International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

# GAPSO: Optimal Test Set Generator for Pairwise Testing

**M. Lakshmi Prasad, A. Raja Sekhar Reddy, J.K.R. Sastry**

***Abstract:*** ***Exhaustive testing is impossible for all sorts of software systems, owing to the cost and time consumption. Combinatorial testing is the solution to this issue and aims at picking the necessary set of parameters which can ensure high degree of interaction between the parameters. This paper presents a new approach for generating unique test cases by exploiting Genetic and Particle Swarm Optimization (GAPSO) algorithm for achieving pairwise testing. The generated test cases are refined, so as to arrive at the optimal test set. The outcome of the proposed algorithm is the minimal count of high quality test cases.***

***Keywords: Exhaustive testing, combinatorial testing, pairwise testing, genetic algorithm, particle swarm optimization.***

## I. INTRODUCTION

The most crucial step of any software development process is the software testing phase. The main objective of this phase is to ensure the quality of the software product, by spotting the software defects. The software testing phase can be performed either manually or by automated testing tools. However, software testing consumes more time and effort. Automated testing techniques are more preferable than manual testing, owing to its simplicity and ease of use [1-5].

Now-a-days, automatic test case generation techniques based on bio-inspired algorithms have gained substantial research interest. The major challenges of a software testing technique are its cost-efficiency, maximum error detection capability and so on. Test case generation is the most critical issue in the software testing process. The count of test cases grows with respect to the count of parameters. The count of possible test cases is directly proportional to the count of parameters [6]. Hence, this kind of testing is impossible to accomplish, as it performs at the cost of time and other resources.

An efficient test generation technique should produce minimal count of test cases in a lesser period of time, quality testing and with minimal resource consumption. Combinatorial testing provides a solution to this problem by taking only a few combined set of input parameters into account.Pairwise testing has its roots on combinatorial testing and it aims at picking out all the possible combinations of user inputs [7]. Pairwise testing minimizes the count of test cases which paves way for reduced time consumption, effective cost and resource utilization.The main objective of this article is to present an optimized pairwise test case generator, which can assure lesser time consumption and resource utilization. The proposed test case generator produces minimal count of test cases and it relies on the fusion of Genetic algorithm (GA) and Particle Swarm Optimization (PSO) algorithm. Both GA and PSO are bio-inspired algorithms, which focuses on the concept of optimization.The remainder of this paper is organized as follows. Section 2 presents the review of literature with respect to pairwise testing. Section 3 presents the preliminaries and background of the proposed work. The proposed methodology is presented in section 4. Finally, the concluding remarks are presented.

## II. LITERATURE REVIEW

The main objective of this section is to present the existing literature with respect to pairwise testing. In [8], a technique is presented for producing the test case configurations. The fitness function of this technique relies on the count of distinct pairs divided by the total possible pairs. Yet, the performance of the technique is evaluated with a simple input set.A genetic algorithm is presented in [9, 10] for pairwise test case generation. This technique represents the chromosomes by integer array encoding. Two fitness functions are defined by this method. The concept of similarity mutation, value occurrence and pair occurrence mutation is defined.In [11], Genetic algorithm is utilized for the generation of test cases. The initial solution is presented by the hamming distance. This algorithm presents a new crossover strategy as a part of it. In [24], a survey was conducted which gives comprehensively the status of use of combinatorial methods for testing the embedded systems and the combinatorial methods in general. Lakshmi Prasad and Dr. J. Sasi Bhanu et al.,[25][26] have proposed a Graph Based Strategy (GBS) that deals with generation of test cases based on the input domain considering testing a standalone embedded system. Lakshmi Prasad and Dr. J. Sasi Bhanu et al., [27][28] had proposed a Particle swarm Optimization (PSO) for the generation of test cases based on the output domain considering with special consideration to testing standalone embedded system. In [29][30], Neural Network Based Strategy (NNBS) have been presented that deals with generation of test cases based on the input domain and output considering testing standalone embedded systems.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
**Dr. M. Lakshmi Prasad**, Department of CSE, NBKRIST, Nellore, India.
**Dr. A. Rajasekhar Reddy**, Department of CSE, NBKRIST, Nellore, India.
**Dr.JKR Sastry**, Department of ECSE, KL University, Guntur, India.

