Influence of Project Duration constraint on efforts in Constructive Cost Model and optimizing the efforts to obtain accuracy

Brajesh Kumar Singh[1], A. K. Misra[2]

[1,2] Department of Computer Science and Engineering, MNNIT, Allahabad, India

Abstract

Software cost estimation accuracy is one of the biggest challenges in the field of software development for developer and customers. In general, many algorithmic models like Constructive Cost Model (COCOMO) are used to estimate cost but they have inability to deal with uncertainties related to software development environment and other factors influencing the software development process. The Evolutionary computation approaches provide the solution for estimating the effort along with handling these uncertainties. In this paper, COCOMO is used as algorithmic model and an attempt is being made to validate the soundness of genetic algorithm using NASA project data. The main objective of this work is to analyze the influence of project duration constraints on efforts and to improve accuracy of system's output when evolutionary computation based approach is applied to the NASA dataset to derive the software effort estimates. Proposed approach is validated by using 63 NASA project dataset. Empirical results show that application of the proposed approach for software effort estimates resulted in smaller mean magnitude of relative error (MMRE) for all cases and probability of a project(PRED) having a relative error of less than or equal to 0.35 as compared with results obtained with COCOMO is improved significantly for most of the cases.

Keywords: *Evolutionary Computation, Genetic Algorithm, COCOMO, Effort estimation, Mean Magnitude of Relative Error, Probability of a project.*

1. Introduction

The software project management is a set of activities that span all phases of the software development life cycle. The most important part of the software project management is to estimate a proposed project effort, duration and cost more accurately [39]. Estimation of effort and schedule of software development has become a topic of growing importance of interest, which is not so much surprising. It often happens that software is more expensive than estimated cost and completion is later than estimated time. Moreover it turns out that most of the software do not meet the demands of the customer[15]. It is due to the characteristics of software and software development makes estimating difficult. For example, the level of abstraction, complexity, measurability of product and process, innovative aspects, etc. A big number of factors have an influence on the effort, cost and time to develop software. These factors are widely known as 'cost drivers'. Few of them are size and complexity of the software, commitment and participation of the user organization, experience and expertise of the development team. In general these cost drivers are difficult to determine accurately in operation.

Several prerequisites must be fulfilled to address the problems listed above and to guarantee a sound basis for predicting effort, duration and the cost for software development [15]. Many project managers like Team leaders and system analysts have developed their own intuitive techniques for dealing with the problems in real world and they face and operate. Many of these techniques are adequate but most of them provide far less precise estimates and control than desired. There is no question that the software development industry desperately needs better techniques of estimating software project costs and completion times, controlling the development process and eliminating errors which are costlier [39]. Software project Management process is used for carefully considering costs and software benefits before committing the required resources to that project or bidding for a contract [2].

In recent times, many quantitative models of software cost estimation have been developed. Most of these models are based on the size measure, such as Source Lines of Code (SLOC) and Function Point (FP), obtained from size estimation. Based on the context that the accuracy of size estimation has direct impact on the accuracy of cost estimation, a new alternative approach in soft computing techniques such as evolutionary algorithms (EA) can be a good choice for software development effort estimation task.

Recently, many questions about the applicability of using evolutionary computational methods to build software estimation models have been introduced [14]. The

major objective of this study is to focus on building an evolutionary model for estimating software effort using genetic algorithms. Genetic algorithms will be used to estimate the parameters of a COCOMO based effort estimation model. Genetic algorithm is an adaptive search algorithm based on the Darwinian theory of natural selection. Genetic algorithm searches the space of all possible solutions using a population of individuals which is considered as potential solutions of the problem under consideration. These solutions are computed based on their fitness. The solutions that best fit to the objective criterion survive in the upcoming generations and produce "offspring" which are variations of their parents[35]. Various papers [17, 21, 24, 27, 28, 29, 30, 31, and 38] in a review of the literature show that there are two major types of cost estimation methods i.e. Algorithmic and Non algorithmic models are the records of the conference. ACM hopes to give these conference by-products a single, high-quality appearance. To do this, we ask that authors follow some simple guidelines. In essence, we ask you to make your paper look exactly like this document. The easiest way to do this is simply to down-load a template from [39], and replaces the content with your own material.

