Genetic Algorithm selection approach estimating COCOMO Model Parameters for NASA software.

Mohammed Mahmud
Computer Science Department
Southern Connecticut State University
New Haven, Connecticut
mahmudm2@southernct.edu

***Abstract*—Estimating the cost of a software project is one of the most important facets of the project. To produce reliable performance, any software project requires cost estimation. For several causes, teams fail to meet their objectives. Two of the most critical causes are overspending and underspending. Several different kinds of models can be used to solve these problems. One of the most popular models for resolving these problems is COCOMO. This study used the genetic algorithm to forecast the COCOMO model parameter using different selection approaches. In this study, we propose three different models. The evolved models were put to the test on the NASA software project dataset. The established models were able to do well in terms of estimation.**

***Index Terms*—COCOMO, KLOC, LOC, NASA software, Genetic algorithms**

I. Introduction

The estimate of resources, cost, and schedule for software engineering effort required to experience, Access to good history information. The courage to commit to a quantitative prediction when qualitative information is all the exits. The estimated carries inherent risk, and the risk leads to uncertainties. For the project, estimation project scope must understand the elaboration and decomposition necessary. Historical metrics are beneficial. At least two different techniques should be used, like past or similar project experience. Conventional estimation techniques such as task breakdown and effort estimation, empirical models, automated tools, function points, line of code (loc). COCOMO model helps software engineers To estimate the cost of any project. COCOMO stands for **"co"** construction **"co"** cost **"mo"** model. One of the best ways to calculate any project estimate is COCOMO.

A well-known model framework for estimating software effort is the Constructive Cost Model (COCOMO). Boehm created COCOMO [1], [2]. 63 separate software projects were used to build his model. The model can be used to figure out the statistical relationship between the time it takes to design software and the time it takes to produce it. This model is based on LOC (line of code), where the project estimates done base on the total line of codes required to develop the system. I.e., the size of the system defines the cost of the project. COCOMO is used for many things, such as estimated cost, development time, average staff size, productivity, etc. It is one of the most hierarchies for the software estimation models. It consists of three hierarchies increasingly detailed and accurate forms such as basic, intermediate, and detailed COCOMO model. In this research, we focused on the basic COCOMO model. The basic model estimates the software roughly and quickly. It's mostly used for small to medium-sized software. There are three fundamental in the basic development such as organic semidetached and embedded. We use those three fundamental to calculate effort, Development time, Average staff size, and productivity according to different criteria. The way we could measure effort using this equation below. The unit of measure is Person-Month. The KLOC stands for thousand lines of code, Where 'a' and 'b' are empirically determined constants.

$$Effort = a(KLOC)^b \tag{1}$$

To calculate development time, we could use this equation below here. The unit of measures Month. Where 'c' and 'd' are empirically determined constants.

$$DevelopmentTime = c(Effort)^d \tag{2}$$

To calculate the average staff size, we could use the equation below here. The unit of measures is a person. Effort and Development from the equation were shown above.

$$Averagestaffsize = (Effort/DevelopmentTime) \tag{3}$$

II. Related Work

Much research comes out regarding this area every year; There's research that uses Artificial Intelligence in this model [3]. [4]The research uses fuzzy logic optimization and Artificial Intelligence on this model to predict the software engineering project estimation cost.

In this research, we'll discuss how to construct estimation models COCOMO using evolutionary computation techniques. This research specifically aims to develop a genetic algorithm-based evolutionary model for estimating software effort. The parameters of a COCOMO-style effort estimation model will be estimated using GAs. A genetic algorithm is an adaptive search algorithm based on natural selection, as described by Darwin [5]. Using a population of individuals deemed potential solutions to the problem under analysis, As we know, GAs explores the space of all possible solutions.

III. Optimization

Stochastic optimization (SO) approaches to generate and use random variables. The random variables appear in the formulation of the optimization problem itself, including random objective functions or random constraints, for stochastic problems. The sarcastic optimization method can work with any problem, but it can guarantee the optimal global solution. Sarcastic methods provide the optimal global solution with a probabilistic guarantee only, and this probability will become 1 in infinite computing time. Nevertheless, there is no algorithm capable of finding a Global optimization solution with certainty to a general optimization problem with endless computing time. In real-world implementations, sarcastic optimization approaches outnumber deterministic ones, according to a literature review [6]. This may be attributed to a variety of factors.

As we know, sarcastic methods do not require sophisticated mathematical analysis to solve the problem Sarcastic methods could handle practical and large-scale problems better than a deterministic one.

