Application of genetic algorithm designed for software effort prediction

G. Vijay Raj, Tadi Srinivas, Ch. Satyananda Reddy
Department of Computer Science and Systems Engineering,
Andhra University, Visakhapatnam.
abburaog@gmail.com, tadisrinivasarthamuru@gmail.com,
satyanandau@yahoo.com

Abstract

Software cost estimation of project is an essential step in software project management as it is used to predict the effort and time needed to complete the project. Hence the accurate estimation of the effort required for software development is necessary. COCOMO II is one of the best-known models for software cost estimation. The coefficients of COCOMO II are used in the prediction of effort, but for the accurate prediction these coefficients should have an optimized value. There are many works done on optimization of these coefficients by using Genetic Algorithms. In this paper, it is proposed an improved Genetic Algorithm to optimize the coefficients of COCOMO II and thereby improve the prediction of the effort required for software development. Experiments were conducted to compare the accuracy of the optimized coefficient with standard COCOMO II dataset. The accuracy of the model is measured by using Mean Magnitude of Relative Error (MMRE). These optimized coefficients are used to predict more accurate effort than the actual coefficients of COCOMO II.

Keywords: COCOMO II, Genetic algorithms, evaluation metrics.

1. Introduction

As the software industry evolving, management of software development and quality of the software development becomes essential. Hence, now a day's software development becomes more expensive and also more demandable. So, software cost estimation becomes an important step in software development. Software Cost Estimation (SCE) is the estimation of the effort, cost and personnel needed in one month (PM) for a software development. Good software estimation can provide great support for the decision making process such as predicting accurate assessment of costs and efforts can help to effectively manage the software development process by reducing the risk.

Effort Estimation may be a realistic forecast of the required effort to construct and maintain software. Bad estimation of software effort will lower the efficiency of the project and it's going to be a waste of company funds and can make failing results during the project. Impact of the failure or success of software projects depends on software effort estimation. Software engineering community has provided many models to solve this problem such as SLIM, COCOMO. After some updating COCOMO II is developed, using various parameters in COCOMO II. But these parameters could differ from organization to organization. There is a need to optimize the parameters to get the optimal results. For this optimization we used Genetic Algorithm (GA).

Genetic algorithms are optimization algorithms, developed by scientist named John Holland in 1975. It's a natural heuristic algorithm that's used to find the precise and approximate solutions. Genetic Algorithm is based on the iterative improvement of the present solution. It can also give solution set rather than one solution. This Genetic Algorithm used in a large scale for predictive models. Also for the software cost estimation Genetic algorithms are used.

In this paper we have optimized the COCOMO II coefficient values by calculating fitness, taking the average for each chromosome and applying Genetic algorithm to it. In Section2 of this paper, it is discussed the background of the work, Section 3 contains the related work , section 4 contains the proposed model, Section 5 contains the experiment part, Section 6 contains results and analysis, Section 7 is concluded with the advantages and disadvantages of the proposed model.

2. Background

COCOMO II Model:

In 1981, Barry Boehm developed an algorithmic model for software cost estimation named COCOMO. COCOMO uses predefined dataset and current project characteristics to predict a software cost for a new model. This model is also called as COCOMO 81. The COCOMO 81 model was modified to new version named COCOMO II in 1995. The PA (Post Architecture) Model of COCOMO II gains more popularity because it directly involves the actual effort in software development.COCOMO II is essentially three different models:

The Application Composition Model

Reasonable for projects worked with current GUI-manufacturer devices based on new Object Points.

The Early Design Model

You can utilize this model to get approximate estimates of a project's expenses and duration before you've decided it's whole design.
It utilizes a little arrangement of new Cost Drivers, and new assessing conditions in light of Unadjusted Function Points or KSLOC.

The Post-Architecture Model

It is the most detailed COCOMO II model. You'll utilize it after you've built up your venture's general engineering. It has new cost drivers, new line tallying rules, and new conditions.

