International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 6 Issue 7, July-2017, Impact Factor: 3.578

Comparative Parameter Estimation of Cocomo II Using Tabu Search and Simulated Annealing Techniques

Dimpi Saini[1], Kamna Solanki[2]

[1]Research Scholar, University Institute of Engineering &Technology, MDU, Rohtak
[2]Assistant Professor, University Institute of Engineering &Technology, MDU, Rohtak

ABSTRACT

Effective cost estimation is one of the major activity in the field of software engineering and invites a lot of research. Cost estimation process involves a series of systematic steps that provide estimate with acceptable risk. The COCOMO 2 models predicts software development effort in Person Months (PM) and project duration in months. Using simulated annealing and Taboo Search techniques, the parameters of COCOMO 2 model will be estimated. We compare the results of simulated annealing over this scenario with Taboo Search technique. The Simulated Annealing algorithm is used to handle these models and will show potential advantages in solving the problem. This technique will minimize functions of various variables. This technique will be applied to arbitrary combinatorial problems. Taboo search is based on introducing flexible memory structures in conjunction with strategic restrictions and aspiration levels as a means for exploiting search spaces. Meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimum by use of a Taboo list. The proposed model will compare the efficiency of both these algorithms and tend to reduce the uncertainty of COCOMO II post architecture model coefficients i.e. a, b, c and d.

Index Terms: COCOMO, COCOMO II, Simulated annealing, PM, TDEV

1. INTRODUCTION

With the growing advancement of technology, the importance of software has increased which leads to a prominent issue of software cost estimation. The cost of any software product is the most difficult task which is critical for its customers, developers and users. This will in turn affect the total software project management process including scheduling, resource allocation and project planning. The objective of software cost estimation method is to estimating the cost and effort required for software production. Effort cost includes overhead costs where they take the total cost of running the organisation and divide this by the number of productive staff. The Taboo search begins by leading to a local minima. To avoid retracing the steps used, the method records recent moves in one or more Taboo lists. The Taboo lists are historical in nature and form the Taboo search memory. The role of the memory can change as the algorithm proceeds. At initialization the goal is make a coarse examination of the solution space known as diversification. In many cases the differences between the various implementations of the Taboo method have to do with the size, variability and adaptability of the Taboo memory to a particular problem domain. The Taboo search has traditionally been used on combinatorial optimization problems. The technique is straightforwardly applied to continuous functions by choosing a discrete encoding of the problem. Many of the applications in the literature involve integer programming problems, scheduling, routing, travelling salesman and related problems. There are three parameters involved in computing the total cost of a software development project:

- Hardware and software costs including maintenance
- Travel and training costs
- Effort costs (the costs of paying software engineers).

Software cost estimation is process of predicting the effort required to develop a software engineering project. This process becomes one of the biggest challenges and most expensive component in the field of software. While the software cost estimation may be simple in concept, it is difficult and complex in reality. The accurate software estimation can provide good support for decision-making process like accurate assessment of costs can help the organization to better analyse the project and effectively manage software development process. Software development projects are disreputable for being completed delayed and over budget and for often failing to satisfy user needs. A myriad of value estimation models are projected to predict development costs early within the lifecycle with the hope of managing the project well within time and budget. However, studies have reported rather high error rates of prediction even within the case of the well-established and wide acknowledged models.

This study focuses on the improvement and fine-tuning of the COCOMO eighty one model. Though this model is primarily based on code development practices that were prevailing within the 80s, its wide use in trade and world, the easy kind of the constant equations and also the accessibility of the info in an internet repository build it attractive as a test-bed for further analysis in code development value estimation. During this study, we show how the recently developed Computational Intelligence techniques will effectively improve the prediction power of existing models. In explicit, we specialize in the difference of the Multi-Objective Particle Swarm Optimization (MOPSO) formula in at the same time minimizing 2 objective functions – prediction error rate and model quality. This provides the project manager with a chance to decide on from a group of best solutions drawn in an exceedingly trade-off relationship on the economist front. Ancient search algorithms use data of the terrain and knowledgeable heuristics to guide the search; such uses build them problem-specific and domain-dependent. The MOPSO meta-heuristic approach depends neither on the data of the problem nor on the experts' heuristics, creating its application wide and in depth.