The method has also been used for generating the test cases that can be used for testing a WEB based application [22] [23].

Lakshmi Prasad et al., [31][32][33] had presented an Optimal Selection Procedure (OSP) and Optimal Mining Technique (OMT) deals with generation of test cases based on the multi-output domain and multi-input domain that can be used for testing standalone embedded system. In [34], testing methods for testing standalone embedded system have been presented that deals with generation of test cases through combinatorial methods. To conclude this literature survey, several approaches have been proposed for achieving reduced set of optimal test cases. However, the time consumption of the existing approaches is comparatively larger. Thus, there is need to formulate a strategy which demands lesser time for generating optimal test cases.

## III. GAPSO PAIRWISE TEST SET GENERATION

Lakshmi Prasad et al., **[27][28]** had presented a Particle swarm Optimization (PSO) has presented the generation of test cases based on the output domain considering with special consideration to testing standalone embedded system.

This section intends to present the proposed pairwise test case generator, which is named as GAPSO Pairwise Test Set Generation. The entire approach can be decomposed into two phases. They are GA phase and PSO phase and are explained in the forthcoming sections.

**Phase I: GA**

Genetic algorithm imitates the evolution of species by selecting the fittest candidate [12]. Initially, the population is the possible set of solutions and the fittest solution is picked out by the fitness function. There are three significant operators in GA, which are selection, crossover and mutation. The obtained solution is enhanced over several iterations. Finally, the process is stopped as soon as the stopping condition is met.

This original GA algorithm is improvised in terms of mutation. The initial solution can be produced in a random fashion or by employing hamming distance, Euclidean distance etc., [13-15].

The proposed approach produces the initial solution by Szymkiewicz-Simpson coefficient. This coefficient measures the overlap between sets and is calculated by the size of intersected sets by the smallest size of the two sets and is given in the below equation.

$$ss_{coeff} = \frac{|x \cap y|}{\min(|x|, |y|)} \quad (1)$$

In the above equation, *x and y* are value sets. $|x|$ *and* $|y|$ are the size of *x and y* sets respectively. As an initial step, test cases are generated in a random fashion. This step is followed by determining the test case with minimal overlap, which is calculated by (1) and that particular test case is added to the test set.

The next step is to analyse all the parameters and their values, so as to have optimal test case outcome. The occurrences of the test parameters are examined, in order to boost up the uniqueness of the test cases. Let P1, P2, P3 are parameters and V1, V2 and V3 are the count of values of every parameter. The parameter occurrence examination is achieved by the following formula.

$$Min\_occ_i = (Max\ (v1, v2, v3, v4))\ for\ 1<=i<=s; \quad (2)$$

Where $v_1, v_2 v_3 \ldots v_s$ are the count of values of a parameter 1, 2,…, s respectively. The main objective of the above equation is to control the occurrence frequency of the same test case. For instance,

*Let p1= (a, b, c, d); p2= (e, f, g) p3= (h, i, j), p4= (k, l). Hence, v1=4; v2=3; v3=3; and v4=2.*

The threshold of minimum occurrence of a parameter is calculated by (2), which is obtained by excluding the current parameter. Thus, the parameter *p1* can occur at the maximum of thrice and the remaining parameters can be present for four times.

Consider that the occurrence of *g* and *e* of parameter 2 are counted as six and two times respectively. This case can be handled by the replacement of *g* and *e*, so as to balance the occurrence frequency of the parameter. This constraint potentially helps the test case generation with unique combination of parameters and is viable for the production of unique combinations of parameters at the initial stage alone. The so formed test set is refined through several iterations to arrive at unique test cases.

This step is followed by the computation of unique pairs of every test case. For instance, the test case *aehl* has different pairwise combinations of values and they are *ae, eh, hl, al, ah, el* etc. The occurrence frequency of these pairs is counted and the count decides the whether or not the replacement is necessary. The distinct pairs of all the test cases are framed and the so framed pairs are passed on to the Particle Swarm Optimization algorithm.

**Phase II: PSO**

Particle Swarm Optimization (PSO) algorithm imitates the biological nature of animals, birds or fish. The swarm or a group of animals collectively involve in food hunt. This meta-heuristic approach is initially formulated by Eberhart and Kennedy in the year 1995 [16]. The implementation of PSO is simple because minimal parameters are needed to be set. Some of the important entities of PSO algorithm are particle, particle's position and fitness function.