In this paper, it is used the COCOMO II Post-Architecture Model, as it is a more detailed and more accurate. In COCOMO-II, The Person Months (PM) is calculated, which is an amount of time one person spends on a software development in one month. This number is restrictive of occasions and excursions yet accounts for weekend time off. The quantity of individual months is not quite the same as the time it will take the task to finish; this is called the development schedule.

$$\mathbf{PM = A \times Size^{E} \times \prod_{i=1}^{17} EM_i} \qquad \ldots \mathbf{Equation\ 1}$$

Where A = 2.94

$$\mathbf{E = B + 0.01 \times \sum_{i=1}^{5} SF_i} \qquad \ldots \mathbf{Equation\ 2}$$

Where B = 0.91

There are 17 values of EM in COCOMO II. The size of the software development is in the KLOC.There are 5 values of SF in COCOMO II which are used for calculating the value of E. The A and B values are constant and all the EM and SF are variables.

Equation 1 is used for the cost estimation in Post-Architecture models of COCOMO II. The inputs taken in this model are size of software project, two constant, A and B, and an exponent, E. The size is in units of thousands of source lines of code (KLOC). The scale (or exponential) factor, B, represents the relative economies or diseconomies of scale experienced for software projects of various sizes. The constant, A, is utilized to capture the multiplicative impacts on efforts (PM value) with projects of expanding size.To calculate the PM value, the variables like scale factors (SF) and effort multipliers (EM) is used. The SF are used for calculating the exponent E for the size of project, later this exponent E is used for calculating the PM value. The formulae for both the exponent and the PM were given in equation 1 and 2.

Equation 2 calculates the value exponent, E, used in Equation 1. There the rating levels for the COCOMO II scale drivers. The selection of scale drivers is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. Each scale driver has a scope of rating levels, from Very Low to Extra High. Each evaluating level has a weight, W, and the particular estimation of the weight is known as a scale factor. A task's scale factors, W_i, are added over the entirety of the components, and used to decide a scale exponent, E.

Each scale factors rating level are Very Low (VL), Low (L), Normal (N), High (H), Very High (H), Extra High (H). The 5 SF values have different six values for different rating. Similarly, all the 17 effort multipliers (EM) have these rating levels. These SF values and the EM values are used for calculating the PM value.

Genetic Algorithm:

Genetic algorithm is a guesstimate approach designed by John Holland in 1975. It works on the natural selection process. The basic structure of genetic algorithm is as follows:

1. Chromosome: Chromosome consists of a set of variables known as genes, there may be many genes in a single chromosome, and we use them as a population.

2. Initial Population: The n number of chromosome is generated randomly. The various solution set is set randomly in this step.

3. Fitness Function: The fitness of every generated chromosome is calculated. Also the average fitness of that generation is also measured by using various formulae. The chromosome's having more fitness are taken into consideration.

4. Operations: Operator generates new population of chromosomes on the basis of current population. Operator set is consisting processes such as selection, crossover and mutation. At every generation, the operations are done on the most suitable individuals, i.e. the ones having more fitness are given priority. For this selection, methods like roulette wheel selection, rank based selection are used. Then the crossover is done between them and after that the mutation is done on the new population.

5. Performance Validation: It is used for measuring the performance of algorithm. Magnitude of Relative Error (MRE), Mean MRE (MMRE), Value Adjustment Factor (VAF) and PRED(X) are some of the ways to check the performance.

These are the basic steps of the Genetic Algorithm. The evaluation of these steps may differ, but these steps remain constant in all the Genetic Algorithm. For example, the evaluation of the fitness may be different, as there are many formulae to calculate the fitness value. The best suited formula for fitness will be used for different models.

Similarly, for selection process, there are various ways such as roulette wheel selection, rank based selection, random selection etc. In roulette wheel, the chromosomes having more fitness value have more probability to be selected as parent. In rank based, the chromosomes are arranged in the decreasing order of fitness, and whoever have highest rank are selected. In random selection, as per the name suggests, chromosomes are selected randomly.