Introduction to COCOMO model

Cost Constructive Model is the most widely used software estimation model in the world. This is the first methods to estimate software effort automatically, where in its simplest form effort is expressed as a function of anticipated size as [3]:

$$E = aS^b \qquad (1)$$

Where E is the effort required, S is the anticipated size, a and b are domain specific parameters to estimate the number of Person-Months required developing a project. Most of the other COCOMO results including the estimates for Requirements and Maintenance are derived from this quantity.

COCOMO is defined in terms of three different models:

- **The Basic Model:-** The basic COCOMO model computes software development effort and cost as a function of program size expressed in estimated lines of code (LOC).The Basic Model estimates the required effort [4].

$$SM = a * (KLOC)^b \qquad (2)$$
$$TDEV = c * (SM)^d \qquad (3)$$

  Where Staff-Months is **SM,** Time of Develop is **TDEV**

- **The Intermediate Model:-** The intermediate model computes software development effort as a function of program size and set of cost drivers that include subjective assessments of product, hardware, personnel and project attributes [4].

$$SM = EAF * a * (KLOC)^b \qquad (4)$$

- **The detailed Model:** - This model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step i.e. Analysis and design of the software engineering process.

SCALE FACTORS

The application size exponent is aggregated of five scale factors (SF) that describe relative economies or diseconomies of scale that are encountered for the software projects of dissimilar magnitude.

Five Scale factors are [13]:

- Precedentedness (PREC) - Reflects the previous experience of the Organization.
- Development Flexibility (FLEX) - Reflects degree of flexibility in Development process.
- Risk Resolution (RESL) - Reflects the extent of risk analysis carried out.
- Team Cohesion (TEAM) - Reflects how well development team knows each other and work together.
- Process Maturity (PMAT) - Reflects the process maturity of the organization.

Table 1: Rating of scale factors

Scaling Factors	Very low	Low	Nominal	High	Very high	Extra high
PREC	6.20	4.96	3.72	2.48	1.24	0.00
FLEX	5.07	4.05	3.04	2.03	1.01	0.00
RESL	7.07	5.65	4.24	2.83	1.41	0.00
TEAM	5.48	4.38	3.29	2.19	1.10	0.00
PMAT	7.80	6.24	4.68	3.12	1.56	0.00

2. LITERATURE SURVEY

Astha Dhiman [1] described that COCOMO II was the most commonly used model because of its simplicity for estimating the effort in person-month for a project at the different stages. Today's effort estimation models were based on soft computing techniques as neural network, genetic algorithm, the fuzzy logic modelling etc. for finding the accurate predictive software development effort and time estimation. As there were no clear guideline for designing neural networks approach and also fuzzy approach is hard to use. Genetic Algorithm can offer some significant improvements in accuracy and has the potential to be a valid additional tool for software effort estimation. This work aims to propose a genetic algorithm for optimizing current coefficients of COCOMO II model to achieve more accuracy in estimation of software development effort.

Praveen Ranjan Srivastava [2] suggested that when to stop testing and release the developed software is the one of the most important questions faced by the software industry today. Software testing is a crucial part of the Software Development Life Cycle. The number of faults found and fixed during the testing phase can considerably improve the quality of a software product thereby increasing its probability of success in the market. Deciding the time of allocation for testing phase is an important activity of quality assurance. Extending or reducing this testing time depending on the errors uncovered in the software components can profoundly affect the overall project success. Since testing software incurs considerable project cost over-testing the project can lead to higher expenditure while inadequate testing can leave major bugs undetected thereby risking the project quality. Hence prioritizing the components for testing is essential to achieve the optimal testing performance in the allotted test time. This paper presents a Test Point Analysis based Module Priority approach to determine the optimal time to stop testing and release the software.

N´estor R. Barraza [3] suggested that the parameter estimation for the Compound Poisson Software Reliability Model is analyzed. The biased characteristic is considered in order to get better performance. A comparison with the well-known Non Homogeneous Software Reliability Models is presented. Several experimental data are used in order to analyze the goodness of fit of both models. The main disadvantage of this model is it cannot be used for a long time prediction, the failure rate needs to be updated time to time.

