Volume 4, Issue 6, June 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Evolutionary algorithms for Test Case Generation in Testing of BPEL processes

Gurpreet Kaur*
Research Scholar
RIMT Mandi Gobindgarh, India

Mrs. Gaganpreet Kaur
AsstProf. CSE Department
RIMT Mandi Gobindgarh India

Abstract— In any software development life cycle the testing is considered as very important phase. Test case generation is the necessary task in any software testing. It is necessary to generate the optimized test case to reduce testing cost.The automation of the test cases reduces the effort to generate test case and to select the optimal test suite that can be executed in the minimum time and that meets the proper adequacy criteria. In this paper we put forward the approach for testing the BPEL processes. Our approach is based on metaheuristic technique GA for the test case generation.

Keywords— BPEL; BPEL processes; test case; test case generation; automated test case generation

I. INTRODUCTION

Software testing is an important and critical phase in the development life cycle for the production of efficient and reliable software. On the average of 50% of the total cost of the software is concerned with the software testing process [1].Software Testing is the process of finding the errors in the program. Test Case Generation is important and critical task in the testing activity. The test case generation is the challenging part in the testing activity .The cost incurred in testing can be reduced by the automation of the generation of test cases by using various techniques .Search based techniques played an important role in the automation of the test generation. Search based techniques are intended to generate the possible solution and search the best optimal solutions from the generated ones. The researchers have shown a keen interest in the areas of the search based software for some years. On the other hand, the nature of Software Engineering problems is ideal for the application of metaheuristic techniques, as is shown in the work of Harman and Jones [2]. Service-Oriented Architecture (SOA) is a new architectural style for developing distributed business applications. These are increasingly being built by organizations to achieve dynamic and better Business Process Management. Many companies outsource their business to the third party . The Service Oriented Architecture is the computing paradigm that is widely adopted for the distributed application development. Its vision is preferably implemented using web services which are built and deployed on the heterogenous platforms and orchestrated to achieve a particular business goal [3]. Since web services are the new paradigm for deploying and developing the business process the composition and the testing of web service composition is new challenge .The enterprise wrap their internal business process as a web service and publish it into the public directory so that new enterprise can combine them to add more value added services. BPEL is the standard for web services composition as standardized by OASIS. Testing the BPEL service composition is the new challenge that attracted the researchers to develop new automated testing techniques in order to reduce the cost and time of the testing the business processes. So the researchers start applying metaheursitic techniques for the automate test case generation and optimization in BPEL processes.this paper focuses on the BPEL processes , testing of the BPEL processes,Genetic Algorithm for the test case generation for BPEL processes and Ant Colony Optimization technique for the optimization of the Test Cases.

.

II. BASIC CONCEPTS

A. Test Case Generation

Software testing is the time and resource consuming task in the development cycle of the software. Test case generation is important and critical task in software testing. In testing the tester is intended to create an optimal code of test data to test the program of the software. The quality of the software is determined by the number of error found by the test case .it is difficult task to find the optimal case among the number of test cases. Many efforts have been done by the researchers for solving the problem of optimization .they applied many search based algorithm for finding the optimal solution of test cases. The automation of the test cases is still an growing research area since the auto- mated testing tools simply generate test data with only less consideration on amount of time spent for test data generation and selection of test cases based on the test adequacy criterion [4].

B. Overview of BPEL processes

BPEL is the process execution language that is used for the web service composition .It is an XML based language and it provides XML semantics for specifying business behavior based on web services. It is defined in terms of its interaction with the partner processes. BPEL is designed to address the composition requirements in the web service environment.

BPEL language supports both paradigms as orchestration and choreography. The Executable Processes follow the orchestration paradigm because they specify the exact detail of the interaction between the services. The choreography paradigm is followed by the Abstract business processes as these lack the interaction details. The specification of BPEL includes the composition model and the XML composition language .It has nothing to do with the development and the runtime environment. The services which are intended to be composed by the BPEL process are needed to be invoked. As mentioned above that the BPEL process is defined in terms of its partners, hence partners play important in BPEL processes. Any web service that interacts with the business process is called the partner of that process. The partner can interact with the process in any way i.e. it can invoke the process or invoked by the process. The BPEL process is defined in XML document. The BPEL document consists of two parts: Declarations and Specification. The partner links and the port types are defined in the declaration part. The variables which store the intermediate values and the services are represented by the partner link .Port type define the interaction between the processes. Some of the important concepts in BPEL are:

Partner Links: it represents the service and provides the communication channel to remote web services. The partner link has a name which is to be used throughout the process. The partner link definition does not specify where to find partner it is assumed that BPEL middleware handles binding the partner links.