Crossover is also can be done in 1-point crossover, multipoint crossover or uniform crossover etc. In this one-point crossover, a random crossover point is selected and the tails of its two parents are swapped to get new off-springs. Multi point crossover is a generalization of the one-point crossover wherein alternating segments are swapped to get new off-springs. In a uniform crossover, we don't divide the chromosome into segments; rather we treat each gene separately. In this, we essentially flip a coin for each chromosome to decide whether or not it'll be included in the off-spring. There are also various types of mutation such as bit flip mutation, swap mutation, inverse mutation.

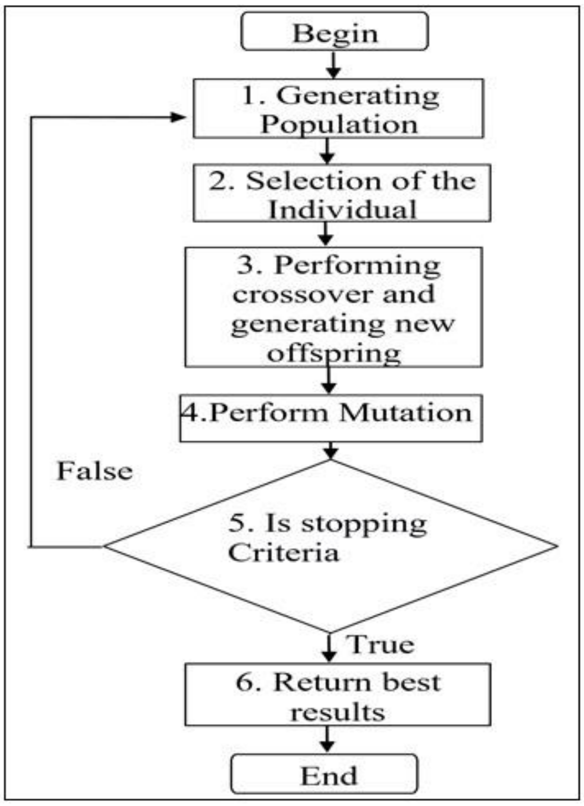

Fig 1: Basic steps of Genetic Algorithm

3. Related Works

A few works in the writing were led attempting to enhance the COCOMO II model (considering different types); coefficients utilizing numerous advancement strategies, for example, Particle Swarm Optimization (PSO), Simulated Annealing (SA), GA, and so on. A portion of the open issues utilized the GA as an optimization method. It moreover presents an overall system on how the GA could be applied to enhance the COCOMO II model coefficients.

In Reference [5], Genetic Algorithm was used to tune the coefficients to optimize organic and semi-detected models of COCOMO 81. The research was done on a data set that contains 20 records for organic model and 19 records for semi-detected mode. The data sets were split into testing and training sets. The results show that the Mean Relative Estimation (MRE) for test dataset by using the optimized coefficients from the Genetic Algorithm was 2.48 times less than the Mean Relative Estimation by using the current COCOMO coefficients for organic model. But for the semi-detected model the result was worse than the original coefficients.

In Reference [6], the GA was used to tune the coefficients to optimize the COCOMO for all models: organic, semi-detected and embedded. The experiments were conducted on NASA 93 COCOMO data set. The result show that for organic and semi-detected model, the estimated development time using the optimized coefficients were better than development time using current COCOMO coefficients and close to the real development time.

In Reference [7], the GA was used for tune the coefficients to optimize the COCOMO model considering effort estimation equation only. Two updates versions of the COCOMO model were provided to think about the result of methodology in effort estimation, so GA was also used for optimizing the parameters for changed models. 18 projects come from NASA information set were considered for the analysis, the dataset was split as follow: 13 projects used for training and 5 for testing. The results show that considering the methodology in effort calculation improves the quantifiable effort, implementation on the same dataset that utilized in Reference [7], the research in Reference [8] proposed a modified model based on COCOMO that also considering the methodology in effort estimation and adding new bias parameter. The GA was used to estimate that new model parameters.