Mr. Karambir [4] suggested that the reliability of web applications is more difficult to measure and improve because the large system has highly distributed nature. Hardware faults can be easily predicted rather than the software faults. The customer required more security and accuracy in web applications. So the reliability of web applications is important to consider over different kind of network. The reliability of web applications is more complex rather than the other software systems. In this paper the author will use the Goel Okumoto SRGM to detect the number of faults in a specified time and estimate its reliability in regard of web applications. The rate of change is calculated by executing the test cases for actual

defects per day. The Goel-Okumoto uses exponential distribution to predict the number of faults in web applications. The author will assess the software reliability of web applications by using SRGM. This work do not predict the reliability of web applications which is a limitation to this proposed work.

Lilly Florence [5] described that in common parlance the traditional software reliability estimation methods often rely on assumptions like statistical distributions that are often dubious and unrealistic. The ability to predict the number of faults during development phase and a proper testing process helps in specifying timely release of software and efficient management of project resources. In the Present Study Enhancement and Comparison of Ant Colony Optimization Methods for Software Reliability Models are studied and the estimation accuracy was calculated. The Enhanced method shows significant advantages in finding the goodness of fit for software reliability model such as finite and infinite failure Poisson model and binomial models.

S. Aloka et. al. [6] described that test effort estimation is an important activity in software development because on the basis of effort cost and time required for testing can be calculated. Various models are available for estimating effort but to some extent all models result in erroneous effort estimation. So there is a need to optimize the effort estimated. Meta heuristic techniques can be used for this purpose, to optimize a problem by iteratively trying to improve a solution, using some computational methods. Particle Swarm Optimization is one such technique which have been incorporated in this work to get good test effort estimates.

Iman Attarzadeh [7] proposed a novel Constructive Cost Model (COCOMO) based on soft computing approach for software cost estimation. This model carried some of the desirable features of neural networks approach, such as learning ability and good interpretability, while maintaining the merits of the COCOMO model. The proposed model could be interpreted and validated by experts and has good generalization capability. The model deals effectively with imprecise and uncertain input and enhanced the reliability of software cost estimates.

Ma Junhai and MU Lingling[8] reviewed the Cost estimation methods and Constructive Cost Model (COCOMO) was further discussed. Considering the variations and uncertainty to calculate LOC (Lines of Code) in Constructive Cost Model we used Function Point (FP) estimation to make up this deficiency which can improve the model accuracy. Finally through comparing and analysing these estimation methods by a case study the feasibility of the improved method was proved.

3. IMPLEMENTATION MODEL AND RESULT ANALYSIS

Proposed Model for Taboo Search

To use the concept of Taboo search algorithm to optimize the COCOMO II model coefficients to achieve accurate software effort estimation. The proposed model will tend to reduce the uncertainty of COCOMO II post architecture model coefficients i.e. a, b, c and d using genetic algorithm. This will also evaluate the predicted effort value as accurate to the given real effort value. Taboo search will increase the efficiency of output values of parameters of COCOMO II model rather than genetic algorithm. The dataset required for proposed model is collected from Turkish Software Industry project data and the Industry dataset. Each record contains information about the completed software project and the record will be described with two attributes i.e. Size (LOC, KLOC) and the actual development effort (Person Month/Man Month). The actual effort and predicted effort using COCOMO II model will be given in dataset.

Execution Model for Taboo Search

In the Taboo Search method in order to improve the efficiency of the exploration process, some historical information related to the evolution of the search is kept basically the itinerary through the solutions visited. Such an information will be used to guide the search from one solution to the next one avoiding cycling. This is one of the most important features of this algorithm. Given an instance x, the algorithm starts from an initial solution typically a random one. At any iteration it has to find a new solution by making local movements over the current solution. The next solution is the best among all possible solutions in the neighborhood. To carry out the exploration process, recently visited solutions should be avoided. To this aim a Taboo list is maintained. Therefore once a solution is visited the movement from which it was obtained is considered Taboo. In a certain case there is a dynamic neighborhood present as compared to the previous local search algorithms. Typically there are two kinds of Taboo lists, a long term memory maintaining the history through all the exploration process as a whole and a short term memory to keep the most recently visited Taboo movements. A movement with a Taboo status (Taboo movement) is avoided to be applied, unless it satisfies certain aspiration criteria. This aims to avoid falling into local optima. Some typical stopping conditions are as follows: when neighbourhood solution is empty, the

maximum number of solutions to be explored is fixed, the number of iterations since the last improvement is larger than a specified number the total number of iterations of the TS algorithm is fixed.