2. About The Problem

2.1 Algorithmic models

Since, at the early stages of the software project development process, all the earned information is not adequately available, the predictions may be inaccurate and this problem is seen in most of the software projects rather than the other project types.

The first idea for software effort estimation was introduced in 1950 by presenting the manual rule of thumb [23]. The late 1970s produced a flowering of more robust models for estimation. In 1965, by increasing the number of software projects and need of user society to earn high quality software, some models based on the linear equations were presented as the software effort techniques [5]. We can consider the name of Larry Putnam, Barry Boehm and Joe Aron, as the ancestors of software estimation methods [23]. Onwards in 1973, the IBM researchers presented the first automated tool, interactive productivity and quality (IPQ) [23]. Barry Boehm proposed a new method based on computing some of the software project factors by means of several mathematical equations called COCOMO [4]. In addition, Boehm explained several algorithms in his book "Software Engineering Economics" [4] that are still used by researchers. Other models such as Putnam Lifecycle Management (SLIM) [27] and software evaluation and estimation of resources – software estimating model (SEER - SEM) [22] were influenced by the principals of COCOMO [5]. Albrecht and Gaffney [1] introduced the function point (FP) as a metric for software size estimation which was the other important event in that decade. Analogy based method was proposed in 1997 [33].

These Traditional algorithmic techniques require long term estimation process. Algorithmic models are based on the statistical analysis of historical data (past projects) [19, 37]. All of them need inputs, which are accurate estimate of specific attributes, such as Line Of Code (LOC), number of user screens, interfaces and complexity, which are not easy to acquire at the early stages of software development. Besides, attributes and relationships used to predict software development effort could change over time and/or vary for different software development environments [36].

Understanding and calculation of algorithmic techniques based past projects are different due to implicit complex relationship between the related attributes. Attributes and relationships used to estimate software development effort could change overtime and differ for software development environment and hence may create problems to software developers in committing resources and controlling costs.

Although most of these pioneers started working on developing models of software cost estimation at about the same time, they all faced the same dilemma: as the software size increases and importance there is also a growth in complexity, which makes it very difficult to accurately predict the cost of software development. In order to address and overcome these problems, a new model with accurate estimation is always desirable. This dynamic field of software cost estimation sustained the researcher's interests who succeeded in setting the stepping-stones of software engineering cost models.

2.1.1 Constructive Cost Model (COCOMO)

COCOMO[4,8] is the best documented and most transparent model currently available. The main focus in COCOMO is upon estimating the influence of 15 cost drivers on the development effort. Before this can be done, an estimate of the software size must be available. COCOMO does not support the sizing estimation stage: it only gives several equations based on 63 completed projects at TRW. The equations represent the relationships between size and effort and between effort and development time. Estimation is dependent on the various modes of projects which are shown in table-1.

Table 1: Describing the values of a and b for intermediate COCOMO

Project modes	A	B
Organic	3.2	1.05
Semidetached	3.0	1.12
Embedded	2.8	1.20

A distinction is made between three development modes: the organic mode (stable development environment, less innovative, relatively small in size); the semi-detached mode (between organic and embedded mode) and the embedded mode (developing within tight constraints, innovative, complex, high volatility of requirements). The effort estimation is adjusted by the influence of 15 cost drivers.

In Table 2, the 15 COCOMO cost drivers are listed with the adjustment for each driver value. For example: where the complexity of the software is determined to be extra high, the effort has to be calculated by multiplicative factor of 1.65. Furthermore COCOMO provides tables to apportion the adjusted estimated effort and development over the project phases and, in the detailed version of the model, to refine the adjustment for each phase. For example: the quality of the programmer has less influence in the feasibility phase than in the design phase. Thus phase dependent adjustment factors are used in the detailed model.

