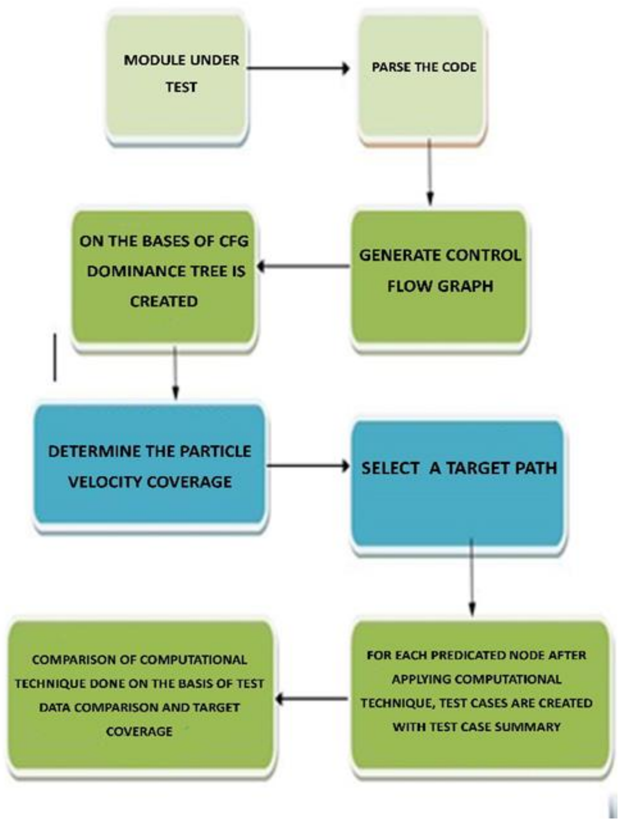
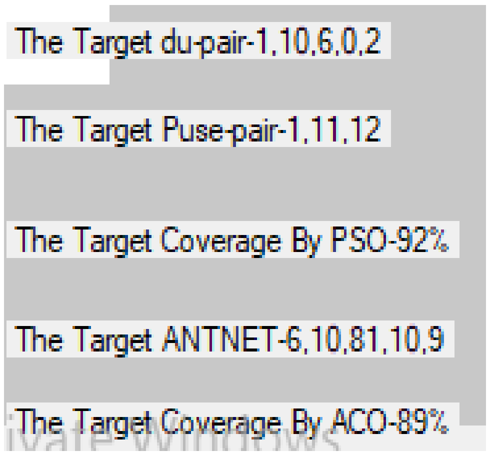

MODULE UNDER TEST
PARSE THE CODE
GENERATE CONTROL FLOW GRAPH
ON THE BASES OF CFG DOMINANCE TREE IS CREATED
DETERMINE THE PARTICLE VELOCITY COVERAGE
SELECT A TARGET PATH
FOR EACH PREDICATED NODE AFTER APPLYING COMPUTATIONAL TECHNIQUE, TEST CASES ARE CREATED WITH TEST CASE SUMMARY
COMPARISON OF COMPUTATIONAL TECHNIQUE DONE ON THE BASIS OF TEST DATA COMPARISON AND TARGET COVERAGE

Fig.6. process for data flow testing

Algorithms derived from the ant colony algorithm are a subset of Swarm Intelligence methods. This methodology is a field of research and study that studies algorithms inspired by the concept of "swarm behaviors" (Swarm Behaviors). Congestion intelligence algorithms consist of a set of simple individual entities that interact and collaborate with each other through self-organization. Self-organization means the lack of a central control system to control and coordinate the members of a crowded intelligence system.



The Target du-pair-1,10,6,0,2

The Target Puse-pair-1,11,12

The Target Coverage By PSO-92%

The Target ANTNET-6,10,81,10,9

The Target Coverage By ACO-89%

Fig.7. path coverage of PSO and ACO

CONCLUSION AND FUTURE WORK

This paper has reviewed various research papers based on dynamic testing and has found that most of the works concentrate on the coverage, but none of them told about which technique is better suited for full coverage. Evolutionary structural testing is a verge used to produce. test cases that use GA, ACO, or other search-based optimization, which is guided by data flow Affinity in the program to cover the complainant use association. Since cost and coverage are two essential factors in case of testing. This paper has proposed a process for data flow testing using PSO and ACO, and it has been observed that in comparison with PSO (particle swarm optimization) and ACO (Ant Colony Optimization) PSO is giving a better path coverage than ACO. In Future work, the PSO and ACO algorithms will be implemented in performing data flow testing to provide an efficient path with maximum code coverage and minimum cost. Furthermore, the results can be compared with other meta-heuristic techniques such as GA (Genetic algorithms) and BCO (Bee Colony Optimization), etc. [42]

References

[1] Ahmed S. Ghiduk (2010) “A New Software Data-Flow Testing Approach via Ant Colony Algorithms” Universal Journal of Computer Science and Engineering Technology 1 (1), ISSN: 2219-2158, pp. 64-72.