Execution Model for Simulated Annealing

Consider one chromosome. Choose a best fitness value initially. If the value of fitness function is low that will be considered as the best optimum solution i.e. user is more near to actual value. Consider a specific chromosome and calculate the fitness of that chromosome and get it checked with the maximum fitness if new result is better than accept otherwise this is a worst solution and check the temp. Anneal function is created to perform this function.

If the temperature is high probability of accepting worst solution is high. Worst sol can be accepted. Max probability of accepting worst sol. 0.3 i.e. 30%. When the temperature is at peak value. Temperature decrease in each iteration and probability also reduces. If the temp reaches 0 probability also reaches to level 0. This is performed in each iteration. GA always deny to accept worst sol. But in this technique the worst sol also have the possibility to be accepted. After many iterations we get the best result. This technique is better than previous as this technique also accepts worst sol. Which will in turn responsible for the best solution**.**

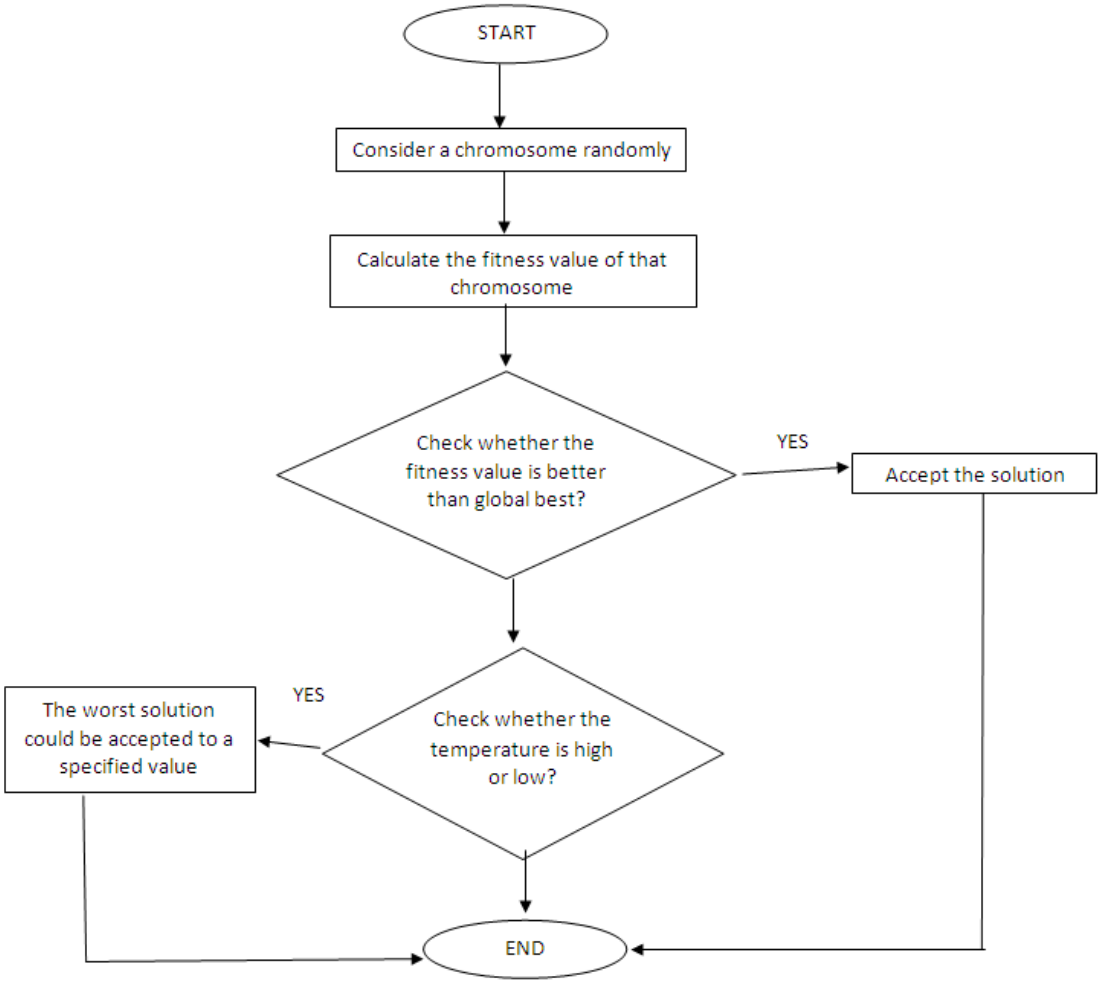

Fig. 1: Flowchart for Simulated Annealing Algorithm