Table 2: Describing fifteen cost drivers

Cost Driver	Ratings					
	Very Low	Low	Nominal	High	Very High	Extra High
product attribute						
RELY	0.75	0.88	1.00	1.15	1.4	-
DATA	-	0.94	1.00	1.08	1.16	-
CPLX	0.70	0.85	1.00	1.15	1.3	1.65
Computer Attribute						
TIME	-	-	1.00	1.11	1.3	1.66
STOR	-	-	1.00	1.06	1.21	1.56
VIRT	-	0.87	1.00	1.15	1.3	-
TURN	-	0.87	1.00	1.07	1.15	-
Personnel Attribute						
ACAP	1.46	1.19	1.00	0.86	0.71	-
AEXP	1.29	1.13	1.00	0.91	0.82	-
PCAP	1.42	1.17	1.00	0.86	0.7	-
VEXP	1.21	1.10	1.00	0.9	-	-
LEXP	1.14	1.07	1.00	0.95	-	-
Project Attribute						
MODP	1.24	1.10	1.00	0.91	-	-
TOOL	1.24	1.10	1.00	0.91	-	-
SCED	1.23	1.08	1.00	1.04	-	-

The three ways of estimating software project effort/cost with increasing levels of accuracy are simple, intermediate and complex models. These three models are defined by increasing the details in mathematical relationship between the developed time, the effort and the maintenance effort [9]. The software cost estimation accuracy is significantly improved when we adopt models such as the Intermediate and Complex COCOMOs [6]. The COCOMO for effort estimation has the form given in Equation 1.

$$\text{Effort} = a\,(\text{KLOC})^{b} * \text{EAF} \quad (1)$$

The software effort is computed in person-months. The values of the parameters a and b depend mainly on the class of software project. Software projects were classified based on the complexity of the project into three categories EAF is Effort Adjustment factor which depends on the values 15 cost drivers. In this paper, the intermediate COCOMO is used. The effort multipliers fall into three groups: those that are positively correlated to more effort; those that are negatively correlated to more effort; and a third group containing just schedule information. In COCOMO-I, cost driver "SCED" has a U-shaped correlation to effort; i.e. giving programmers either too much or too little time to develop a system can be disruptive [7]. This exhibits some nonlinearity characteristics.

The limitations of the algorithmic models led to the exploration of the non-algorithmic techniques which are soft computing based. So, based on these contexts, new alternative approaches like soft computing techniques are required for better solutions.

3. Solution of the Problem

3.1 Non-Algorithmic Models

Newer computation techniques, to estimate the software effort are non-algorithmic approaches. Most of them came in 1990s are soft computing based, and drew the attention of researchers towards them. This section discusses a few of such non-algorithmic models for software development effort estimation. Soft computing consists of various approaches like evolutionary algorithm (EA), fuzzy logic (FL) and artificial neural networks (ANN). These methodologies use flexible data processing mimicking human behavior to deal with real life problems. Soft computing techniques have been widely used by researchers for software development effort prediction, with an objective to manage the imprecision in data and uncertainty in data. The Evolutionary Algorithms have to been effectively used to search the optimal solution for a given problem. The first model based on fuzziness of several aspects is one of the best known [10], most successful and widely used model for cost estimation, COCOMO, was that of Fei and Liu [40]. They observed that is not feasible before starting the project to accurately estimate the delivered source instruction (KDSI); and it is unreasonable to assign a finite number for it.

In summary, the previous research reveals that all of the software effort prediction models based on soft computing, have it own pros and cons. Therefore Selecting a suitable technique for the given problem is a difficult decision. It itself requires some ranking for each computing technique so as to decide when a particular approach is to be applied to any prediction problem. In the present study an effective model based on Genetic Algorithm has been proposed to heuristically search the various values of the parameters in the given search space.

4. Proposed Approach for Solving Problem

4.1 Dataset Description

We have considered the data from 63 NASA projects from different centers for projects sourced from Boehm's 1981 text, p.496-497 Table 29-1, transcribed by Srinivasan and Fisher [7]. Dataset consists of 15 cost drivers, 1 attribute of the 3 development modes, Project Size (in KLOC), and Actual effort used to evaluate the prediction done by different approaches.