Taking the COCOMO-II model, the Reference [9] projected GA to optimize the COCOMO-II PA model coefficients to improve the model accuracy in estimating the trouble. The experiments conducted on 15 projects from TURKISH and INDUSTRY datasets [10]. With this optimized coefficients, Mean Relative Estimation (MRE) was 1.51 times lesser than the actual coefficients. Thus, the estimated efforts with optimized coefficient were more accurate than the effort calculated by using default coefficient. The work in the References [11] and [12, is more focused on tuning the values of A and B.

4. Proposed Model

4.1. Genetic Algorithm

The proposed genetic algorithm is introduced briefly in this part. The steps in this algorithm are:

1. Generating Initial Population: The GA starts with a randomly generated initial population. Individual chromosomes are generated randomly. All the variables which need to be optimized are taken in a single chromosome. Here, the values of A and B are taken as a single chromosome. The random values of A and B are generated within a given range.

```
xxxx | xxxx
 A      B
```

2. Calculating PM value: The Person Months (PM) is calculated for every chromosome in that generation. We calculate the PM value for every training record in the dataset, for each chromosome.

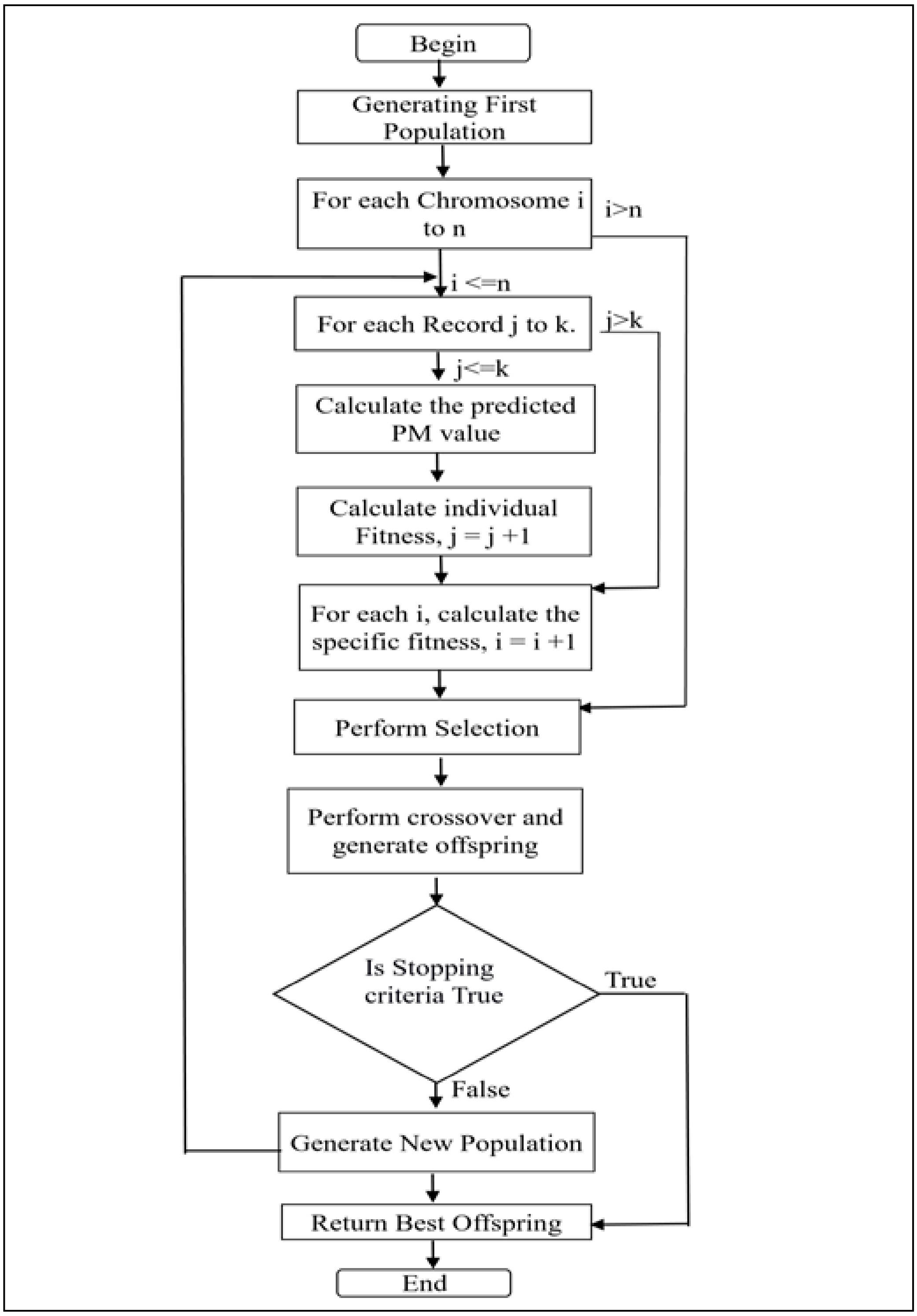
Begin
Generating First Population
For each Chromosome i to n
i>n
i <=n
For each Record j to k.
j>k
j<=k
Calculate the predicted PM value
Calculate individual Fitness, j = j +1
For each i, calculate the specific fitness, i = i +1
Perform Selection
Perform crossover and generate offspring
Is Stopping criteria True
True
False
Generate New Population
Return Best Offspring
End