Tools Used and Techniques

Net Beans

The Net Beans IDE is open source and is written in the Java programming language. It provides the services common to creating desktop applications - such as window and menu management, settings storage -and is also the first IDE to fully

support JDK 5.0 features. The Net Beans platform and IDE are free for commercial and non- commercial use, and they are supported by Sun Microsystems.

Using Net Beans

To be able to successfully build programs it is recommended to follow the intended procedure, described here. First it's important to create a new project. Various project types are available however in this case the intended type will be Java Application.

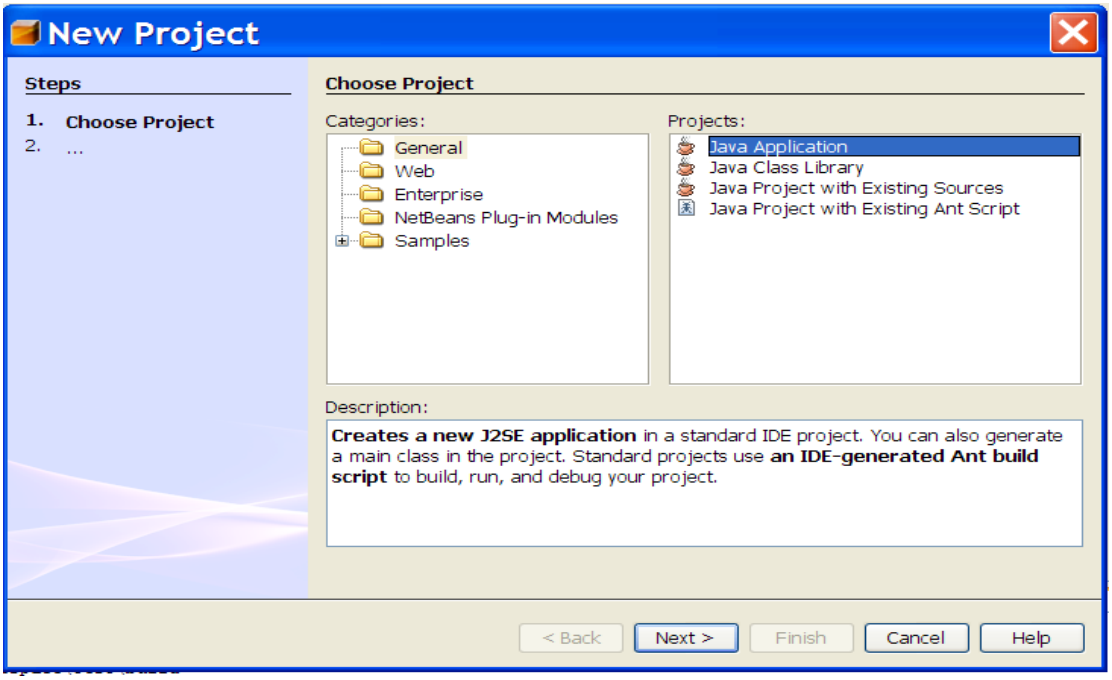

Figure 2: Choosing a Project type

The Net Beans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM. The Net Beans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the Net Beans Platform including the Net Beans IDE itself which can be extended by third party developers. The Net Beans IDE bundle for Java SE contains what is needed to start developing Net Beans plugging and Net Beans Platform based applications; no additional SDK is required. Applications can install modules dynamically.

