Evolving Software Effort Estimation Models Using Multigene Symbolic Regression Genetic Programming

Sultan Aljahdali and Alaa Sheta
Computer Science Department
College of Computers and Information Technology
Taif University
Taif, Saudi Arabia
aljahdali@tu.edu.sa, asheta66@gmail.com

***Abstract*—Software has played an essential role in engineering, economic development, stock market growth and military applications. Mature software industry count on highly predictive software effort estimation models. Correct estimation of software effort lead to correct estimation of budget and development time. It also allows companies to develop appropriate time plan for marketing campaign. Now a day it became a great challenge to get these estimates due to the increasing number of attributes which affect the software development life cycle. Software cost estimation models should be able to provide sufficient confidence on its prediction capabilities. Recently, Computational Intelligence (CI) paradigms were explored to handle the software effort estimation problem with promising results. In this paper we evolve two new models for software effort estimation using Multigene Symbolic Regression Genetic Programming (GP). One model utilizes the Source Line Of Code (SLOC) as input variable to estimate the Effort (E); while the second model utilize the Inputs, Outputs, Files, and User Inquiries to estimate the Function Point (FP). The proposed GP models show better estimation capabilities compared to other reported models in the literature. The validation results are accepted based Albrecht data set.**

I. Introduction

Estimating software effort on the early stage of development might produce uncertainty of up to 400% as mentioned in [1]. In 2001, it was also reported by the Standish group that, 53% of U.S. software projects ran over 189% of the original estimate [2]. In the 21st century software technology was capable on providing variety of software tools, techniques and software estimation models with many features which can help software project developer, manager, analyst and tester to do their job in a better way. The question that arises according to this opportunity, which tool and which model can really help in providing an accurate estimate? In most cases, the models adopted were based on expert judgment including Delphi technique [3] and work breakdown structure based methods. Models inspired by mathematical equations later came in line and named as Algorithmic Method. For example, Constructive Cost Model (COCOMO) [1], [4], Software Life Cycle Management (SLIM) [5], [6], and Software Evaluation and Estimation of Resources-Software Estimating Model (SEER-SEM) [7], Function Point models [8] and many others.

Practitioners figured out that the inability to correctly estimate software development costs is a challenging problem. Solving this problem becomes a pressure on IT companies since costs associated with their development became higher than before due to software complexity. As a result, more research focused on gaining a better understanding of the software development life cycle as well as the intelligent techniques which can help in developing accurate and efficient software cost estimation models.

In this paper, we continue exploring the idea of developing evolutionary software effort estimation models based on COCOMO and FP models [9]. Multigene Symbolic Regression GP shall be used to derive a mathematical model in both cases. The models should take in consideration the most important attributes which affect the effort modeling process for both the COCOMO and FP models.

II. Literature Review

Early investigations on using Machine Learning techniques as a tool for software development effort estimation were presented in [10]–[12]. Recently, Machine Learning techniques were also explored to solve the effort and cost estimation problem for software systems. In [10], author explored the use of Neural Networks (NNs), Genetic Algorithms (GAs) and Genetic Programming (GP) to provide a methodology for software cost estimation. A novel soft computing model to increase the accuracy of software development cost estimation was presented in [13]. Authors claims that their proposed NNs model can be interpreted and validated by experts, and has good generalization capability.

CI techniques were presented and analyzed for software cost estimation along with the emerging trends was presented in [14]. A new approach to find architectural design models based on multi-criteria genetic algorithm with optimal performance, reliability, and cost properties was presented in [15]. In [16], author provided a state of the art article on the use of search based approaches for software development effort estimation. The capabilities of these approaches were fully explored and the empirical analysis was carried out. A comparison between Neuro-fuzzy model and the most common software models such as Halstead, WalstonFelix, Bailey-Basili and Doty models was presented in [17].

In [18], author provided an innovative set of models modified from the famous COCOMO model with interesting results. Later on, many authors explored the same idea with some modification [19]–[22] and provided a comparison to the work presented in [18]. Exploration of the advantages of Fuzzy Logic using the Takagi-Sugeno (TS) technique on building a set of linear models over the domain of possible software Kilo Line Of Code (KLOC) were investigated in [23]. Authors in [24], [25] presented an extended work on the use of Particle Swarm Optimization (PSO) and Differential Evolution (DE) to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. The developed PSO model provided promising results. Many model structures were explored including COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models. The potential of the developed COCOMO-PSO and FL models were high compared to other models from the literature.

