International Journal of Advanced Trends in Computer Applications
www.ijatca.com

A Study of Various Metaheuristic Techniques used for Software Testing

Jyotsna Agnihotri[1], Vijay Kumar[2]
[1]Jyotsna Agnihotri
Department of Computer Science
Chandigarh University
Gharuan, India
jyotsnaagni@gmail.com

[2]Vijay Kumar
Department of Computer Science
Chandigarh University
Gharuan, India
Vijaykumar_nrw@gmail.com

Abstract: *In today's competitive world every software company wants to deliver high quality software. So software testing is essential task as it will locate errors and ensure error free software. Basically software testing is a process of validating software with requirements and testing for bugs however it is a labor intensive and very costly task. So automation of testing is needed as exhaustive testing is not possible. A properly generated test suite has a strong impact on the efficiency and effectiveness of software testing. In recent years, metaheuristic techniques are the focus of researchers. This paper enlightens on different metaheuristic techniques that are used for optimizing test suite. A brief description of genetic algorithm, particle swarm optimization, ant colony algorithm, artificial bee colony algorithm, algorithm is given along with its pseudo code to facilitate the implementation of these algorithms. This study will be beneficial for both practitioners and researchers.*

Keywords: *Evolutionary algorithms, Ant colony optimization, Artificial bee colony optimization, Particle swarm optimization, Genetic algorithm.*

1. Introduction

Software testing is essential to ensure quality in software industry. Software testing is done to evaluate how well an application confirms to its specifications. Software testing is divided into three stages: first is generation of test data, second is application of data to the software being tested and evaluation. However the main goal of software testing is to generate an optimal test suite that reveals as many errors as possible according to its test adequacy criterion. Industries have spent a lot of time and cost in testing their software. Testing mainly depends on the test cases i.e. inputs taken for the software. Finding that kind of test cases is itself a difficult task. Randomly generated test cases takes a lot of time to test the software.

Evolutionary algorithms are rather new techniques which are emerging these days as difficulties are associated with using mathematical solutions on large scale problems. NP-hard problems are often difficult to solve with these techniques or by using dynamic programming. These techniques are stochastic (random) techniques that mimic the natural behavior of species and generate useful solutions to optimization and search problems.

1.1 Meta heuristic techniques

Evolutionary algorithms are also called metaheuristic techniques. Meta heuristic techniques can often find good solution with less computational effort. Meta heuristic is a higher level procedure which is designed to find, generate or select a lower level procedure that may provide a sufficiently good solution to an optimization problem especially with incomplete or imperfect information with limited computation capacity. [4]

1.2 Meta heuristic techniques and software testing

Search based optimization techniques have been applied to a number of software engineering activities from the requirement engineering to the maintenance.[4] The application of software testing has witnessed intense activity in 2004. There is a number of optimization techniques used for software testing. But no matter which technique is used it is the fitness function that captures the critical information.[19] The first technique introduced was genetic algorithm which was developed by Darwinian. However GA requires long time for processing to find a near optimal solution. In an attempt to improve quality of solutions other algorithms have been developed particularly to avoid being trapped in local minima. Ant Colony Optimization algorithm is inspired by swarm intelligence introduced by Marco Dorgio in 1991 which is one of the first technique to give optimize solutions. ACO can be used in dynamic applications. Artificial Bee Colony algorithm was introduced by Karabora in 2005. It mimics the behavior of bees which are classified as search bees, onlooker bees and scout bees. ABC has the ability to get out of local minima and performs better for local search. Particle Swarm Optimization algorithm was developed by Kennnedy and Eberhart [15]. PSO is inspired by the behavior of birds flocking and the way by which they find unknown destinations, their food sources and their habitat.

2. LITERATURE SURVEY

In this paper the author James Andrews et.al [13] have discussed Nighthawk, a system which uses genetic algorithm to find parameters used for randomized unit testing. Feature subset selection tool is used to access the size and content of the representations which is helpful in reducing the size of representations. This GA achieves 100% of results in only 10% of time.