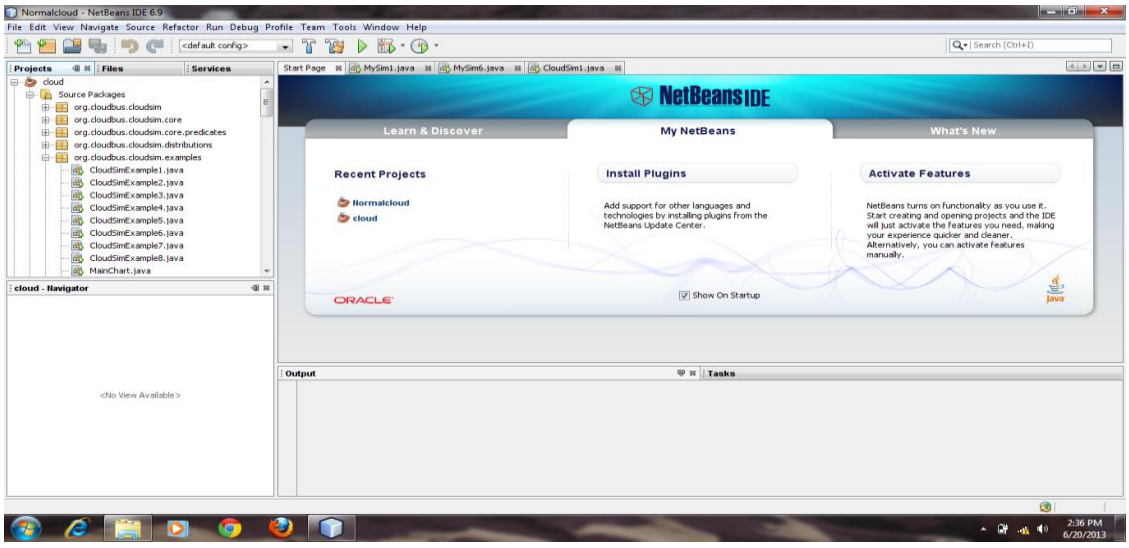

Fig 3: Net Beans IDE 6.9

Any application can include the Update Center module to allow users of the application to download digitally signed upgrades and new features directly into the running application. Reinstalling an upgrade or a new release does not force users to download the entire application again. The platform offers reusable services common to desktop applications allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management
- Window management
- Wizard framework
- Net Beans Visual Library
- Integrated development tools

Net Beans IDE is a free, open-source, cross-platform IDE with built-in-support for Java Programming Language. Net Beans IDE supports development of all Java application types Java SE including Java FX, Java ME, web, EJB and mobile applications out of the box. Among other features are an Ant-based project system, Maven support, refactoring, and version control supporting CVS, Subversion, Mercurial and Clear case.

All the functions of the IDE are provided by modules. Each module provides a well defined function such as support for the Java language, editing, or support for the CVS versioning system, and SVN. Net Beans contains all the modules needed for Java development in a single download allowing the user to start working immediately. Modules also allow Net Beans to be extended. New features such as support for other programming languages can be added by installing additional modules.

CONCLUSION

To use the concept of particle swarm optimization to optimize the COCOMO II model coefficients is to achieve accurate software effort estimation. To reduce the uncertainty of COCOMO II post architecture model coefficients i.e. a, b, c and d using Particle Swarm Optimization and evaluate the predicted effort value as accurate to the given real effort value. Computational Effort of SA is better than simple Genetic algorithm. SA performs the GA with a larger differential in computational efficiency when used to solve unconstrained nonlinear problems with continuous design variables and less efficiency differential when applied to constrained nonlinear problems with continuous or discrete design variables.