Variables: These store the message data of the web service interaction and control data of the process. The variable must be declared before use with name and type. The type of variable is WSDL message. The variables can be declared global or local.

Basic activities: These perform the basic functions of the business process. These are receive, invoke, reply, assign, wait.

Structured Activities: These define the block oriented control flow. It also defines the parallel execution, conditional branching, receipt of the message, links that represent the synchronization between two activities.

Handlers: These are defined to dealt with the exceptional situations. These are used to specify deadlines on the process level. To find the internal fault , the fault handler is used.

BPEL process is a terminology which defines the entire scenario of the an organization . The bpel process is organized in such a manner that high entity will be place in the first part and lower entity will be placed under the main entity . If any sub entity has a child entity , the least child entity will have the access to all the base entities provided . The process of creation of the BPEL process resolves the high and low level hierarchy so that the organization can run properly . The BPEL process also removes the circumstances of the of any duplicate node in the list.

```
<process name="loanapproval" [...]>
 <-- declarations -->
 <variables>
  <variable name="riskAssessment"

messageType="asns:riskAssessmentMessage"/>
[...]
</variables>
<partners>
<partner name="customer" [...]/>
<partner name="assessor" [...]/>
<partner name="approver" [...]/>
</partners>
<-- behaviour of the business process -->
<flow>
   <links>
      <link name="receive-to-assess"/>
      <link name="assess-to-setMessage"/>
[...]
   </links>
   <receive name="receive1"
         partner="customer" [...]>
[...]
   </receive>
   <invoke name="invokeAssessor"
          partner="assessor"
          portType="asns:riskAssessmentPT"
          operation="check"
          inputVariable="request"
          outputVariable="riskAssessment">
    <target linkName="receive-to-assess"/>
    <source linkName="assess-to-setMessage"
           transitionCondition=
```



```
            "bpws:getVariableData
            ('riskAssessment','risk') ='low'"/>
    <source linkName="assess-to-approval"
            transitionCondition="
            bpws:getVariableData
            ('riskAssessment','risk') !='low'"/>
    </invoke> [...]
</flow>
</process>
```

Fig.1 Extract from the "loan approval" BPEL specification [5]