In this paper Vivek Kothari, Satish Chandra [19] discussed a modification to the artificial bee colony algorithm which reduces its variations by applying genetic operators to the ABC algorithm. In this crossover phase is used to provide better solutions as it helps solutions to persist in the population.

In this paper the author Soma Sekhara Babu Lama et.al [12] discussed generation of feasible independent paths. Artificial bee colony algorithm is used for generation of test data where parallel behavior of the bees makes generation of test data efficient and faster and path is selected based on the priority of all edge coverage criteria. This technique helps to solve local optima problem.

In this paper the author Sanjay Singla et.al [15] presents a technique which is based on genetic algorithm and particle swarm optimization algorithm that is used to automatically generate the test data for data flow coverage. A number of programs of different size and complexity are used to analyze performance which shows its coverage ratio is more.

In this paper the author [21] give focus on generation of test data. A state based software testing is applied by creating a directed dynamic graph which is used to represent the software system under test. The ACO algorithm developed is efficient and generates optimal test data.

In this paper the author Praveen Ranjan Srivastava et.al [10] presents approach which generates test sequence in order to obtain the complete software coverage. Take state diagram of given system under test then find cyclomatic complexity. Decision is based on feasible transition set; pheromone test; heuristic set ; visited status set; probability set. This paper shows that the whole path is covered.The result is also compared with the genetic algorithm it shows that ACO is better in path coverage.

In this paper the author Tai-hoon Ki et.al [22] have discussed the application of genetic algorithm in software testing. This algorithm works on control flow graph. Assigning weights to edges of CFG; distribution of weights; Fitness value is calculated; probability is calculated; crossover is done; mutation is done. In this GA outperforms the exhaustive search and local search techniques by examining the most critical paths first a more effective way to approach testing is obtained which in turn helps to refine effort and cost estimation in the testing phase.

In this paper the author Praveen Ranjan Srivastava et.al [23] have discussed prioritization in ABC which is presently done using factors like code complexity, application feasibility and implementation complexity. It computes the probability value of the sources then traverse the food sources by desolated. The scouts search area for exploring new food sources. The best food source found so far is retained in the memory. In the proposed ABC approach, optimized test suite is generated for each independent path of the program where each path will have two types of data.

In this paper the author Sapna Varshney et.al [24] proposes a novel approach based on genetic algorithm to generate test data for a program. Its performance is evaluated based on data flow dependencies of a program by comparing with random testing. Based on the experimental results on a number of C programs, it shows that the proposed approach outperforms random testing in test data generation and optimization.

In this paper the author Adisrikanth et.al [25] proposes a test case optimization approach using artificial bee colony optimization algorithm. This method generates optimal number of test cases based on the cyclomatic complexity find on the basis of paths. It guarantees full path coverage and chances of falling into local optimum solution are low.

In this paper D.Jeya Mala et.al [26] gives a automated software test optimization framework based on intelligent behavior of honey bees. The proposed system is evaluated based on the coverage based test criteria and its results are compared with sequential ABC, genetic algorithm and random testing. Results shows that ABC outperforms the other approaches in test suite optimization.

3. ALGORITHMS

I. Genetic algorithms

Genetic algorithms were discovered by Holland [20]. Genetic algorithms create population of individuals which is represented by chromosomes where chromosome is composed of genes and a gene is a pair of a name and an integer. These chromosomes are candidate solutions to given problem. Fitness function of chromosomes is calculated. Selective chromosomes form a search space and these represent the value of solutions which is encoded in the chromosomes. After this chromosomes undergo a process of evolution i.e. selection, mutation and recombination.