REFERENCES

[1]. Jyoti Mahajan and Simmi Dutta “COREAN: A proposed Model for Predicting Effort Estimation having Reuse” IJSCE ISSN: 2231-2307, Volume-2, Issue-6, January 2013
[2]. Divya Kashyap and A. K. Mishra “Software Cost Estimation Using Particle Swarm Optimization In The Light Of Quality Function Deployment Technique”, in proceedings of International Conference on Computer Communication and Informatics, IEEE, pp. 1-8, 2013
[3]. Srinivasa Rao T., Hari CH.V.M.K. and Prasad Reddy “Predictive and Stochastic Approach for Software Effort Estimation”, in International Journal of Software Engineering, IJSE, pp. 86-115, 2013
[4]. D. Manikavelan and Dr. R. Ponnusamy, “To find the accurate software cost estimation using Differential Evaluation algorithm”, in proceedings of the IEEE International Conference on Computational Intelligence and Computing Research, IEEE, pp. 1-4, 2013
[5]. Astha Dhiman and Chander Diwaker “ Optimization of COCOMO II Effort Estimation using Genetic Algorithm” AIJRSTEM 13-278; 2013
[6]. Praveen Ranjan Srivastava, Subrahmanyan Sankaran, Pushkar Pandey, “Optimal Software Release Policy Approach Using Test Point Analysis and Module Prioritization” MIS Review Vol. 18, No. 2, March 2013
[7]. N´estor R. Barraza “Parameter Estimation for the Compound Poisson Software Reliability Model” International Journal of Software Engineering and Its Applications V o l. 7, No. 1, January, 2013
[8]. Mr. Karambir and Jyoti Tamak “Use of Software Reliability Growth model to Estimate the Reliability of Web Applications” International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 6, June 2013
[9]. Lilly Florence andLatha Shanmugam “Enhancement And Comparison Of Ant Colony Optimization For Software Reliability Models” Journal of Computer Science 9 (9): 1232-1240, 2013
[10]. Iman Attarzadeh and Siew Hock Ow “Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model”, in proceedings of the IEEE International Conference on Fuzzy Systems, IEEE, pp. 2458-2464, 2011

[11]. Tirimula Rao Benala, Satchidananda Dehuri, Suresh Chandra Satapathy and Ch. SudhaRaghavi "Genetic Algorithm for Optimizing Neural Network Based Software Cost Estimation", Lecture Notes in Computer Science Volume 7076, 2011

[12]. Ekrem Kocaeli and Tim Menzies described in the research "How to Find Relevant Data for Effort Estimation?", in proceedings of the International Symposium on Empirical Software Engineering and Measurement, ESEM, pp. 255-264, 2011

[13]. Alejandra Yepez Lopez and Nan Niu the research work "Multiple Criteria Decision Support for Software Reuse: A Case Study", in proceedings of the International Conference on IRI, IEEE, pp. 200-205, 2011

[14]. S. Aloka, Peenu Singh, Geetanjali Rakshit, and Praveen Ranjan Srivastava "Test Effort Estimation-Particle Swarm Optimization Based Approach" Springer-Verlag Berlin Heidelberg, pp. 463–474, 2011

[15]. Iman Attarzadeh, Siew Hock Ow "A Novel Soft Computing Model to Increase the Accuracy of Software Development Cost Estimation" IEEE, volume 3, pp – 603-607, 2010

[16]. Ma Junhai and M.U. Lingling "Comparison Study on methods of software cost Estimation" in proceedings of the International Conference on EBISS,IEEE, pp. 1-4, 2010

[17]. Tad Gonsalves , Kei Yamagishi , Ryo Kawabata and Kiyoshi Itoh "Optimizing software development cost estimates using Multi objective Particle Swarm Optimization" in International Publisher of Progressive Academic Research Books and Journals, IGI, pp. 45-47, 2010

[18]. Taeho Lee, Donoh Choi and Jongmoon Baik, "Empirical Study on Enhancing the Accuracy of Software Cost Estimation Model for Defense Software Development Project Applications" ICACT, ISBN 978-89-5519-146-2, pp. 1117-1119, 2010

[19]. Maged A. Yahya, Rodina Ahmad and Sai Peck Lee "Effects of Software Process Maturity on COCOMO II's Effort Estimation from CMMI Perspective", in IEEE Transaction on Software Engineering, IEEE, pp. 255-262, 2008

[20]. Shih-Wei Lina,b , Zne-Jung Lee b, Shih-Chieh Chenc and Tsung-Yuan Tseng "Parameter determination of support vector machine and feature selection using simulated annealing approach" Applied Soft Computing, 2008, pp: 1505–1512,