III. Cost Estimation Models

A. COCOMO Model

COCOMO is one of the most famous software effort estimation model used in the literature. This model was originally developed by Barry Boehm [1], [4] and was extensively revised in [26]. The model is given by Equation 1. Recently, tuning the parameters of the COCOMO model using differential evolution to provide a better effort estimate was presented [25].

$$E = A \times Size^{B} \times EAF \tag{1}$$

Given that:

- E is the effort in person-months
- A is a calibrated constant
- B is a size scale factor
- $Size$ is measured by the Kilo Source Line of Code
- EAF is an Effort Adjustment Factor from cost factor multipliers

Software size may not be the most significant attribute in effort estimation but it does have major influence on the effort and time computation. If we could not accurately estimate the project size it is always hard to plan for project budget and duration. The values of the parameters A and B can be found in Table I. Three types of COCOMO models are presented. They are: Organic, Semidetached and Embedded models [27].

TABLE I. Basic COCOMO models

Model Name	Effort (E)	Time (T)
Organic Model	$E = 2.4(KLOC)^{1.05}$	$T = 2.5(E)^{0.38}$
Semi-Detached Model	$E = 3.0(KLOC)^{1.12}$	$T = 2.5(E)^{0.35}$
Embedded Model	$E = 3.6(KLOC)^{1.20}$	$T = 2.5(E)^{0.32}$

B. Function Point Model

Function points are a well-known concept although only recently they gained wider acceptance as a software size measure [28], [29]. Function points measure software size based on the functionality requested by and provided to the end user. Albrecht's function point gained acceptance during the 1980's and 1990's because of the tempting benefits compared to the models based on the SLOC [30], [31]. Albrecht proposed his model of computing the software size based on the system functionality [32], [33]. Albrecht originally proposed four function types [32]: files, inputs, outputs and inquiries with one set of associated weights and ten General System Characteristics (GSC). In 1983, the work developed in [33], proposed the expansion of the function type, a set of three weighting values (i.e. simple, average, complex) and fourteen General System Characteristics (GSCs) were proposed as given in Table II.

TABLE II. 1983 function types and weights

Function Type	Simple	Average	Complex
External Input	3	4	6
External Output	4	5	7
Internal Files	7	10	15
External Files	5	7	10
External Inquiry	3	4	6

Because FP is self-governing and independent of language type, platform, it can be used to identify many productivity benefits. FP is designed to estimate the time required for a software project development, and thereby the cost of the project and maintaining existing software systems. Because FP is self-governing and independent of language type, platform, it can be used to identify many productivity benefits. FP is designed to estimate the time required for a software project development, and thereby the cost of the project and maintaining existing software systems.

The Albrecht FP model consists of two parts 1) *Unadjusted Function Point* (UFP) and 2) *Adjusted Function Point* (AFP). The UFP consists of five components. They are given in Table II. There are also 14 GSCs factors that affect the size of the project effort, and each is ranked from ”0”- no influence to ”5”- essential. GSCs consists of 14 factors known as $f_1, f_2, \ldots, f_{14}$. These factors are listed in listed in Table III. The sum of all factors is then multiplied given in Equation 2 which constitute the Adjustment Factor (AF) defined in the range [0.65,-1.35]. Then, the Unadjusted FP is then multiplied by the UFP to create the Adjusted Function Point (AFP) count as given in Equation 3. The Adjusted FP value-will is within 35% of the original UFP figure.

$$AF = 0.65 + 0.01 \sum_{i=1}^{14} f_i \tag{2}$$

TABLE III. General System Characteristics (GSCs)

1	Data Communications
2	Distributed Functions
3	Performance
4	Heavily Used Configuration
5	Transaction Rate
6	Online Data Entry
7	End User Efficiency
8	Online Update
9	Complex Processing
10	Reusability
11	Installation Ease
12	Operational Ease
13	Multiple Sites
14	Facilitate Change

$$Adjusted\ FP = Unadjusted\ FP \times AF \tag{3}$$

IV. Genetic Programming

GP is an evolutionary computation technique which allows computer programs to evolve and produce a solution to a problem. GP a biologically inspired machine learning method which randomly generate a population of computer programs (i.e. solutions) represented by a trees structure of LISP expression [34], [35]. Using mutation and crossover, GP produce a new population of solution which is more likely to have better solution than their parents. This process in repeated till the end of certain number of generations or the best solution is reached. GP often use symbolic regression to build a mathematical model or expression based on given data set [36], [37]. The foremost advantages of GP is that; it evolves both the model structure (i.e. function) and the tune the model parameters. This makes GP more suitable to modeling and identification of nonlinear dynamic systems [38]–[42].