Fig 2.The proposed algorithm for the optimization of the coefficients

3. Calculate the Fitness Value: The fitness value of every chromosome is measured with respect to its calculated PM. The PM value is calculated on the standard values of A = 2.94 and B = 0.91. The fitness of every record is calculated as:

$$\mathbf{Fitness_{ij}} = \frac{\mathbf{1}}{\mathbf{1 + abs(Calculated\ PM_{ij} - \ Estimated\ PM_{ij}\ )}} \quad \ldots \mathbf{Equation\ 3}$$

Where ***abs*** is absolute value

For every chromosome j the average fitness is calculated taking all the fitness of the training records with respect to chromosome j.

$$\mathbf{Fitness_i} = \sum_{\mathbf{j=1}}^{\mathbf{k}} \mathbf{Fitness_{ij}} \quad \ldots \mathbf{Equation\ 4}$$

Let the number of records are k and number of chromosomes be N. For a chromosome i the fitness is calculated for record j (1 to k) by using equation 3. Then the total fitness of chromosome i is calculated as summation of all the fitness for every record.

4. Selection of Chromosomes: Random based selection algorithm is used for the selection of the parents. The parents are chosen from the chromosomes randomly.

5. Crossover: As the parents are selected, n-point crossover is done on individual values of A and B. The offspring is taken into the consideration for that generation. The offspring’s fitness is calculated by using similar formula. The generations which produce the offspring with most fitness is send as the best results.

6. Generating new population: The new population is also generated randomly, and the offspring is generated by a similar way.

5. Experiments

The proposed model was implemented using Python 3.6. Various Python libraries like Pandas, Numpy, Matplotlib are used. The training dataset is consisting of 15 records. The algorithm is simulated multiple times with the change in number of chromosomes, no. of generations and crossover method. Every other time we get different type of result. 1-point crossover, 2-point crossover and 3-point crossover are used in the model. The algorithm generates the optimized value of A and B alongside the MMRE. The analysis of all these different crossovers alongside the best results for optimization of the value of A and B.

The standard COCOMO II dataset is used. There are 63 records in the given dataset, consisting the size and actual efforts of the records alongside respective SF and EM values. We need to optimize the coefficient A, B with using this dataset. The data is split as 15 records (Table 1) for training and remaining records for testing.