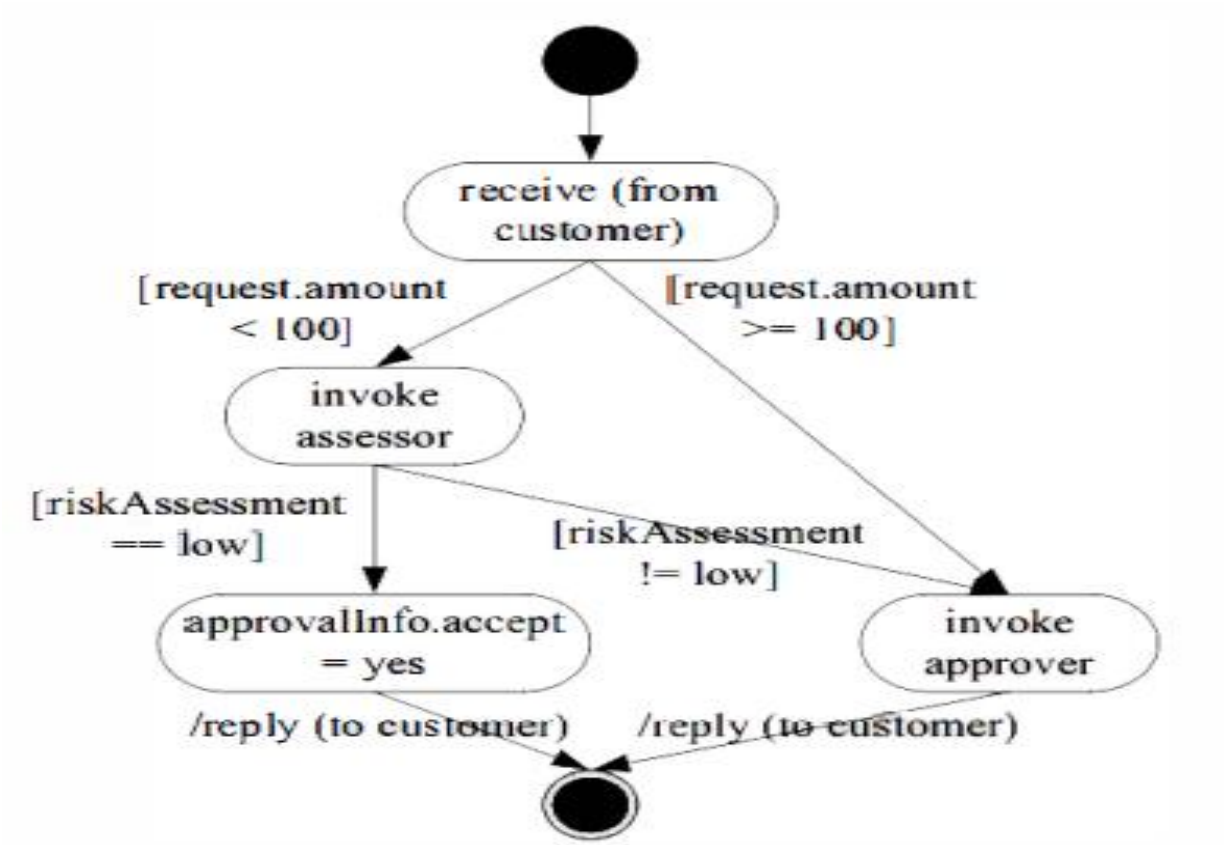

Fig 2. State graph for Loan Approval process [6]

III. RELATED WORK

Random methods for test case generation are exhaustive, time consuming and unreliable .So the test case generation technique must be automated .Size and complexity are main issues in testing problems. By applying the metaheuristic techniques to the problems these issues can be solved. Many researchers have put forward different techniques for the automation of test case generation in the testing process. They applied the heuristic to provide the solution for the computational problems that are NP Hard and NP Complete. The general idea behind the automated test case generation is that the set of test cases forms the search space and adequacy criterion is coded as fitness function [7] This section deals with the relates researches that has been done for test case data generation using Genetic algorithm and the approaches for testing the BPEL processes.

.

A. GA based Test Case Generation approaches.

Xanthakis et al.[8] attributed to apply optimization technique to software engineering for the first time. It used the GA's to generate test data for the structural coverage. Davie et al .[9]developed GA based approach for test case generation for an expert system. Tonella [10] proposed a GA based test case Generation technique for unit testing of the classes in Object Oriented Software . Roberiroet et al.[11]use GP approach for automated test Case generation for Object Oriented software. Liaskos et al. [15] use the artificial Immune system for path testing of object Oriented Software .Li et al. [12]applied an ACO based approach for automatic test Case generation in the state base software testing. R. Krishnamoorthi has proposed a new test case prioritization technique using Genetic Algorithm (GA)[13]. Dr. Arvinder Kaur has proposed an approach for regression test suite prioritization based on BCO algorithm[14].

B. Test Case Generation appraoches for BPEL processes

The authors [17] proposed a graph based approach for BPEL test case generation to deal with BPEL concurrency semantics. In this approach an extension of CFG is defined to represent a BPEL program in graphical model. In [18],a method to test composite web service was proposed which used model checking. A model checker SPIN had exploited by the authors [19]. In this the transition coverage criterion is used for choosing the test case. The authors in [16] has surveyed the approaches to support test case generation for BPEL processes and the tool is presented for test case generation that is based on the design and implementation of data dependency. The developers define the data dependency using XPath expression. The authors used Category Partition Method (CPM)to generate test cases on the basis of possible paths[20].Since many techniques and approaches have been developed or proposed by many authors .But the use of metaheuristic techniques for automated test case generation in BPEL testing has not gained much attention of the researchers. The first approach for automatic test case generation based on metaheuristic technique was proposed in [21].the authors proposed a scatter search based method to generate test case for BPEL processes. In this the adequacy criterion used is transition coverage .Taking similar approach for formal model and generation of the test data but the difference is only in the application of the algorithm the authors [22] proposed a new approach for automatic test case generation [ABC based].they transform BPEL into static graph according to some rules and use ABC algorithm to work on the state graph. Artificial Bee Colony Optimization is a nature inspired optimization technique which is based on the foraging behaviour of the bees to find the solutions in hand. ABC algorithm mixes local search method carried out by employed bees with global search method managed by onlooker and scouts.