Each and every participant of a swarm is stated as the particle and the particle's position is tracked by the particle's position. The fitness function determines the quality of the solution being attained. Initially, all the particles are randomly assigned with the velocity and location. For every round of iteration, the particle *n* stores the best location of the particle and also the best global location of the swarm. In the next iteration, the velocity and position with respect to the last attained solution. This process repeats until the optimal solution is attained.

In this work, the distinct pairs of test cases are generated by the first phase and the pairs are passed as the input to the second phase. The position and velocity of every pair is assigned in a random fashion. The fitness function of this algorithm is the occurrence frequency of the pair. This fitness function controls the redundant pair occurrence and thus generates more unique pairs. The process of replacement happens with respect to the occurrence frequency of the values of parameters.

This idea produces very minimal combinations of highly qualified test cases. The algorithm for this work is presented below.

**Algorithm**

*//Phase 1*
*Input: Set of parameters with values;*
*Output: Distinct pairs;*
*Begin*
*// GA Mutation*
*Generate random test cases;*
*Compute $SS_{coefft}$ by (1);*
*Pick the test case with $min_{olp}$ and load them into the test set;*
*Calculate $Min\_occ_i$, for all parameters and find the threshold;*
*Replace $Max_{occval}$ with the $Min_{occval}$*
*Generate refined test cases;*
*Produce all distinct pairs of refined test cases;*
*End;*
*//Phase 2*
*Input: Distinct pairs of test cases;*
*Output: Distinct test cases;*
*Begin*
*Test Set:=Null;*
*Bestcase:=Null;*
*P_n:= Count of distinct pairs of test cases;*
*For i=0;i<P_n;i++*
*Do*
*Initialize the position and velocity of particle pl_i randomly;*

*plBest_i:=Null;*
*GBest:=Null;*
*// Fitness value computation*
*If f(p1_m )>f(p1_n)*
*Then replace p1_m with p1_n;*
*plBest_i:= p1_n*
*If f(p12_op )>f(p12_qr)*
*Then replace p12_op with p12_qr;*
*GBest:= p12_qr;*
*Populate the test case;*
*Return test set;*
*End;*

The above presented algorithm takes any number of parameters with different values. This is followed by the generation of random test case and the $\boldsymbol{SS_{coefft}}$ is calculated. The first step towards test case minimization is achieved by picking the top ranking test cases with minimal overlap. The test set is loaded with such test cases.

This step is followed by the computation of minimum occurrence of $i^{th}$ parameter $\boldsymbol{Min\_occ_i}$, which acts as the threshold. The presence of parameter below the threshold is replaced by the other value of the same parameter. The refined test cases are then generated and the distinct pairs of all test cases are computed. The next goal is to generate unique combinations of test cases, which is attained by PSO algorithm. In the initialization phase, the position and the velocity of particles are randomly allocated. For every distinct pair of test cases the occurrence frequency of parameter values is checked and replaced accordingly. By this way, a unique test set with high quality is generated.

**Phase III: Case study**

Let P1, P2, P3, P4 are parameters with values (a b c d), (e f g), (h i j), (k l). The count of values of a parameter is denoted by v. Thus, v1=4; v2=3; v3=3, v4=2; the test cases are generated randomly and the overlap coefficient is found. For instance, consider the test cases (a e h k), (a e i k) and (c g j l) (b f i l), whose overlap coefficient is computed as 0.75 and 0.25.

From the computed overlap coefficient, 0.25 is the least and it enters the test set. On keen observation, it is evident that the test cases with least overlap coefficient arrive at several unique parameter values. Same way, the test case with least overlap coefficient enters the test set.

This step is followed by the computation of $\boldsymbol{Min\_occ_i}$, which is 3 for P1 and 4 for P2, P3 and P4. Thus, P1 must occur at least thrice and the other parameters have to occur for four times. Thus, the test cases are framed and are shown in table I.

**Table- I: Refined Test Cases**

| *P1* | *P2* | *P3* | *P4* |
|---|---|---|---|
| **a** | E | h | k |
| **a** | F | i | l |
| **a** | G | j | k |
| **b** | E | h | l |
| **b** | F | i | k |
| **b** | G | j | l |
| **c** | E | h | k |
| **c** | F | i | l |
| **c** | G | j | k |
| **d** | E | h | l |
| **d** | F | i | k |
| **d** | G | j | l |

This step checks whether the occurrence of all parameters satisfies $\boldsymbol{Min\_occ_i}$ value.