Table 1. COCOMOII Dataset considered

Project ID	Size	Actual Effort
1	3.965	6
2	1.605	7.3
3	1.485	5.9
4	11.25	55
5	12	33
6	21.45	83
7	26.25	106
8	17.25	36
9	22.5	423
10	3.445	14
11	20.8	218
12	24.05	201
13	4.65	8
14	1.95	8
15	67.5	45

6. Result Analysis

The standard values of A and B are 2.94 and 0.91 respectively. We need to tune this value and optimize to reduce the MRE and MMRE.

$$\mathbf{MRE_i} = \frac{\mathbf{abs(Actual\ Effort_i - Estimated\ PM_i)}}{\mathbf{Actual\ Effort_i}} \times \mathbf{100} \qquad \dots \textbf{Equation 5}$$

where abs is absolute value

$$\mathbf{MMRE} = \frac{\sum_{i=1}^{n} \mathbf{MRE_1}}{\mathbf{No.\ of\ Records(63)}} \qquad \dots \textbf{Equation 6}$$

By training for numerous times, the number of chromosomes 800 for each 1000 generations is producing best results. The algorithm optimized the value of A and B with following results.

Table 2. Comparison of COCOMO II model and Proposed Model

Project ID	A	B	PM value by COCOMO	A value by Proposed Model	B value by Proposed Model	PM value by Proposed Model	Actual Effort
1	2.94	0.91	3.207	3.26	1.12	6.0167	6
2	2.94	0.91	5.688	3.26	1.12	7.3017	7.3
3	2.94	0.91	5.246	3.26	1.12	5.9256	5.9
4	2.94	0.91	28.811	3.26	1.12	55.011	55
5	2.94	0.91	15.881	3.26	1.12	33.0347	33
6	2.94	0.91	36.680	3.26	1.12	82.9477	83
7	2.94	0.91	43.360	3.26	1.12	105.992	106
8	2.94	0.91	17.757	3.26	1.12	36.6507	36
9	2.94	0.91	189.002	3.26	1.12	418.774	423
10	2.94	0.91	10.098	3.26	1.12	13.9953	14
11	2.94	0.91	120.257	3.26	1.12	217.147	218
12	2.94	0.91	96.818	3.26	1.12	200.886	201
13	2.94	0.91	4.73	3.26	1.12	7.9741	8
14	2.94	0.91	6.165	3.26	1.12	7.9891	8
15	2.94	0.91	182.155	3.26	1.12	449.290	453

Table 3. Comparison of MMRE

	COCOMO Model	***Proposed Model***
MMRE	*45.9557*	*33.9603*

The proposed model showing the best results when the crossover is done by 1-point crossover. Table 4 shows the comparison of different crossover with respect to MMRE. As shown in the table, 2-point crossover gives 39.4501 MMRE while 3-point Crossover produce better result with 37.8205 MMRE and 1-point crossover gives best result with 33.9603 MMRE.

Table 4. Comparison between Crossover Methods

	1-Point Crossover	***2-Point Crossover***	***3-Point Crossover***
MMRE	*33.9603*	*39.4501*	*37.8205*

As shown in the table, the COCOMO PM values is optimized by tuning the coefficients A and B, the values of A and B are estimated so does the New PM value.

Table 5. Observed Parameters

No. of Chromosomes	*800*
No. of Generations	*1000*
Crossover Method	*1 -Point Crossover*

As stated above, the algorithm is simulated multiple times with different conditions and different methods. The conditions which are giving the best produce results by the algorithm are shown in Table 5.As the optimized values are tested on the all records, we get a reduced MMRE. The graphs comparing the calculated PM and estimated PM with actual effort are shown below in Figures 3 and 4. By the graphs it can be deduced that the estimated PM value is getting closer to the actual effort required for the software development.