IV. TEST CASE GENERATION AND OPTIMIZATION

A. Genetic Algorithm

Genetic algorithms (GA), is based on a direct analogy to Darwinian natural selection and genetics in biological systems, is an alternative to conventional heuristic methods. GA work with a set of candidate solutions called a population. The GA obtains the optimal solution after a series of iterative Computations according to the principle of 'survival of the fittest'. In GA the successive populations of alternate solutions represented by a chromosome are generated first, until acceptable results are obtained. GA can deal with large search space and hence has less chance to get local optimal solution than other algorithms .The quality of the solution evaluation step is assessed by the fitness function. The fitness function values is impacted by the crossover and mutation functions. Chromosomes are selected for reproduction by evaluating the fitness value. The chromosome that has the higher fitness value has the probability to be selected for recombination pool. Once individuals or potential solutions have been selected as parents, they must produce offspring to populate a new generation for the genetic algorithm . The two operators that are almost always used in every genetic algorithm are recombination and mutation. The common operators in the genetic algorithm are:

Recombination : In the recombination evolutionary process the two parent chromosomes combine to form a offspring. This is called 'crossover' Recombination is similar to the sexual reproduction where each characterstic comes from the one of two parents. There are two forms of recombination i.e.the single –point crossover and uniform crossover. In single-point crossover a common point of exchange between the two parent chromosomes is set at a randomly selected location. The offspring chromosome adopts all coding before the common point from one parent and all coding after the common point from the other [23]In uniform crossover, the coding at any given location is chosen with 50/50 probability from either parent's coding at that same location 'The Crossover rate' parameter determines the number of choromosomes to be selected for recombination in each population.

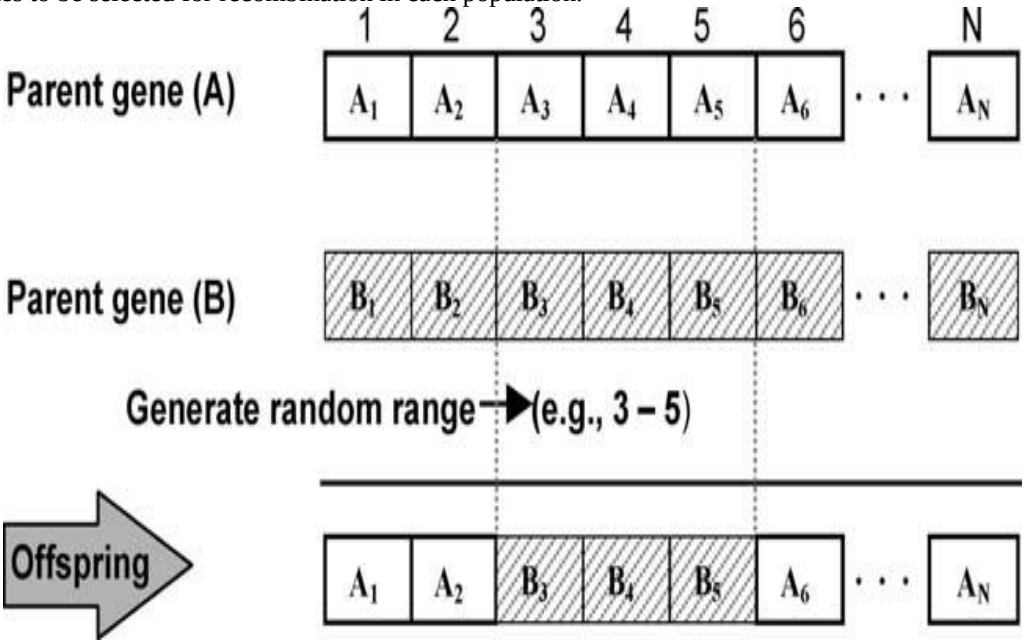

Fig 3. crossover operation[24]


```
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
         If crossover;
            Select two parents at random i_a and i_b;
            Generate on 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 then
         replace the worst chromosome by i_c;
       Next i;
     Check if termination=true;
End;
```