4.2 Proposed Approach

A brief overview of the binary genetic algorithm is as follows:

4.2.1Genetic Algorithms

A genetic algorithm (GA) is a search heuristic method that imitates the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems [26]. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover, based on the Darwin theory of natural selection. This concept was first introduced by John Holland [20] and considerably studied by Goldberg [18], De Jong [12, 13] and back [3]. GAs search the space of all possible solutions using a population of individuals which is taken as potential solutions of the problem under study. These solutions are computed based on their fitness. The solutions that best fit to the objective criterion survive in the upcoming generations and produce "offspring" which are transformations of their Parents [34].

GAs has been successfully used in a wide range of difficult numerical optimization problems. They have been successfully used to solve system identification, signal processing and path searching problems [11, 16, 24 and 32]. String representation of genetic algorithms was evolved by Holland [20].

4.2.2 Evolutionary Process of Genetic Algorithm

In all Evolutionary Algorithms (EAs) techniques, it is required to map the problem from its real domain to the Evolutionary algorithms domain. GAs offers various kinds of representations. The evolutionary process starts with the evaluation of the fitness for each individual belonging to initial population set. Until the stopping criterion is not reached, the following tasks are to be preformed;

- Select the good individuals for reproduction in matting pool using some selection approach (like. roulette wheel, tournament, rank, etc.).
- Use crossover and mutation operators to generate new offspring's. The probability of crossover and mutation are selected based on the application.
- Evaluate the fitness function for offspring's.

This Stopping condition for above steps is either the optimal solution required or the maximum numbers of iteration specified are completed.

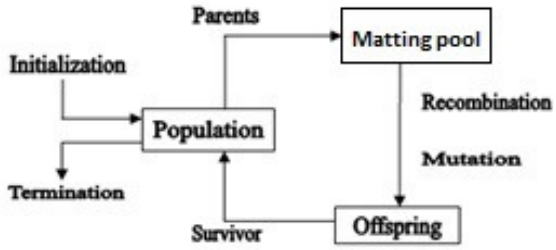


Figure 1: General Scheme of Evolutionary Process

To see, how the ideas of evolutionary algorithms is implemented on function optimization, It is assumed that without any loss in generality we are desired to minimize/maximize a function of n arguments $f(a_1, a_2, ..., a_n)$. Each argument a_i of the function has it range from α_i to γ_i, which is used as search space for each parameter that can be given by equation below:

$$D_i = \{(\alpha_i, \gamma_i) : 1 \le i \le n\} \quad (2)$$

f(a1, a2, ..., an) is positive function, Such that ai always in there domain Di. Candidate solutions are represented by n-dimensional vectors of argument of the form: a1, a2,..., an which is known as "Chromosomes" and these chromosomes the order pair of the arguments of the functions which can independently called as "genes". For each such vector of arguments, have associated functions

to serves a single value that behaves as fitness value. These small values are used for minimization problems.

The GA search process works on population of individuals each of which is assigned a fitness value. Individuals with higher fitness value are transferred to the mating pool to generate the offspring's which posses many but not all of the features of their parents. This is done using genetic operators like mutation and crossover [23, 28].

5. Evaluation Method

The performance of an effort predicting algorithm can be evaluated in many ways but the most commonly used are Mean Magnitude of Relative Error (MMRE) and probability of a project having a relative error of less than or equal to L (PRED(L)).
MMRE and PRED are computed from the Magnitude relative error, or MRE, which is the relative magnitude of the difference between the actual and estimated value of individual effort i.

$$MRE_i = \frac{|\,Estimated`\_effort_i - actual\_effort_i\,|}{actual\_effort_i}$$

The MRE value is calculated for each observation i of actual and predicted effort where i range from 1 to N. Then the mean of MRE over multiple observations (N) can be achieved through the formula of Mean MRE (MMRE) as follows:

$$MMRE = \frac{1}{N}\sum_{i}^{N} MRE_i$$

Another criterion is the prediction at level L, Pred(L) = k/N, Here k is the number of observations where MRE is less than or equal to L and N is the total number of observations. Thus, Pred(35) gives the percentage of projects which were predicted with a MRE less than or equal to 0.35.