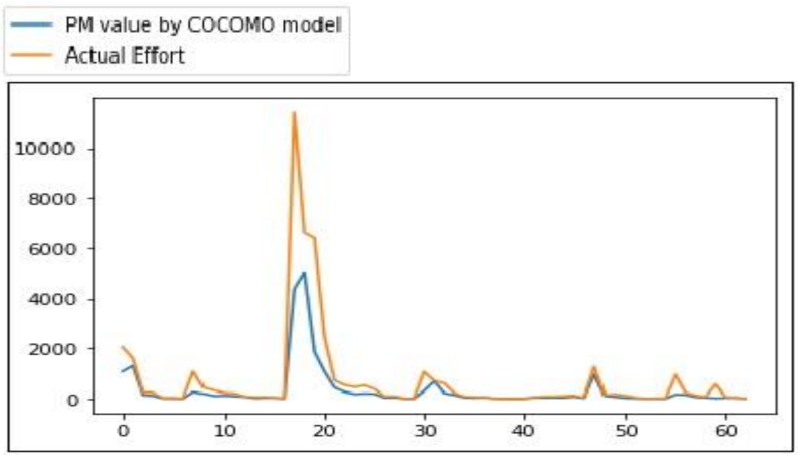

Figure 3. PM value by COCOMO Model Vs Actual Effort

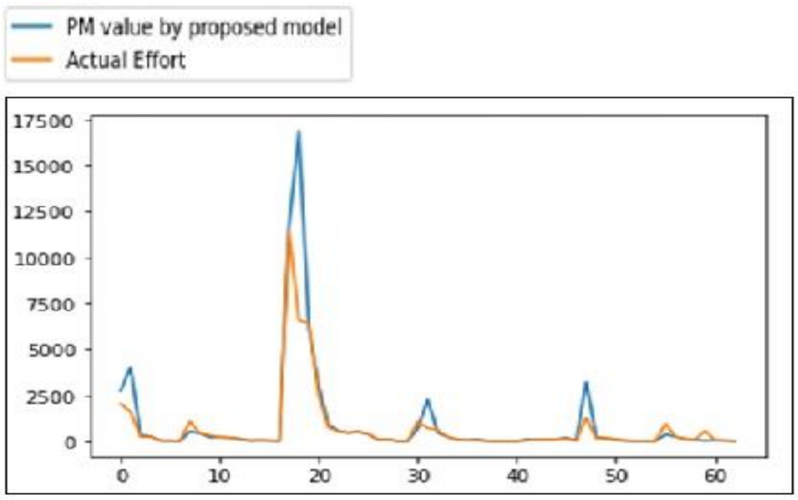

Figure 4. PM value by Proposed Model Vs Actual Effort

7. Conclusions

The objective of the paper is to optimize the COCOMO II coefficients, using genetic algorithms. This optimization is nothing new, it is done previously by neural network, fuzzy logics etc. In this paper, the optimization is based on the Genetic Algorithms (GA). In a series of tests, the proposed algorithm was re-tested the results obtained were compared with those obtained using current coefficients of COCOMO II. The results show that in most cases the results obtained using coefficients prepared by the proposed algorithm they approach obtained using current coefficients. According, to this research, if appropriate statistical data is given Genetic Algorithms are efficient in the optimization of COCOMO II coefficients.

The advantages are COCOMO II is real and easy to interpret. One can clearly understand and see how it functions. Works on historical data and thus is more predictable and precise. The drivers are very helpful to understand the impact on the different factors that affect the project costs.By using the Genetic Algorithm on the COCOMO II model; the coefficients of COCOMO II are optimized, giving us a better result. With the help of GA, the model is adaptable and easily modified. It contains a wide solution space, try to reaching the global minima and avoiding the local minima.

The disadvantages are COCOMO II model ignores requirements and all documentation. It disregards client aptitudes, participation, information and different parameters. It is reliant on the amount of time spent in each stage. In our model, as the GA is a probabilistic model, it might not find the most optimum results.It's also hard to choose parameters like number of generations, chromosomes etc.