A. Multigene Symbolic Regression

Assume we are using GP to develop a model for a system with x inputs and y output. GP can produce a tree structure which introduce the mathematical relationship $y = f(x_1, x_2, \ldots, x_n)$. Given that n is the number of input variables. In multigene symbolic regression, each prediction of the output variable $\hat{y}$ is formed by a weighted output of number of trees/genes in the multigene individual plus a bias term. Each tree is represents a model of zero or more of the given inputs n.

Mathematically, a multigene regression model can be written as:

$$\hat{y} = a_0 + a_1 \times tree_1 + \cdots + a_M \times tree_M \tag{4}$$

where a_0 represents the bias or offset term while $a_1, \ldots, a_M$ are the gene weights and M is the number of genes (i.e. trees) which constitute the available individual. The weights (i.e. regression coefficients) usually computed using least square estimation for each tree. A multigene symbolic model usually consists of one of more gene (i.e. GP tree) weighted by linear combination parameter. An example of multigene model is shown in Figure 1. The presented model can be introduced mathematically as given in Equation 5.

$$\hat{y} = a_0 + a_1[x_1 x_2 + 9x_2] + a_2[22x_1 + x_3] \tag{5}$$

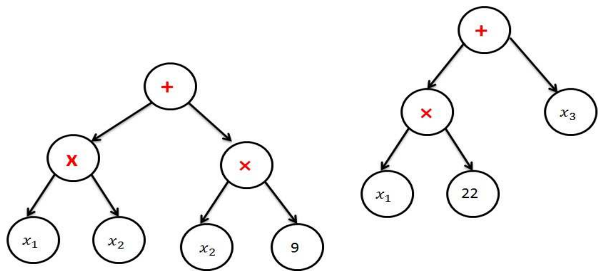

Fig. 1. A pseudo linear multigene model of output $\hat{y}$ along with x_1, x_2 and x_3 as inputs

B. Performance Criterion

The Route Mean Square (RMS) was used as the fitness function for genetic programming. RMS can be described by Equation 6.

$$RMS = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2} \tag{6}$$

Other performance criterion was used to evaluate the goodness of the developed GP model. They are given in following equations:

1) Variance-Accounted-For (VAF):

$$VAF = [1 - \frac{var(y-\hat{y})}{var(y)}] \times 100\% \tag{7}$$

2) Euclidian distance (ED):

$$ED = \sqrt{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2} \tag{8}$$

3) Manhattan distance (MD):

$$MD = (\sum_{i=1}^{n} |y_i - \hat{y}_i|) \tag{9}$$

4) Mean Magnitude of Relative Error (MMRE):

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} \frac{|y_i - \hat{y}_i|}{y_i} \tag{10}$$

where y and $\hat{y}$ are the actual and the estimated effort based on the developed GP model.

V. Experimental Results

A. Experimental Setup

To develop the proposed GP effort estimation model we used the GPTIPS [43] toolbox with the setting parameters given in Table IV. The default GPTIPS multigene symbolic regression function was used in order to minimize the root mean squared (RMS) prediction error on the training data. The default recombination operator probabilities were used as 0.85 for crossover, 0.1 for mutation and direct reproduction of 0.05.

TABLE IV. Tuning parameters of the GPTIPS Toolbox

Parameter	Value
Population size	25
Number of generations	300
Tournament size	10
Elitism	0.05
Maximum depth of trees	10
Maximum No. of genes	7;
function node set	+, -, ×

B. GP effort model as a function of SLOC

The famous COCOMO model always used the SLOC as main input to develop the effort equation as given in Equation 1. In our case, we adopted the COCOMO model as a basis for our development. Thus, we used the SLOC as an input and the effort as an output. The developed model is given in Equation 11 where x_1 stands for the SLOC.

$$\begin{aligned} y &= -\left(8.33\cdot10^{-9}\right)\mathrm{x_1}^5+\left(4.036\cdot10^{-6}\right)\mathrm{x_1}^4\\ &- 0.0005124\,\mathrm{x_1}^3+0.02337\,\mathrm{x_1}^2\\ &- 0.08683\,\mathrm{x_1}+1.401 \end{aligned} \tag{11}$$

In Figure 2, we show the GP convergence process where the best RMS was measured as 6.3475 which were received at generation 292. Figure 3 shows the actual and estimated effort using GP over the sorted list of projects. The characteristics between the two curves look very similar with high VAF criteria. In Table V, we show the values of each evaluation criteria adopted in this study.