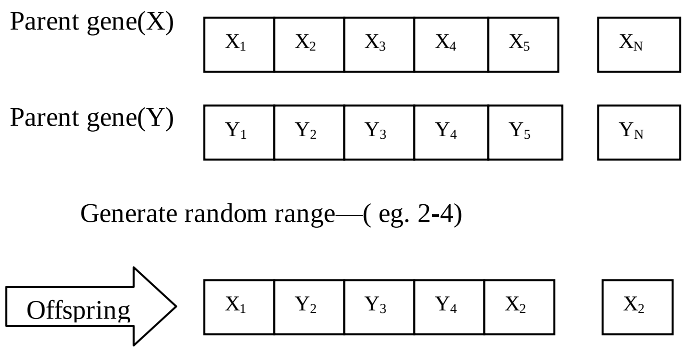

Parent gene(X)
X_1 X_2 X_3 X_4 X_5 X_N
Parent gene(Y)
Y_1 Y_2 Y_3 Y_4 Y_5 Y_N
Generate random range—(eg. 2-4)
Offspring
X_1 Y_2 Y_3 Y_4 X_2 X_2

Fig. 1 Crossover operation to generate offspring

Crossover and mutation operators are used to form a new population. Crossover operator swaps the genetic information but mutation operator changes population slightly either by individual level or on bit by bit basis. The GA in this example is a steady state i.e. an offspring replaces the worst chromosome only if it is better than it.

Genetic Algorithms are best in finding solutions to complex problems. [1]

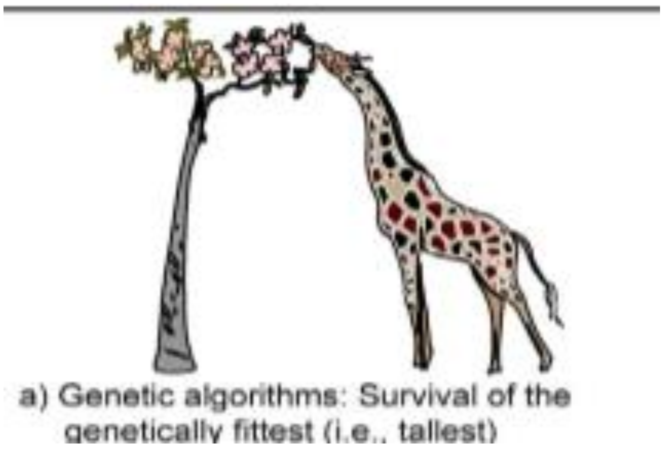

a) Genetic algorithms: Survival of the genetically fittest (i.e., tallest)

Fig. 2

Pseudo code for genetic algorithm is as follows:

```
Begin
  Generate random population of P solutions (chromosomes);
    for each individual I P: calculate fitness (i);
      For i= 1 to number of generations;
      Randomly select an operation (crossover or mutation);
      If crossover;
          Select two parents at random $i_a$ and $i_{b;}$
          Generate an offspring $i_c$= crossover ($i_a$ and $i_b$);
      Else if mutation;
          Select one chromosome i at random;
          Generate an offspring $i_c$ = mutate(i);
      End if
          Calculate the fitness of the offspring $i_{c;}$
          If $i_c$ is better than the worst chromosome than replace the worst chromosome by $i_c$;
     Next i;
   Check if termination= true;
End
```

Strengths:

- Parallel search eliminate undesirable components.
- Mutation operator helps in avoiding stagnation around local minima.
- Likelihood of obtaining a global optimum solution.

Weakness:

- Processing time is high.
- External optimization is there which provides a single solution.

- Its results are less stable.
- These are not sufficient to converge to a solution.
- Memorization is weak.

II. Artificial Bee Colony Algorithm

Bee Colony System was identified by sato and Hagiwara in 1997. Bee Colony Optimization was introduced by Lucic and Teo dorovic in 2001 and Artificial Bee Colony algorithm was introduced by Karabora in 2005. ABC is different from BCO because in ABC we only use scouts and foragers in equal proportion as initial population. Bees are used as agents who explore the minimum set of test cases. Half of the bees will initially start foraging with randomly selected test cases. Now bees will ass new test cases on explored path if adding test case increases its fault detection capacity.

After adding one or more test case the bees return to their hive and exchange information.