6. Result and Discussion

Initially the data set is divided into different categories according to the different values of SCED. The 80 percent of each set will be used for evaluating MMRE as the fitness function of binary genetic algorithm. The initial population is randomly created having the chromosome as SCED. This population is assigned a fitness value and transferred to matting pool, where crossover and mutation are used. After the stopping criterion is reached, the best individual will be used for testing the result on the complete set. The details about the various genetic algorithm operator and parameters are provided below in Table 3.

Table 3: Genetic algorithm operator and parameters

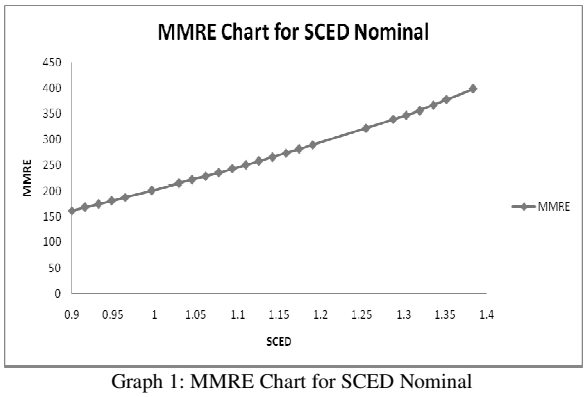
Population Size	10
SCED range	0.9 to 1.4
Selection Operator	Tournament Selection
Cross over	Single point crossover
Probability of Crossover	0.8
Mutation	Single bit mutation
Probability of Mutation	0.3
Fitness Function	MMRE
Number of Generations	10

The result obtained at evaluation phase of new values of SCED are described and compared with COCOMO values on the basis of MMRE and PRED in the Table 4.

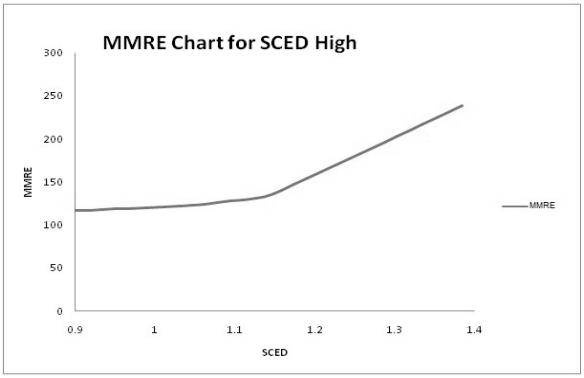
Table 4: Describing COCOMO SCED, NEW SCED, MMRE, PRED

SCED OLD	SCED NEW	MMRE COCOMO	MMRE NEW	PRED(35) COCOMO	PRED(35) NEW
1	0.9	0.3382421	0.29053524	21	25
1.04	0.9	0.4004661	0.35403361	7	7
1.08	1.31935	0.2975117	0.26223239	6	7
1.23	1.10968	0.1808959	0.18040218	8	6

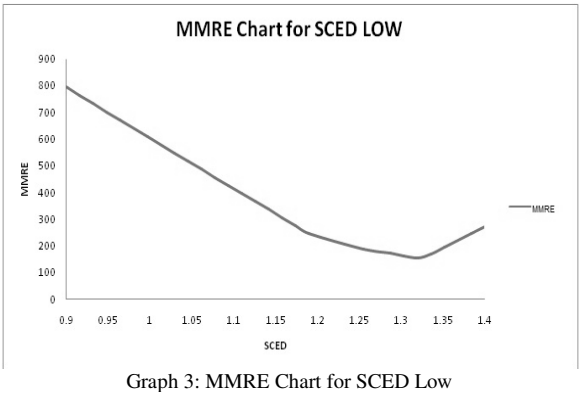
The Graph 1, Graph 2, Graph 3 and Graph 4 are describing the change in the value of MMRE for the various values in the range that is from 0.9 to 1.4.