IV. Methodology

A. Genetic Algorithms:

A genetic algorithm is a nonlinear, non-differentiable optimization technique. To find a global minimum optimization of a problem, they use evolutionary biology concepts. The genetic algorithm begins by generating a set of candidate solutions that are then evaluated against the objective function. We then use selection crossover and mutations to produce subsequent generations of points from the first generations to many generations.

B. Data Set

Experiments on a data set provided by Bailey and Basili [7] have been undertaken to create an effort estimating model. The Developed Line of Code (DLOC), the Methodology (ME), and the Measured Effort (ME). The Effort is measured in man-months, and the DLOC is measured in Kilo Lines of Code (KLOC). Table 1 contains the Dataset of the project data. The first 13 projects' data were used to approximate model parameters, while the remaining 5 projects were used to assess their accuracy.

C. Specific Approach

The model parameters for these models are usually set depending on the type of software project. We aim to use GAs to recalculate the COCOMO model parameters. This will allow us to calculate the effort spent on the NASA program. The approximate parameters would make the computation of the built effort for all projects even more general.

$$e = a(KLOC)^b \qquad e = 2.7076(KLOC)^0.8431 \tag{4}$$

TABLE I
NASA software project data

Project No	KDLOC	ME	Measured Effort
1.	90.2000	30.0000	115.8000
2.	46.2000	20.0000	96.0000
3.	46.5000	19.0000	79.0000
4.	54.5000	20.0000	90.8000
5.	31.1000	35.0000	39.6000
6.	67.5000	29.0000	98.4000
7.	12.8000	26.0000	18.9000
8.	10.5000	34.0000	10.3000
9.	21.5000	31.0000	28.5000
10.	3.1000	26.0000	7.0000
11.	4.2000	19.0000	9.0000
12.	7.8000	31.0000	7.3000
13.	2.1000	28.0000	5.0000
14.	5.0000	29.0000	8.4000
15.	78.6000	35.0000	98.7000
16.	9.7000	27.0000	15.6000
17.	12.5000	27.0000	23.9000
18.	100.8000	34.0000	138.3000

V. Experimental Results and Discussion

As we know, In a genetic algorithm, there are various parameters. Our first experiment was conducting manipulating parameters in a genetic algorithm such as changing Population size, Crossover Percentage, Number of Iterations, Mutation Rate and etc. we did not succeed on this approach.

Our second approach was the same as first approach, but we change the selection method, we test on 3 three different methods. Such as Roulette Wheel, Tournament and Random We can represent roulette wheel selection using this equation below

$$Pi = \frac{fi}{\sum_{j=1}^{N} fi} \tag{5}$$

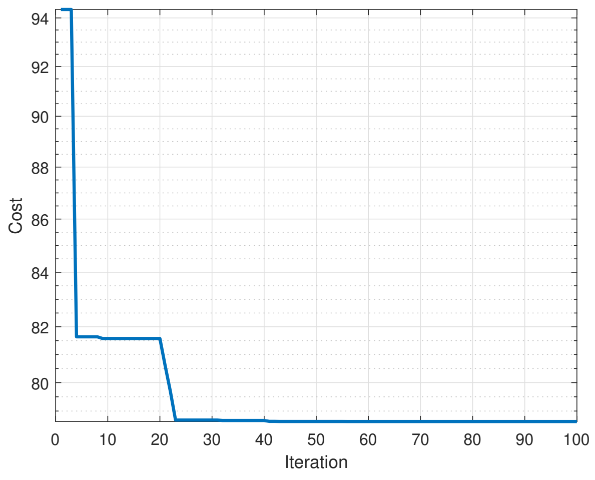
For Tournament and Random is just a traditional approach. No equation is needed to represent.

A. Hardware, Software Environment

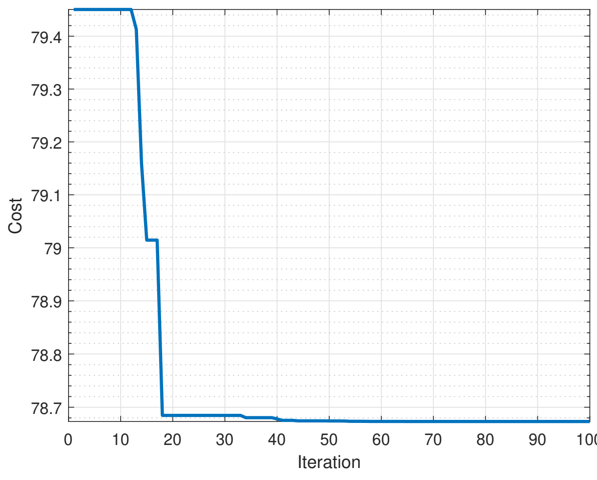
TABLE II
Workstation Specification

Machine Types	Desktop Workstation
CPU	Intel(R) Xeon(R) CPU @ 2.00GHz
RAM	16 GB
OS	Windows 10
Software	Matlab