Fig 4. Pseudocode for Genetic algorithm

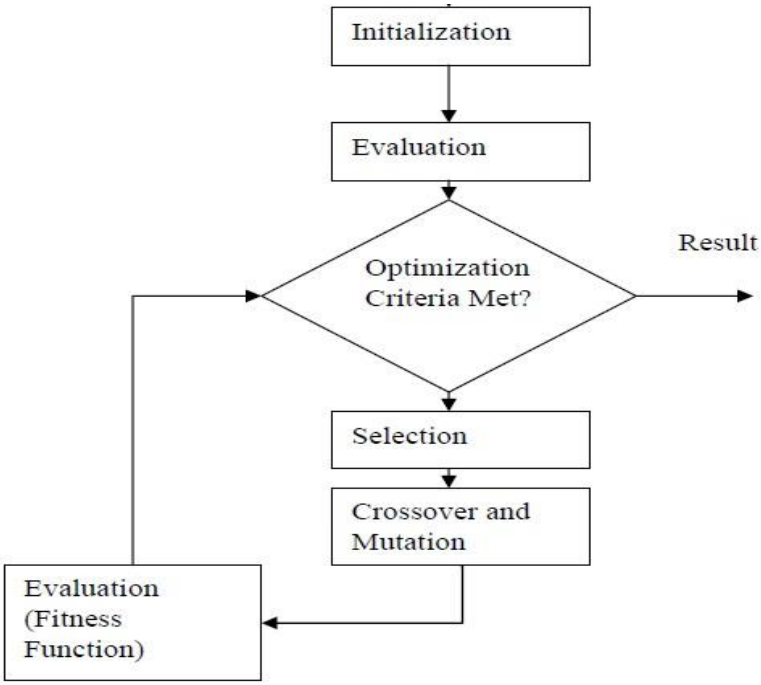

Fig.5 Flow chart of Genetic algorithm

B. Ant Colony Optimization

ACO is metaheurisitic algorithm that is used to obtain solutions to hard computational problems in reasonable amount of time.The foraging behaviour of the real ants is the inspiring source of ACO. Ants initially explore the area surrounding their nest in a random when these are searching for food. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. the ants deposits a chemical pheromone trail on the ground during the return trip. The quantity of pheromone will guide other ants to the food source. There is a communication between the ants trails and it enables them to find shortest paths between their nest and food sources. This communication is indirect between the ants via pheromone .The term given to communication is called stigmergy . Deneubourg et al. [25] thoroughly investigated the pheromone laying and following behavior of ants.In the 'double bridge experiment 'the nest of the ants is connected to the food source via two bridges.The ants start exploring around the nest for the food source and reach the food source.The argentine ants deposit the pheromone along their path from nest to the food source. Initially, each ant randomly chooses one of the two bridges. One of the bridges present a high concentration of the pheromone than the other due to random fluctuations hence this bridge attracts more ants. After sometimes the whole colony starts using the same bridge for searching the food source. In ACO, a number of artificial ants build solutions to the optimization problem and exchange information on the quality of these solutions via a communication that is similar to the one adopted by real ants. Various Ant Colony Optimization Algorithm have been proposed . Ant System is the original ant colony optimization that was proposed in early nineties.


```
Set parameters, initialize pheromone trails
while termination condition not met do
    ConstructAntSolutions
    ApplyLocalSearch (optional)
    UpdatePheromones
endwhile
```

Fig.6 Pseudocode for ACO

After initialization, the metaheuristic iterates over three phases: a number of solutions are constructed by the ants at every iteration The description of the three phases:

ConstructAntSolutions: A set of *t* artificial ants constructs solutions from elements of a finite set of available solution components **C** = {*cij* }, *i* = 1, . . . , *n, j* = 1, . . . , |*i* | . From an empty partial solution ,the construction of the solutions starts. *s* = ∅.The partial solution *s* is extended by adding a feasible solution component from the set at each construction step . The set of components that can be added to the current partial solution *s is defined as* ***N(s)*** ***⊆ C.*** The decision to choose a solution component from **N***(s)* is guided by a mechanism, which is biased by the pheromone associated with each of the elements of **N***(s)*.