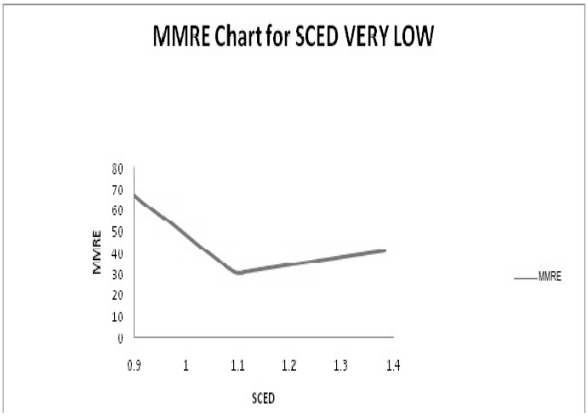
Graph 1: MMRE Chart for SCED Nominal



Graph 2: MMRE Chart for SCED High



Graph 3: MMRE Chart for SCED Low



Graph 4: MMRE Chart for SCED Very Low

7. Conclusion

Paper presented the optimized values of duration parameter there by increasing the accuracy of the estimated effort. This work has further shown that accurate effort estimation is possible by evaluating algorithmic and non algorithmic software effort estimation models. The proposed model showed better software effort estimates in view of the MMRE, Pred(0.35) evaluation criteria as compared to the traditional COCOMO.

The various graph for different values of SCED ie Nominal, high, low and very low were plotted as described in Graph 1, Graph 2, Graph 3, Graph 4 to show their change in MMRE according to the different values of SCED. From Table 2 we can see that the MMRE has significantly reduced and also for most of the cases, the value of PRED (35) has come down. The utilization of Soft computing based approaches for searching the optimal values software engineering field can also be explored in the future.

References

[1] Albrecht AJ, Gaffney JA (1983). Software function, source lines of 6392 Sci. Res. Essays codes, and development effort prediction: a software science validation. IEEE Trans Software Eng. SE. 9(6): 639-648.

[2] B. Boehm and et all, Software Cost Estimation with COCOMO II. Prentice Hall PTR, 2000.

[3] Back, T. and H.P. Schwefel, 1993. An overview of evolutionary

[4] Boehm BW (1981). Software engineering economics. Englewood Cliffs, NJ: Prentice Hall.

[5] Boehm BW, Valerdi R (2008). Achievements and Challenges in Cocomo-Based Software Resource Estimation. IEEE Softw.,25(5): 74-83

[6] Boehm, B. Cost models for future software life cycle processes: COCOMO 2.0. Ann. Software Eng. 1: 45-60. ,1995

[7] Boehm, 1981 text,p.496-497, Table 29-1, "Software Engineering Economics", "Prentice Hall",

[8] Boehm, B W 'Software engineering economics' IEEE Trans. Soft. Eng. Vol 10 No 1 (January 1984)

[9]]C. F. Kemere, "An empirical validation of software cost estimation models," Communication ACM, vol. 30, pp. 416–429, 1987.

[10] C. Kirsopp, and M. J. Shepperd, "Making inferences with small numbers of training sets", Sixth International Conference on Empirical Assessment & Evaluation in Software Engineering, Keele University, Staffordshire, UK, 2002.

[11] Chipperfield, A.J. and P.J. Fleming, 1996. Genetic algorithms

[12] De Jong, K., 1992. Are genetic algorithms function optimizers? Proc. Sec. Parallel Problem Solving From Nature Conference, pp:3-14. The Netherlands: Elsevier Science Press.

[13] De Jong, K.A., 1975. Analysis of Behavior of a Class of Genetic Adaptive Systems. Ph.D. Thesis. University of Michigan, Ann Arbor, MI.

[14] Dolado. C.J. and M. Leey, 2001. Can genetic programming improve software effort estimation?A comparative evaluation. Inform. Software Technol., 43: 863-873.

[15] F J Heemstra, Vol 34 No 10 October 1992 09505849/92/100627-13, Information and Software Technology.