The above test cases successfully satisfy the condition and the next step is to compute the distinct pairs from the above set of test cases. Some of the distinct pairs are presented below. ae, af, ag, be, bf, bh, bk, ce, cf, ch, ck, ci, etc.

These distinct pairs are passed into the second phase, in order to obtain unique test cases. The occurrence frequency of each pair is computed. The most occurring value pair of the parameters is replaced with the least occurring pair. By following this method, the final test set is presented below.

**Table- II: Final Test Set**

| *P1* | *P2* | *P3* | *P4* |
|---|---|---|---|
| **a** | e | h | k |
| **b** | f | i | l |
| **c** | g | j | l |
| **d** | e | h | k |
| **a** | e | i | l |
| **b** | g | j | l |
| **c** | f | i | k |

Thus, the unique set of test cases with high quality is produced and the results are

presented in table 2. Besides this, the proposed work produces only limited number of test cases.

## IV. PERFORMANCE EVALUATION

Before The empirical evaluation is carried out by implementing the algorithm in Java language. The general flow of the system starts by reading the parameters with different count of values. The minimal, unique and optimal test cases are formed by the proposed algorithm and the experimental results are presented below.

**Table- III: Performance analysis**

| Parameters | Values | GAPTS | AETG | Pairtest | All Pairs | QICT | GA PSO |
|---|---|---|---|---|---|---|---|
| **4** | 11 | 12 | n/a | n/a | 12 | 12 | 11 |
| **4** | 12 | 9 | 9 | 9 | 10 | 11 | 8 |
| **13** | 39 | 15 | 15 | 19 | 22 | 22 | 16 |
| **61** | 339 | 35 | 41 | 36 | 41 | 42 | 33 |
| **75** | 291 | 27 | 28 | 29 | 30 | 34 | 21 |
| **100** | 200 | 10 | 10 | 15 | 16 | 16 | 09 |
| **20** | 200 | 196 | 194 | 218 | 664 | 219 | 182 |

The above given table presents the total count of possible pairs. For instance, T1 has 4 parameters and 11 values. The parameters can be represented by P1, P2, P3, P4 and the total count of values is 11. Here, any parameter can take any number of values. Let P1, P2, P3 and P4 be (A, B, C, D), (E,F), (G,H,I) and (J,K) respectively. The performance of the proposed work is evaluated with respect to the count and quality of test cases. The experimental outcome of the proposed work is compared with the existing works such as QICT, Allpairs, Pairtest, AETG and GAPTS [17-21]. QICT is a command line utility implemented in C# language. Allpairs is also a command line utility and is implemented in Perl language. Pairtest is implemented in Java which generates test case by considering each parameter at a time. The results of AETG and Pairtest were taken from [19] and so the results are not available for test case T1. The time consumption for test case generation of GAPSO is comparatively lesser than the other methods. From the experimental results, it is evident that the GAPSO surpasses the comparative algorithms.The performance statistics of different methods like GAPTS, AETG, pairtest, Allpairs etc. are shown in figure 1.

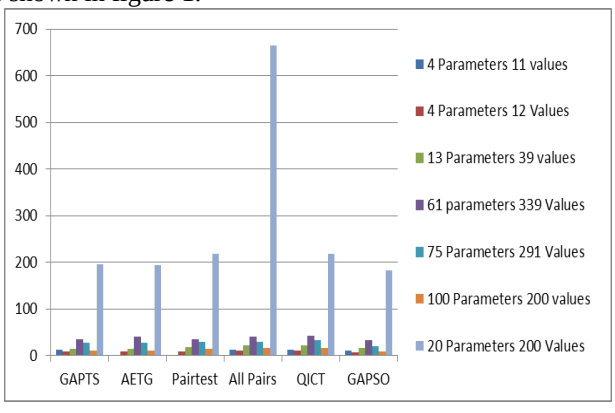


**Fig. 1.Performance statistics of different methods**

## V. CONCLUSION

This paper presents an algorithm for test case generation for achieving pairwise testing by exploiting Genetic and Particle Swarm Optimization algorithm. The proposed algorithm is efficient in terms of cost, time and quality. The test cases are refined for achieving high quality. The experimental results prove that the proposed algorithm generates minimal count of high quality test cases in shorter span of time.