ApplyLocalSearch: Once solutions have been constructed, and solutions obtained by the ants through a local search are improved before updating. This phase, is optional.

UpdatePheromones: To increase the pheromone values associated with good solutions, and to decrease the values that are associated with bad solutions, is the main aim of the this phase. Usually, this aim is achieved by Phermone evaporation and by increasing the pheromone levels associated with a chosen set of good solutions.

V. CONCLUSION

The Testing of the software and the BPEL is the important issue. To reduce the cost incurred on testing the automation of the test case generation process is necessary. The software engineering deals with the Optimization problems .The Optimzation the test case is also an important issue in Testing This paper is intended to use Genetic Algorithm for the test case generation in the testing of the BPEL processes.This paper gives the introduction to GA and ACO algorithms.The Genetic Algorithm can be used for the test case generation for the BPEL processes amd The ACO can be used for Optimization of the Test Cases after their generation by GA.

REFERENCES

[1] R. Ramler and K. Wolfrnaier, "*Economic perspectives in test automation: balancing automated and manual testing with opportunity cost,* " in Proceedings of the 2006 international workshop on Automation of software test, 2006, pp. 85-91.

[2] M. Harman, B.F. Jones, "*Search-based software engineering*", Information and Software Technology, 43(14), 2001, pp. 833-839.)

[3] M. Grottke and K. S. Trivedi, *"Fighting bugs: Remove, retry, replicate, and rejuvenate, "* Computer, vol. 40, no. 2, pp. lO7-lO9, 2007. Trans. Roy. Soc. London, vol. A247, pp. 529-551, April 1955.

[4] *Hangal, S.—Lam, M. S.: Tracking Down Software Bugs Using Automatic Anomaly Detection.* ICSE 2002, pp. 291–301.

[5] Organization for the Advancement of Structured Information Standards (OASIS), Web Services Business Process Execution Language (WSBPEL), URL:http://www.oasis-open.org.]

[6] R. Blanco, J. Garcia-Fanjul, and J. Tuya, "*A first approach to test case generation for BPEL compositions of web services using scatter search*" in Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on, 2009, pp. 131-140.

[7] P. McMinn, "*Search-based software test data generation: a survey*, " Software Testing, Verification and Reliability, vol. 14, no. 2, pp. lO5-156, 2004.

[8] Xanthakis, S., Ellis, C., Skourlas, C., Le Gall, A., Katsikas, S., and Karapoulios, K. "*Application of Genetic Algorithms to Software Testing*" In Proceedings of the 5th International Conference on Software Engineering and Applications, pages 625–636, Toulouse, France.

[9] Davies, E., McMaster, J., and Stark, M. "*The Use of Genetic Algorithms for Flight Test and Evaluation of Artificial Intelligence and Complex Software Systems*." Technical Report AD-A284824, Naval Air Warfare Center, Patuxent River.

[10] Tonella, P. "*Evolutionary Testing of Classes*" In Proceedings of the 2004 ACM SIGSOFT Iinternational Symposium on Software Testing and Analysis (ISSTA '04), pages 119–128, Boston, Massachusetts, USA. ACM.

[11] Ribeiro, J. C. B., Zenha-Rela, M., and de Vega, F. F. (2007a*) "An Evolutionary Approach for Performing Structural Unit-Testing on Third-Party Object-Oriented Java Software."*In Proceedings of International Workshop on Nature Inspired Cooperative Strategies for Optimization (NICSO '07), pages 379–388, Acireale, Italy. Springer.