8. References

[1] B. W. Boehm et al., Software engineering economics. Prentice-hall Englewood Cliffs (NJ), 1981, vol. 197.

[2] B. Boehm, B. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby, "Cost Models for Future Software Life Cycle Processes: COCOMO 2.0," http://dx.doi.org/10.1007/BF02249046

[3] K. Tang, K. Man, S. Kwong, and Q. He, "Genetic algorithms and their applications," Signal Processing Magazine, IEEE, vol. 13, no. 6, pp. 22–37, 1996

[4] S. Bhatia, A. Bawa, and V. K. Attri, "A Review on Genetic Algorithm to deal with Optimization of Parameters of Constructive Cost Model," International Journal of Advanced Research in Computer and Communication Engineering, vol. 4, no. 4, April 2015.

[5] A. Galinina, O. Burceva, and S. Parshutin, "The Optimization of COCOMO Model Coefficients Using Genetic Algorithms," Informationn Technology

and Management Science, vol. 15, no. 1, pp. 45–51, 2012.

[6] M. Algabri, F. Saeed, H. Mathkour, and N. Tagoug, "Optimization of Soft Cost Estimation using Genetic Algorithm for NASA Software Projects," in Information Technology: Towards New Smart World (NSITNSW), 2015 5th National Symposium on, Feb 2015, pp. 1–4.

[7] A. F. Sheta, "Estimation of the COCOMO Model Parameters using Genetic Algorithms for NASA Software Projects," Journal of Computer Science, vol. 2, no. 2, pp. 118–123, 2006.

[8] B. K. Singh and A. K. Misra, "Software Effort Estimation by Genetic Algorithm Tuned Parameters of Modified Constructive Cost Model for NASA Software Projects," International Journal of Computer Applications, vol. 59, no. 9, pp. 22–26, Dec 2012.

[9] E. Manalif, "Fuzzy Expert-COCOMO Risk Assessment and Effort Contingency Model in Software Project Management," Master's thesis, The University of Western Ontario, 2013.

[10] Tim Menzies, Member ,IEEE , zhihacchen , JairusHihn and Karen Lum Selecting Best Practices for Effort Estimation, IEEE Transaction on Software Engineering , Vol. 32, No. 11, November 2006

[11] S. Adaekalavan, and C. Chandrasekar,A Comparative Analysis of Fuzzy C-Means Clustering and Harmonic K Means Clustering Algorithms. European Journal of Scientific Research ISSN 1450-216X Vol.65 No.1 pp. 45-49, 2011.

[12] Chipperfield, A. J., P. J. Fleming, and H. P. Pohlheim, "AGenetic Algorithm Toolbox for Matlab", Proceeding of International Conference on System engineering, Coventry, 6-8 september 1994.

[13] E. Mendes, N. Mosley, and s Counsell, "A Replicated Asessment of the use of adaption rules to improve web cost estimation", International symposium on empirical software engineering, pp.-100-109.2003.

[14] S. J. Huang, N. H. Chiu, L. W. Chen, "Integration of the grey relational analysis with genetic algorithm for software effort estimation", European Journal of Operational Research 188, 898-909, 2008. http://dx.doi.org/10.1016/j.ejor.2007.07.002

[15] Enrique Alba a, Gabriel Luque and Lourdes Araujo. 2006. ―Natural language tagging with genetic algorithms‖. Information Processing Letters 100 (2006) 173–182.

[16] Praveen Ranjan Srivastava1 and Tai-hoon Kim2. 2009. ―Application of Genetic Algorithm in Software Testing‖. International Journal of Software Engineering and Its Applications Vol. 3, No.4, October 2009.

[17] Behrouz Minaei-Bidgoli , William F. Punch. 2003. ―Using Genetic Algorithms for Data Mining Optimization in an Educational Web-based System‖. Genetic and Evolutionary Computation — GECCO 2003