## REFERENCES


1. G. O. P. Ammann and J. Offutt. Introduction to Software Testing. Cambridge University Press, 2008.
2. Maha alzabidi et. al., "Automatic software structural testing by using evolutionary algorithms for test data generations", International Journal of Computer science and Network Security 9 (4), 2009, pp. 390 – 395.
3. Praveen Ranjan Srivastava and Tai-hoon Kim, "Application of genetic algorithm in software testing" International Journal of software Engineering and its Applications, 3(4), 2009, pp.87 – 96.
4. Timo Mantere, "Automatic software testing by Genetic Algorithms" Phd thesis, University of Vaasa, Finland, 2003.
5. Velur Rajappa et. al., "Efficient software test case generation Using genetic algorithm based graph theory" International conference on emerging trends in Engineering and Technology, IEEE, 2008, pp. 298 - 303.
6. D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: an approach to testing based on combinatorial design," IEEE Transactions on Software Engineering, vol. 23, no. 7, pp. 437-443, 1997.
7. J.D. McCaffrey, "Pairwise Testing with QICT", Microsoft Developer Network Magazine, September 2009, vol. 24, no. 9, (in press).
8. B. S. Ahmed, K. Z. Zamli, and C. P. Lim, "Application of particle swarm optimization to uniform and variable strength covering array construction," Applied Soft Computing, vol. 12, no. 4, pp. 1330-1347, 2012.
9. S. A. Ghazi and M. A. Ahmed, "Pair-wise test coverage using genetic algorithms," in Proceedings of the 2003 Congress on Evolutionary Computation (CEC'03), Canberra, Australia, 2003, pp. 1420-1424.
10. P. Bansal, S. Sabharwal, S. Malik, V. Arora, and V. Kumar, "An approach to test set generation for pair-wise testing using genetic algorithms," in Proceedings 5th International Symposium on Search Based Software Engineering (SSBSE2013), St. Petersburg, Russia, 2013, pp. 294-299.
11. T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Hong Kong, 2004, pp. 72-77.
12. D. E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, 1st ed. Reading, MA: Addison-Wesley, 1989, pp. 1-22.
13. J. D. McCaffrey, "An empirical study of pairwise test set generation using a genetic algorithm," in Proceedings of 7th International Conference on Information Technology: New Generations (ITNG), Las Vegas, NV, 2010, pp. 992-997.
14. T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04), Hong Kong, 2004, pp. 72-77.
15. L. Gonzalez-Hernandez, N. Rangel-Valdez, and J. Torres-Jimenez, "Construction of mixed covering arrays of strengths 2 through 6 using a tabu search approach," Discrete Mathematics, Algorithms and Applications, vol. 4, no. 3, 2012.
16. J. Kennedy and R. Eberhart. Particle Swarm Optimization. In proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942-1948.
17. J.D. McCaffrey, "Pairwise Testing with QICT", Microsoft Developer Network Magazine, September 2009, vol. 24, no. 9, (in press).
18. J. Bach and P. Shroeder, "Pairwise Testing: A Best Practice that Isn't", Proceedings of the 22nd Pacific Northwest Software Quality Conference, October 2004, pp. 180–196.
19. Yu Lei and K.C. Tai, "In-Parameter-Order: A Test Generation Strategy for Pairwise Testing", Proceedings of Third IEEE International High-Assurance Systems Engineering Symposium, Nov. 1998, pp. 254-261.
20. D.M. Cohen, S.R. Dalal, M.L. Fredman, and G.C. Patton, "The AETG System: An Approach to Testing Based on Combinatorial Design", IEEE Transactions on Software Engineering, vol. 23, no. 7, 1997, pp. 437-444.