[2] Praveen Ranjan Srivastava, KmBaby (1977) “Automated Software Testing Using Metahurestic Technique Based on An Ant Colony Optimization”, IEEE Transactions on Software Engineering, vol. 3, no. 4, pp. 266-278.

[3] D.Jeya Mala, V.Mohan (2007) “ABC Tester - Artificial Bee Colony Based Software Test Suite Optimization Approach.”, in theProc. of 7th International Conference on Hybrid Intelligent Systems (HIS‘07), pp. 84-89. IEEE Press.

[4] K. Li, Z. Zhang and W. Liu (2009) “Automatic Test Data Generation Based On Ant Colony Optimization” in theProc. of Fifth International Conference on Natural Computation 2009, pp. 216-219. IEEE Press.

[5] P. R. Srivastava et al. (2009) “An Approach of Optimal Path Generation using Ant Colony Optimization” in the Proc. of TENCON 2009, pp.1-6. IEEE Press.

[6] M. Dorigo and C. Blum (2005) “Ant colony optimization theory: A survey” Theoretical Computer Science, 344(2-3), pp. 243-278.

[7] Sanjay Singla, H. M. Rai and Priti Singla (2011) “Automatic Test Data Generation Approach using Combination of GA and PSO with Dominance Concepts”International Journal of Electronics Engineering, 3 (1), 2011, pp. 95– 98.

[8] B. Adil et al. (2007) “Artificial Bee Colony Algorithm and Its Application to Generalized Assignment Problem Swarm Intelligence: Focus on Ant and Particle Swarm Optimization “in the proceedings of International Conference of Computational Intelligence, pp. 532–564.

[9] Mark Harman and Afshin Mansour (2010) “Search-Based Software Engineering: Introduction to the Special Issue of the IEEE Transactions on Software Engineering” IEEE Transaction on Software Engineering Vol. 36, NO. 6, pp 479-486

[10] H. Li and C. Peng LAM, (2005) “An Ant Colony Optimization Approach to Test Sequence Generation for State-based Software Testing,”in theProceedings of the Fifth International Conference on Quality Software (QSIC‘05), pp 255 – 264

[11] S. Rapps and E.J. Weyuker, (1985) “Selecting software test data using data flow information” IEEE Transactions on Software Engineering, vol.11, no. 4, pp. 367-375

[12] M.R. Girgis and M.R. Woodward, (1985) “An integrated system for program testing using weak mutation and data flow analysis,” in the Proceedings of Eighth International Conference on Software Engineering, IEEE Computer Society, pp. 313-319

[13] K. Li, Z. Zhang and J. Kou, (2010) “Breading software test data with Genetic-Particle swarm mixed Algorithms”, Journal of Computers, Vol. 5, No. 2, pp. 258-265.

[14] M.Darbandi; “Proposing New Intelligence Algorithm for Suggesting Better Services to Cloud Users based on Kalman Filtering”; Published by Journal of Computer Sciences and Applications (ISSN: 2328-7268), Vol. 5, Issue 1, 2017; PP. 11-16; DOI: 10.12691/JCSA-5-1-2; USA

[15] S.Singhla, M. Pezz`e, and M. Vivanti,(2013) “Quantifying the complexity of data flow testing,” in IEEE International Workshop on Automation of Software Test (AST), pp. 132–138

[16] A. J. Offutt, J. Pan, K. Tewary, and T. Zhang,(1996) “An experimental evaluation of data flow and mutation testing,” Softw.. Pract. Exper., vol. 26, pp. 165–176.

[17] E. J. Weyuker, (1990) “The cost of data flow testing: An empirical study,” IEEE Transactions on Software Engineering (TSE), vol. 16, no. 2, pp. 121–128.

[18] P. G. Frankl and E. J. Weyuker,(1988) “An applicable family of data flow testing criteria.,” IEEE Trans. Software Eng., vol. 14, no. 10, pp. 1483–1498.

[19] P.Mathiyalagan, (2010) “Grid scheduling Using Enhanced PSO Algorithm” International Journal on Computer Science and Engineering, Vol. 02, No. 02, pp.140-145.

[20] M.Darbandi; “Proposing New Intelligent System for Suggesting Better Service Providers in Cloud Computing based on Kalman Filtering”; Published by HCTL International Journal of Technology Innovations and Research, (ISSN: 2321-1814), Vol. 24, Issue 1, PP. 1-9, Mar. 2017, DOI: 10.5281/Zenodo.1034475.