[12] Li, H. and Lam, C. P. (2005a). "*An Ant Colony Optimization Approach to Test Sequence Generation for Statebased Software Testing*." In Proceedings of the 5th International Conference on Quality Software (QSIC'05), pages 255–264, Melbourne, Australia. IEEE Computer Society

[13] R.Krishnamoorthi and S.A.Sahaaya Arul Mary " *Regression Test Suite Prioritization using Genetic Algorithm*"International Journal of Hybrid Information TechnologyVol.2, No.3, July, 2009 35

[14] Dr. Arvinder Kaur. Shivangi Goyal"*A Bee Colony Optimization Algorithm for Fault Coverage Based Regression Test Suite Prioritization* " International Journal of Advanced Science and Technology Vol. 29, April, 2011 17

[15] Liaskos, K. and Roper, M. (2007). "*Automatic Test-Data Generation: An Immunological Approach.*" In Proceedings of Testing: Accademic and Industrial Conference – Practice and Research Techniques (TAIC PART '07), pages 77–81, Windsor, UK. IEEE.

[16] Choy Kho Yee, "*Design and Implementation of Test Case Generation Tool for BPEL Unit Testing*, " Osaka University, Japan, 2008.

[17] Yuan Yuan ,Zhongjie,Li. "*A graph based approach to BPEL4WS test case generation* "in the proceedings of international conference on software engineering advances(ICSEA 106)IEEE,2006.

[18] H. Huang, W.-T. Tsai, and R. Paul, "*Automated model checking and testing for composite web services*, " in Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on, 2005, pp. 300-307.

[19] J. Garcia-Fanjul, 1. Tuya, and C. De La Riva, "*Generating test cases specifications for BPEL compositions of web services using SPIN*" in International Workshop on Web Services-Modeling and Testing (WS-MaTe 2006), 2006, p. 83

[20] T. Bakota, A. Beszedes, T. Gergely, M. I. Gyalai, T. Gyimthy, and D. FUleki, "*Semi-automatic test case generation from business process models*, " in 11th Symposium on Programming Languages and Software Tools, 2009.

[21] R. Blanco, J. Garcia-Fanjul, and J. Tuya, "*A first approach to test case generation for BPEL compositions of web services using scatter search*" in Software Testing, Verification and Validation Workshops, 2009. ICSTW'09. International Conference on, 2009, pp. 131-140.

[22] Mohammad Daghaghzadeh, Morteza Babamir "*An ABC Based Approach to Test Case Generation for BPEL Processes*" 3rd International Conference on Computer and Knowledge Engineering (ICCKE 2013), October 31 & November 1, 2013, Ferdowsi University.

[23] Marczyk, Adam (2004). "Genetic Algorithms and Evolutionary Computation." The Talk.Origins Archive. Retrieved December 4, 2004 from the World Wide Web: http://www.talkorigins.org/faqs/genalg/genalg.html

[24] Emad Elbeltagia,, Tarek Hegazyb, Donald Grierson "*Comparison among five evolutionary-based optimization algorithms*" Advanced Engineering Informatics 19 (2005) 43–53

[25] J.-L. Deneubourg, S. Aron, S. Goss, and J.-M. Pasteels, "*The self-organizing exploratory pattern of the Argentine ant*" *Journal of Insect Behavior*, vol. 3, p. 159, 1990.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, "*Ant System: Optimization by a colony of cooperating agents*" IEEE Transactions on Systems, Man, and Cybernetics—Part B, vol. 26, no. 1, pp. 29–41, 1996.

[27] S. Fenet and C. Solnon, "*Searching for maximum cliques with ant colony optimization,*"in Applications of Evolutionary Computing, Proc. EvoWorkshops 2003, ser. LNCS, G. R. Raidl etal., Eds., Springer Verlag, vol. 2611, pp. 236–245, 2003.

[28] L. Bianchi, L.M. Gambardella, and M. Dorigo, "*An ant colony optimization approach to the probabilistic traveling salesman problem*," in *Proc. PPSN-VII*, ser. LNCS, J. J. Merelo et al.,Eds., Springer Verlag, vol. 2439, pp. 883–892, 2002.

[29] Alander, J. T., Mantere, T., and Turunen, P. (1997a)."*Genetic Algorithm based Software Testing*" In Proceedings of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA '97), pages 325–328, Norwich, UK. Springer-Verlag.

[30] Alander, J. T., Mantere, T., Turunen, P., and Virolainen, J. (1996)" *GA in Program Testing*" In Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications (2NWGA), pages 205–210, Vaasa, Finland.

[31] Zhu, H.—Hall, P.—May, J. *"Software Unit Test Coverage and Adequacy. ACM Computing Ssurveys",* Vol. 29, 1997, pp. 366–427..