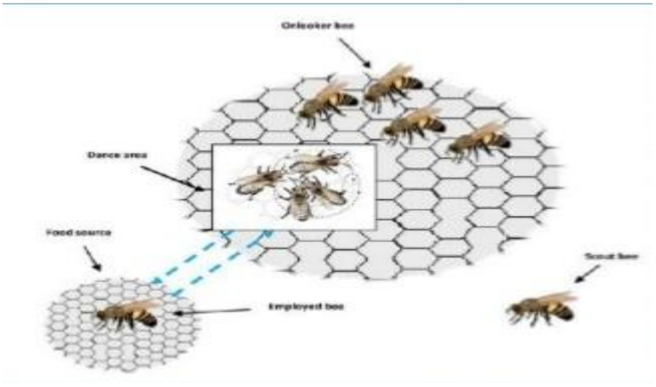


Fig. 3

Pseudo code for artificial bee colony algorithm is as below:

```
1: Initialize the population of solutions
2: Evaluate the population
3: cycle=1
4: repeat
5: Produce new solutions υi,j for the employed bees by using (2) and evaluate them
6: Apply the greedy selection process
7: Calculate the probability values Pi,j for the solutions xi,j by (1)
8: Produce the new solutions υi,j for the onlookers from the solutions xi,j selected depending on Pi,j and evaluate them
9: Apply the greedy selection process
10: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution xi,j by (3)
11: Memorize the best solution achieved so far
12: cycle=cycle+1
13: until cycle=MCN
```

Strengths:

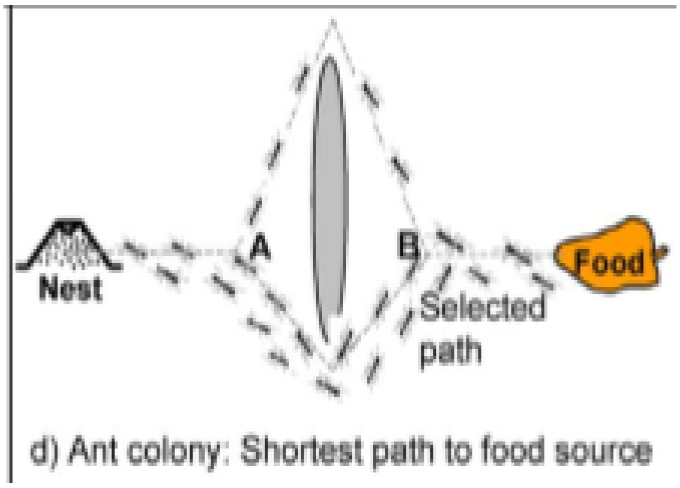
- Ability to get out of local minima
- Requires a few parameters
- Efficient for multivariable and multimodal optimization
- Results are stable

Weaknesses:

- Pre knowledge required
- Slow in sequential processing

III. Ant Colony Optimization Algorithm

Ant Colony Optimization algorithm is based on ant's behavior and their communication by means of pheromone trail, it enables them to find shortest path first. Ants initially search their surroundings and then where to go ants decide by using pheromone information. An isolated ant moves essentially at random, when encountering a previous trail they detect it and decide whether to follow it or not. Where the more ants are following a trail the more that trail gets attractive. This process can be characterized by a positive feedback loop. The quantity of pheromone laid while returning to colony detecting food source depends on quantity and quality of food.



Pseudo code for Ant colony optimization algorithm is as follows:

```
Begin
Initialize the pheromone trail and parameters;
Generate population of m solutions;

For each individual ant k Em ;
Calculate fitness (k);
For each ant determine its best position;
Determine the best global ant;
Update the pheromone trail;
Check if termination= true;
End
```

Strengths:

- Feedback mechanism is easy
- Results are comprehensible
- Adaptive in nature
- Robust and scalable
- Useful in dynamic application
- **Weaknesses**:
- Search efficiency is low
- Search pheromone is scarce
- Processing time is more

4. Particle Swarm Optimization Algorithm

A particle is analogous to a chromosome in genetic algorithms.

PSO does not create new birds from parents as opposed to GAs. The birds only evolve their social behavior and accordingly their movement towards a destination.