[21] JanviBadlaney,F.E.Allen, and J. Cocke, (1976) “A program data flow analysis procedure,”Communication of the ACM, 19 (3), pp. 137-147.

[22] X. Zhang, H. Meng, and L. Jiao, (2005) “Intelligent particle swarm optimization in multi-objective optimization”, in theProc. of the 2005 IEEE Congress on Evolutionary Computation, Vo1, pp. 714-719. IEEE Press.

[23] A. Bouchachia, (2007) “An immune genetic algorithm for software test data generation” in the Proc. of 7th International Conference on Hybrid Intelligent Systems (HIS‘07), pp. 84-89. IEEE Press.

[24] Alok Singh, (2009) “An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem”, Applied Soft Computing, Volume 9, Issue 2, pp. 625-631.

[25] K. Li, Z. Zhang and J. Kou (2010) “Breading software test data with Genetic-Particle swarm mixed Algorithms”, Journal of Computers, Vol. 5, No. 2, pp. 258-265.

[26] P. McMinn, (2004) “Search-based software test data generation: A Survey,” Software Testing, Verification and Reliability, vol. 14, no. 2, pp. 105–156.

[27] V. Gazi, (2007) "Asynchronous Particle Swarm Optimization," in Signal Processing and Communications Applications, SIU 2007. IEEE 15th, 2007, pp. 1-4.

[28] A. Windisch, S. Wappler, and J. Wegener, (2007) “Applying particle swarm optimization to software testing”, ACM, GECCO, London, England, United Kingdom, New York, pp. 1121-1128.

[29] S. Rapps and E. J. Weyuker, (1985) “Selecting software test data using data-flow information,” IEEE Transactions on Software Engineering (TSE), vol. 11, pp. 367–375.

[30] Huaizhong Li and C.Peng Lam,(2004) “Software Test Data Generation using Ant Colony Optimization”, Transactions on Engineering, Computing and Technology, ISSN 1305-5313, pp. 105- 156.

[31] Chen Yonggang, Yang Fengjie, Sun Jigui. (2006). “A new Particle swam Optimization Algorithm.” Journal of Jilin University, 24(2), pp. 181-183.

[32] Praveen Ranjan Srivastava, (2009) “Optimization of software testing using Genetic Algorithm”, International Journal of Artificial Intelligence and Soft Computing, Interscience publisher, Volume 1, Numbers 2-3, pp. 363-375.

[33] Marco Dorigoa and Thomas Stutzle, (2005) “Ant colony optimization, The Knowledge Engineering Review", Cambridge University Press New York, NY, USA., Volume 20, Pp: 92 – 93.

[34] Praveen Ranjan Srivastava, Baby, (2010) “Evolutionary Computation and Optimization Algorithms in Software Engineering: Applications and Techniques”, IGI Global USA, pp 161-183.

[35] Sumesh Agarwal, Shubham Gupta, and Nitish Sabharwal, (2016) “Automatic Test Data Generation-Achieving Optimality Using Ant-Behaviour” International Journal of Information and Education Technology, Vol. 6, No. 2, pp 117-121.

[36] H. Zhu, P. Hall, and J. May,(1997) “Software unit test coverage and adequacy,” ACM Computing Surveys, vol. 29, no. 4, pp. 366–427.\

[37] A. S. Andreou, K. A. Economides, and A. A. Sofokleous, (2007) “An automatic software test-data generation scheme based on data flow criteria and genetic algorithms,”7th IEEE International Conference on Computer and Information Technology, pp. 867-872.

[38] F. E. Allen and J. Cocke, (1976) “A program data flow analysis procedure”, Communication of the ACM, Vol. 19, No. 3, pp. 137–147.

[39] J. Kennedy and R. Eberhart, (1995) “Particle swarm optimization”, IEEE International Conference on Neural Networks, IEEE Press, pp. 1942–1948.

[40] K. O. Jones, (2005) “Comparison of genetic algorithm and particle swarm optimization”, in the Proceedings of the International Conference on Computer Systems and Technologies.

[41] S. Yuhui and R. C. Eberhart, (1998) “Parameter selection in particle swarm optimization,” in the Proceedings of the 7th International Conference on Evolutionary Programming, Vol. 1447, pp. 591- 600.

[42] Z. W. Liang, W. H. Sen, Z. Jun and X. D. Jian, (2008) “Novel particle swarm optimization with heuristic mutation,”Computer Engineering and Design, pp. 3402-3405.

[43] S. Haghgoo, M. Hajiali, A. Khabir, “Prediction and Estimation of Next Demands of Cloud Users based on their Comments in CRM and Previous usages”, International IEEE Conference on Communication, Computing & Internet of Things; Feb. 2018, Chennai.DOI:10.1109/IC3IoT.2018.8668119 .