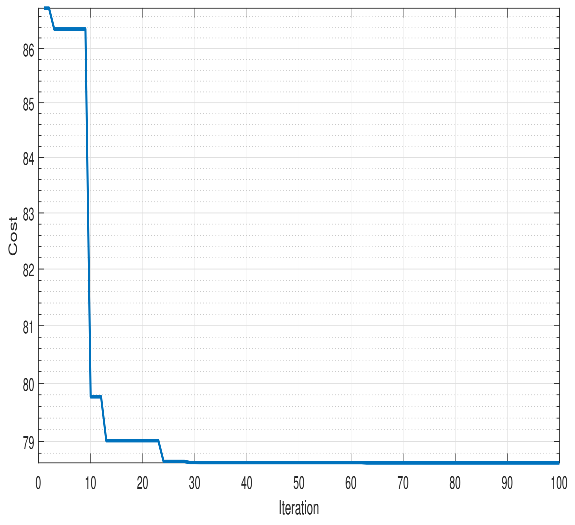
Although most of the optimization and learning models required high-performance computation and time. Our



a. Roulette Wheel



b. Tournament



c. Random

Fig. 1. Iteration vs Cost

model separates the old traditional computational approach from a smaller and narrower approach. In table 3 below a specification that summarizes the workstation. As we know, the genetic algorithm can be computationally heavy if it's a bigger search space. As we discussed earlier, our search space is lower bound 10 and upper bound 10. With lower search space, we save computation time significantly. Since the research was conducted using the Matlab programming language. To improve the fitness function, we also use the Matlab - optimization tools.

B. Error Measure Metric

For our error metric, we use MSE. Mean square error most common loss function for regression. As we can look into the data, the data is normally distributed. We can represent MSC using this equation below here Eq: 6

$$MSE = \frac{1}{n}\sum_{t=1}^{n}(Y_i - \hat{Y_i})^2 \tag{6}$$

This formula gives us the spread of the result around the true value. Since MSE is vulnerable to outliers, the best approximation would be the mean target value of multiple examples of the same input function values. Compared to Mean Absolute Error, where the best approximation is the median, this is the better option. MSE is useful if you assume your target data is naturally spread around a mean value, conditioned on the information, and it's important to penalize outliers heavily.

C. Discussion and Result

In fig 1, as you can see, all the iteration vs. cost charts. As you could see, the roulette wheel approach

TABLE III
RESULTS OF THE SEARCH PROCESS.

Project No	Measured Effort	Roulette Wheel	Tournament	Random
1	115.80	124.38	124.35	124.18
2	96.00	69.72	69.73	69.88
3	79.00	70.11	70.12	70.27
4	90.00	80.43	80.43	80.54
5	39.00	49.50	49.52	49.73
6	98.40	96.79	96.78	96.80
7	18.40	22.97	22.98	23.19
8	10.30	19.35	19.37	19.56
9	28.50	35.97	35.99	36.21
10	7.00	6.73	6.74	6.86
11	9.00	8.76	8.77	8.90
12	7.30	14.96	14.98	15.15
13	5.00	4.81	4.82	4.91
14	8.40	10.18	10.20	10.34
15	98.70	110.41	110.39	110.33
16	15.60	18.07	18.08	18.27
17	23.90	22.50	22.52	22.72
18	138.30	136.93	136.88	136.62

has a better efficient result than the tournament and the Random approach. Table 3 represents The Roulette Wheel, Tournament & Random approach and shows the real assessed effort for each of the 18 projects and the projected effort based on the GAs model.

TABLE IV
ALL SELECTION APPROACH

Operator	Type
Crossover Percentage	0.3
Mutation Percentage	0.2
Population size	125
Maximum generation	100

Table 4 lists the tuning parameters for the GA evolutionary method used to approximate COCOMO model parameters such as population size, crossover, mutation types, and selection mechanisms.