21. James D. McCaffrey, "Generation of Pairwise Test Sets using a Genetic Algorithm", 33rd Annual IEEE International Computer Software and Applications Conference, pp. 626- 631, 2009.
22. M. Lakshmi Prasad and Dr.JKR Sastry, "Generating Test cases for Testing WEB sites through Neural Networks and Input Pairs," International Journal of Applied Engineering research, Vol. No.9, Issue.No.22, 2014.
23. M. Lakshmi Prasad and Dr.V. Chandra Prakash, "Generation of less number of pair wise test cases using artificial neural networks (ANN)," International Journal of Applied Engineering research, Vol. No.9, Issue.No.22, 2014.
24. M. Lakshmi Prasad and Dr.JKR Sastry, "A Comprehensive Survey on Combinatorial Testing," PONTE Journal, Vol.73 Issue No.2, pp.no.187-261, Oct 2017.
25. M. Lakshmi Prasad and Dr. JKR Sastry, "A Graph Based Strategy (GBS) For Generating Test Cases Meant for Testing Embedded Systems Using Combinatorial Approaches," Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 01-Special Issue, pp.no.314-324, Jan 2018.
26. Dr. J. SasiBhanu M. Lakshmi Prasad and Dr. JKR Sastry, "Testing Embedded systems using a Graph Based Combinatorial Method (GBCM)," Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 07-Special Issue, pp.no.355-375, June 2018.
27. M. Lakshmi Prasad and Dr. JKR Sastry, "Building Test Cases by Particle Swarm Optimization (PSO) For Multi Output Domain Embedded Systems Using Combinatorial Techniques," Journal of Advanced Research in Dynamical and Control Systems, Vol. 06-Special Issue, pp.no.1221-1229, May 2018.
28. Dr. J. SasiBhanu, M. Lakshmi Prasad and Dr. JKR Sastry, "A Combinatorial Particle Swarm Optimization (PSO) for Testing an Embedded Systems," Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 07-Special Issue, pp.no.321-336, June 2018.
29. M. Lakshmi Prasad and Dr. JKR Sastry, "A Neural Network Based Strategy (NNBS) For Automated Construction of Test Cases for Testing an Embedded System Using Combinatorial Techniques," International Journal of Engineering Technology Vol. No.7 Issue No.1, pp.no.74-81, Jan 2018.
30. Dr.J.SasiBhanu, M. Lakshmi Prasad and Dr. JKR Sastry, "Combinatorial Neural Network Based Testing of an Embedded System," Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 07-Special Issue, pp.no.605-616, June 2018.
31. M. Lakshmi Prasad and Dr. JKR Sastry, "Generation of Test Cases Using Combinatorial Methods Based Multi-Output Domain of an Embedded System through the Process of Optimal Selection," International Journal of Pure and Applied Mathematics, Volume 118 No. 20, pp.no.181-189, March 2018.
32. M. Lakshmi Prasad and Dr. JKR Sastry, "Testing Embedded System through Optimal Combinatorial Mining Technique," Journal of Advanced Research in Dynamical and Control Systems, Vol. 10, 07-Special Issue, pp.no.337-354, June 2018.
33. Dr. JKR Sastry and M. Lakshmi Prasad, "Testing Embedded System through Optimal Mining Technique (OMT) Based on Multi-Input Domain," International Journal of Electrical and Computer Engineering Vol. No.9 Issue No.3, pp.no.2141-2150, 2019.
34. M. Lakshmi Prasad and Dr. JKR Sastry, "Testing Embedded Systems using test cases generated through Combinatorial Methods," International Journal of Engineering Technology Vol. No.7 Issue No.1, pp.no.146-158, March 2018.

## AUTHORS PROFILE

Dr. M.Lakshmi Prasad is working as an Assistant Professor in the department of CSE in N.B.K.R Institute of Science & Technology, Nellore, and Andhra Pradesh. He has completed M.E. under Anna university and completed Ph.D. in KL University, Guntur, AP. He has done B-Tech in NBKR Institute of Science & technology, Nellore.

Dr. A. Rajasekhar Reddy working as a Professor in the department of CSE in N.B.K.R Institute of Science & Technology, Nellore, Andhra Pradesh. He has done Ph.D. under Sri Krishnadevaraya university, Ananthapur.

Dr JKR. Sastry is a double doctorate in the disciplines of management, computer science and engineering and also holds 3 Post graduate degrees in Electronics and communication engineering, Management and Applied Statistics. He has served K L University as Professor in 4 different disciplines, Dean (R&D), and Dean (P&D) and presently serving as Professor of Electronics and Computer Engineering.