[16] Fonseca, C., E. Mendes, Fleming and S.A. Billings,1993. Nonlinear model term selection with genetic algorithms. Proc. IEE/IEEE Workshop on Natural Algorithms in Signal Process., pp: 27/1 –27/8.

[17] G.N. Parkinson, Parkinson's Law and Other Studies in Administration, Houghton-Miffin, Boston, 1957.

[18] Goldberg, D., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. New York, Addison-Wesley.

[19] Hodgkinson, A.C. and P.W. Garratt, A neurofuzzy cost estimator. Proceedings of the 3rd International Conference on Software Engineering and Applications, (SEA'99), ePrint, pp: 401-406. http://eprints.ecs.soton.ac.uk/2659/. , 1999.

[20]] Holland, J., 1975. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: University of Michigan Press.

[21] J. R. Herd, J.N. Postak, W.E. Russell, K.R. Steward, and Software cost estimation study: Study results, Final Technical Report, RADCTR77- 220, vol. I, Doty Associates, Inc., Rockville, MD, pp. 1-10, 1977.

[22] Jensen R – "An Improved Macro level Software Development Resource Estimation Model," Jensen R., Proceedings 5th ISPA Conference, , pp. 88-92, April 1983

[23] Jones C (2007). Estimating software costs: Bringing realism to estimating. New York, NY: McGraw-Hill.

[24] Kristinsson. K. and G. Dumont, 1992. System identification and control using genetic algorithms. IEEE Transaction

[25] L. H. Putnam, A general empirical solution to the macro software sizing and estimating problem, IEEE Trans. Soft. Eng., pp. 345-361, July 1978.

[26] Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. Cambridge, MA: MIT Press. ISBN 9780585030944.

[27] Putnam LH (1987). A general empirical solution to the macrosoftware sizing and estimating problem. IEEE Trans. Software Eng., 4(4): 345-361

[28] Putnam, L. and Myers, W. (1992), Measures for Excellence, Putnam, L. and Myers, W, Yourdon Press Computing Series., 1992.

[29] R. E. Park, PRICE S, The calculation within and why, Proc. of ISPA Tenth Annual Conference, Brighton, England, pp. 231-240, July 1988.

[30] R. Tausworthe, Deep Space Network Software Cost Estimation Model, Jet Propulsion Laboratory Publication 81-7, pp. 67-78, 1981

[31] R.K.D. Black, R. P. Curnow, R. Katz, M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March, pp. 5-8, 1977.

[32] Schultz. A. and J. Grefenstette, 1994. Evolvingrobot behavior. Proc. Artificial Life Conf. MIT Press.

[33] Shepperd M, Schofield C (1997). Estimating Software Project Effort Using Analogies. IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 23(11): 736-743

[34] Sheta. A. and K. DeJong, 1996. Parameter estimation of nonlinear systems in noisy environment using genetic algorithms. Proc. IEEE Intl. Symp. Intelligent Control (ISIC'96), pp: 360-366.

[35] Sheta. A. and K. DeJong, 1996. Parameter estimation of nonlinear systems in noisy environment using genetic algorithms. Proc. IEEE Intl. Symp. Intelligent Control (ISIC'96), pp: 360-366

[36] Srinivasan, K. and Fisher D., Machine learning approaches to estimating software development effort. IEEE Trans. Software Eng., 21: 126-137. DOI: 10.1109/32.345828, 1995.

[37] Strike, K., K. El-Emam and N. Madhavji. Software cost estimation with incomplete Data. IEEE Trans. Software Engg., 27: 890-908. DOI: 10.1109/32.962560, 2001.

[38] W. S. Donelson, Project Planning and Control, Proc. Datamation, pp. 73- 80, June 1976.

[39] W.Shouli, K.Dionysios,1992.Proc of IEEE Int. Conf. on Tools with AI Arlington, VA, Nov. 1992, IASCE: An Intelligent Assistant to Software Cost Estimation.

[40] Z. Fei, and X. Liu, "f-COCOMO: fuzzy constructive cost model in software engineering", Proceedings of the IEEE International Conference on Fuzzy Systems, IEEE Press, New York, pp. 331–337, 1992.