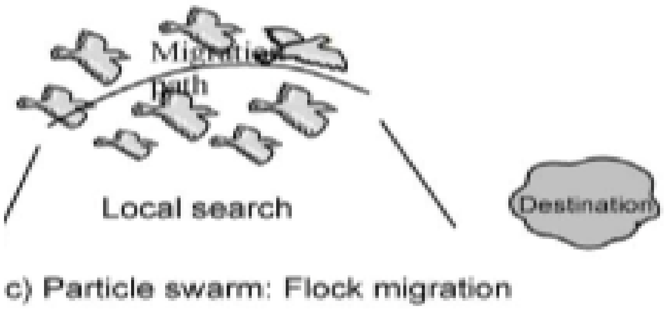
Suppose the solution space of the problem is D-dimensional and the size of particle swarm is m, respectively. Then each particle is defined by two D-dimension vectors, one denotes the particle's location and the other represents its velocity. The location of a particle is a potential solution of the problem, so we should define a fitness function according to the problem. The principle for the definition of the fitness function is the higher fitness, the better solution. Each particle is able to memorize two items, namely the historical best position of itself and the best position of the whole population. Denote the location, velocity, historical best position of the ith particle as Xi, Vi, Pi, and the best position of the whole population as Pg, respectively. Here i=1, 2, …, m, and all of the above four vectors are D-dimensional.

The PSO approach begins with an initial particle population, and the locations and the velocities of which are both randomly produced. Then the velocities and locations are updated according to the following two equations:

$vid = w \times vid + c1\ r1\ (pid - xid) + c2\ r2\ (pgd - xid)$ (1) $xid = xid + vid$ (2)

where $i=1,2,\ldots,m$, $d=1,2,\ldots,D$, w is a inertial parameter, c1 and c2 are learning rates, r1 and r2 are random real values in interval [0,1],

$vid \in [-vmax, vmax]$, and vmax is a designated value. When the fitness of the best population location reaches a designated value or after running a defined upper limit iteration number, the program will output the best solution and terminate.



Figure

Pseudo code for Particle swarm optimization algorithm is as follows:

```
Begin
Generate random population of N solutions;
For each individual I E N ;
Calculate fitness(i);
Initialize the value of the weight factor, w;
For each particle
Set pbest as the best solution for the particle I;
If fitness (i) is better than the pbest;
Pbest(i)=  fitness (i);
End;
For gbest as the best fitness of all particles;
For each particle;
Calculate particle velocity;
Update particle velocity;
End
Update the value of the weight factor w;
Check if termination = true;
End
```

Strengths:

- Faster convergence
- Greater diversity
- Update themselves
- They have memory
- Useful in non linear optimization problems

Weaknesses:

- Not suitable for combinatorial problems

CHALLENGES

There are so many challenges in testing software in time and cost constraint environment. The most eminent challenge faced in software testing is generating an optimized test suite which helps in finding error in less time and path cover is more. As only an appropriate test suite will result into an optimal solution. There are many algorithms and techniques available for software testing but selection of best technique according to requirement is needed. Metaheuristic techniques provide better solutions. Genetic algorithm and Artificial Bee Colony algorithms are best in providing optimal solutions so these can be used in software test suite optimization.

5. CONCLUSION

In this paper, four optimization algorithms were presented. These are genetic algorithm, ant colony optimization, artificial bee colony, particle swarm optimization. The benefits and limitations of these techniques are also discussed. In terms of solution quality and success rate Particle swarm optimization algorithm was found to perform better than other algorithms, while in terms of processing time it is second best algorithm. ABC algorithm gives good results on multimodal problems.ACO algorithm technique is good for solving test case selection and prioritization.

REFERENCES

[1] Marco Dorigo, Senior Member, IEEE, and Luca Maria Gambardella, Member, IEEE, "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, April 1997.

[2] Dervis Karaboga, Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm", Springer Science, April 2007.

[3] David Martens, Manu De Backer, Raf Haesen, Student Member, IEEE, Jan Vanthienen, Monique Snoeck, and Bart Baesens, "Classification With Ant Colony Optimization" IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, Oct 2007.