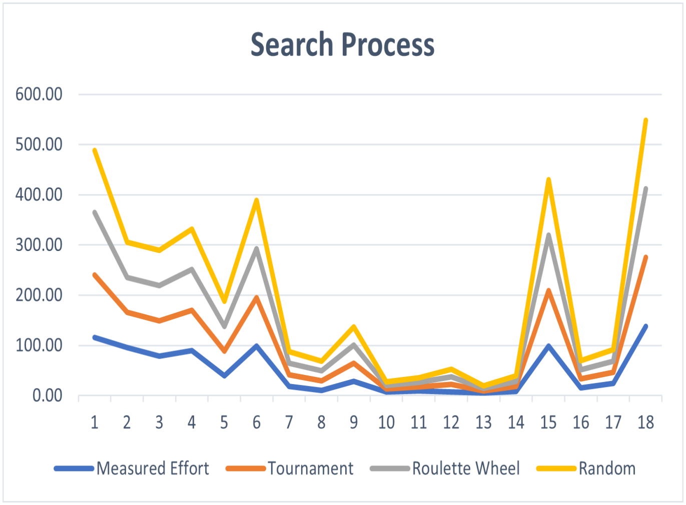


Fig. 2. Genetic Algorithm Selection Approach

As you could see, all these tests/models were running on a maximum of 100 iteration populations. Our search space for lower and upper bound was 10, 10. our mutation percentage was 20 and Crossover Percentage 30

D. Proposed Model

As can be shown, our tournament selection method model generates better results than any other approaches so we proposed a model with a better parameter. Table 5 shows the specific parameters

TABLE V
TOURNAMENT APPROACH

Operator	Type
Crossover Percentage	0.2
Mutation Percentage	0.1
Population size	150
Maximum generation	100
Upper Bound	10
Lower Bound	0

TABLE VI
ME VS TOURNAMENT APPROACH

Project No	ME	Tournament approach
1	115.80	120.38
2	96.00	85.72
3	79.00	70.11
4	90.00	90.43
5	39.00	49.50
6	98.40	96.79
7	18.40	22.96
8	10.30	9.35
9	28.50	35.97
10	7.00	6.73
11	9.00	8.76
12	7.30	10.96
13	5.00	4.81
14	8.40	10.18
15	98.70	110.41
16	15.60	16.06
17	23.90	22.50
18	138.30	136.93

Our aim, as previously mentioned, is to identify the model parameters that are best suited to predict project development software effort reliably. Table 6 displays the real calculated commitment and the projected effort utilizing the same dataset as the proposed model.

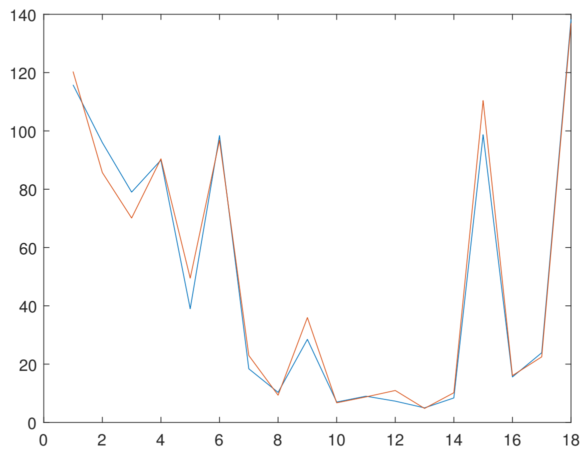


Fig. 3. Genetic Algorithm Selection Approach

Fig3 above shows a visual representation of the effort distribution

VI. Proposed future Work and Conclusion

We suggested one new model system with three different approaches in this analysis to estimate the software initiative for NASA-sponsored projects using genetic algorithms. To account for the impact of methods on effort estimation, modified versions of the popular COCOMO model were given. The built models were put to the test on NASA software project data provided in [7]. The established models were able to do well in terms of estimation. We propose that the neural network be used to refine the genetic algorithm's parameter to an appropriate model configuration for the software effort.

References

[1] J. Matson, B. Barrett, and J. Mellichamp, "Software development cost estimation using function points," *IEEE Transactions on Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[2] C. B. H. E. e. a. Boehm, B., "ost models for future software life cycle processes: Cocomo 2.0," *springer*, 1995.

[3] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736–743, 1997.

[4] D. S. N. Rahul Kumar Yadav, "Software effort estimation using fuzzy logic: A review,," *International Journal of Engineering Research Technology (IJERT)*.

[5] K. Man, K. Tang, and S. Kwong, "Genetic algorithms: concepts and applications [in engineering design]," *IEEE Transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519–534, 1996.

[6] Z. Michalewicz, "Genetic algorithms, data structures=evolution programs.]," *Springer-Verlag.*, 1994.

[7] J. W. Bailey and V. R. Basili., "A meta model for software development resource expenditure. proc. intl. conf. software engineering]," *Springer-Verlag.*, 1981.