[4] Mark Harman, "The Current State and Future of Search Based Software Engineering", Future of Software Engineering, IEEE, 2007.

[5] Raluca Lefticaru, Florentin Ipate, "Automatic State-Based Test Generation Using Genetic Algorithms", Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, 2008.

[6] Praveen Ranjan Srivastava1 and Tai-hoon Kiml, "Application of Genetic Algorithm in Software Testing", International Journal of Software Engineering and Its Applications Vol. 3, No.4, October 2009 .

[7] Xiaohu Shi1,2, Yanwen Li3, Haijun Li4, Renchu Guan1, Liupu Wang1 and Yanchun Liang, "An Integrated Algorithm Based on Artificial Bee Colony and Particle Swarm Optimization", Sixth International Conference on Natural Computation, IEEE 2010.

[8] Kewen Li, Zilu Zhang, Jisong Kou, "Breeding Software Test Data with Genetic- Particle Swarm Mixed Algorithm", JOURNAL OF COMPUTERS, FEBRUARY 2010.

[9] B. Akay and D. Karaboga "A modified artificial bee colony algorithm for real-parameter optimization" Information Sciences, 2010.

[10] Praveen Ranjan Srivastava, Km Baby, "Automated Software Testing Using Metahurestic Technique Based on An Ant Colony Optimization," Electronic System Design (ISED), 2010 International Symposium, Dec. 2010.

[11] Qurat-ul-ann Farooq, Muhammad Zohaib Z. Iqbal, Zafar I Malik, Matthias Riebisch, "A Model-Based Regression Testing Approach for Evolving Software Systems with Flexible Tool Support", 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, 2010.

[12] Soma Sekhara Babu Lama, M L Hari Prasad Rajub, Uday Kiran Mb, Swaraj Chb, Praveen Ranjan Srivastavb, a*, "Automated Generation of Independent Paths and Test Suite Optimization Using Artificial Bee Colony", International Conference on Communication Technology and System Design, 2011.

[13] James H. Andrews, Tim Menzies and Felix C.H. Li, "Genetic Algorithms for Randomized Unit Testing", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, February, 2011.

[14] YXiaohui Yan1,2, Yunlong Zhu1, Wenping Zou1, "A Hybrid Artificial Bee Colony Algorithm for Numerical Function Optimization, IEEE, 2011.

[15] Sanjay Singla, Dharminder Kumar, H M Rai and Priti Singla1, "A Hybrid PSO Approach to Automate Test Data Generation for Data Flow Coverage with Dominance Concepts", International Journal of Advanced Science and Technology Vol. 37, December, 2011.

[16] Mohammad Daghaghzadeh, Morteza Babamir, "An ABC Based Approach to Test Case Generation for BPEL Processes", 3rd International Conference on Computer and Knowledge Engineering, November 2013.

[17] Vani Maheshwari, Unmukh Dutta, "Comparative Study of Different Modified Artificial Bee Colony Algorithm with Proposed ABC Algorithm", International Journal of Soft Computing and Engineering , January 2014.

[18] Mustafa Servet Kiran, Ahmet Babalik, "Improved Artificial Bee Colony Algorithm for Continuous Optimization Problems", Journal of Computer and Communications, March 2014.

[19] Vivek Kothari, Satish Chandra, "The Application of Genetic Operators in the Artificial Bee Colony Algorithm", IEEE International Conference on Recent Advances and Innovations in Engineering, May, 2014.

[20] Jogi John, Mangesh Wanjari, "Performance Based Evaluation of New Software Testing Using Artificial

Neural Network" , International Journal of Science and Research, May 2014.

[21] Praveen Ranjan Srivastava1 and Tai-hoon Ki, "Application of Genetic Algorithm in Software Testing ", IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, October 2009.

[22] Praveen Ranjan Srivastava and Tai-hoon Kim, "Developing optimization algorithm using artificial bee colony system" "International Journal of Software Engineering and Its Applications" October 2011.

[23] Sapna Varshney, Monica Mehrotra, " Automated software test data generation for data flow dependencies using genetic algorithm", IJARCSE, February,2014