TABLE V. GP BASED SLOC MODEL: PERFORMANCE CRITERION

VAF	ED	MD	MMRE
94.794%	31.097	100.27	0.48502

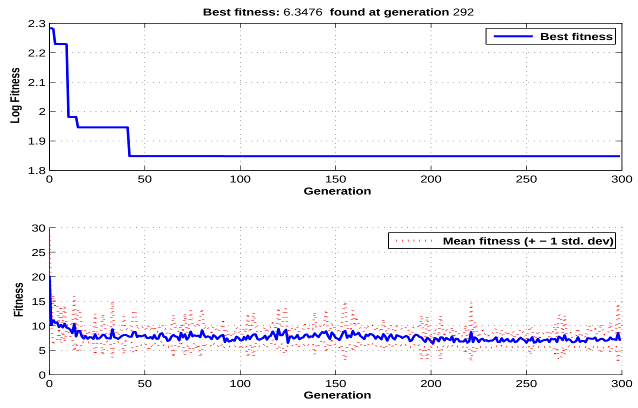

Fig. 2. GP convergence process in the effort estimation case

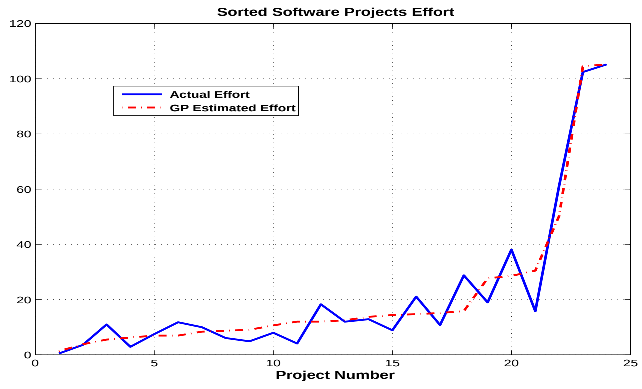

Fig. 3. Actual and estimated GP effort using SLOC Model

C. GP Effort model based FP

In this case, we are developing a GP model based the FP. The FP model utilizes the Inputs (x_1), Outputs (x_2), Files (x_3), and User Inquiries (x_4) to estimate the Function Point (FP). Thus, we considered these attributes as input to our model and the number of FP as an output. We run the GPTIPS [43] toolbox with the setting parameters given in Table IV. The developed GP model for the FP is given in Equation 12.

TABLE VI. ACTUAL AND ESTIMATED EFFORT USING GP

SLOC	Effort	GP-Effort
3	0.5	1.3373
15	3.6	3.8261
20	11	5.5339
22	2.9	6.2505
24	7.5	6.9704
24	11.8	6.9704
28	10	8.386
29	6.1	8.7295
30	4.9	9.0674
35	8	10.651
40	4.1	12.023
40	18.3	12.023
42	12	12.508
48	12.9	13.748
52	8.9	14.421
54	21.1	14.723
57	10.8	15.148
62	28.8	15.832
93	19	27.72
94	38.1	28.597
96	15.8	30.497
110	61.2	50.263
130	102.4	104.48
318	105.2	105.2

$$\begin{aligned} y &= 3.713\,\mathrm{x_2}-10.29\,\mathrm{x_1}+4.939\,\mathrm{x_3}+3.682\,\mathrm{x_4}\\ &+ 0.1965\,\mathrm{x_1}\,\mathrm{x_2}+1.011\,\mathrm{x_1}\,\mathrm{x_3}+0.2481\,\mathrm{x_1}\,\mathrm{x_4}\\ &- 0.1838\,\mathrm{x_2}\,\mathrm{x_3}+0.006156\,\mathrm{x_2}\,\mathrm{x_4}-0.3135\,\mathrm{x_3}\,\mathrm{x_4}\\ &- 0.006156\,\mathrm{x_1}\,\mathrm{x_3}^2+0.006687\,\mathrm{x_2}\,\mathrm{x_3}^2\\ &- 0.001063\,\mathrm{x_3}\,\mathrm{x_4}^2-0.002834\,\mathrm{x_3}^2\,\mathrm{x_4}\\ &+ 0.003078\,\mathrm{x_2}^2-0.2509\,\mathrm{x_3}^2-0.001063\,\mathrm{x_3}^3\\ &+ 0.0001771\,\mathrm{x_1}\,\mathrm{x_2}\,\mathrm{x_3}^2-0.0001771\,\mathrm{x_1}\,\mathrm{x_3}^2\,\mathrm{x_4}\\ &- 0.01298\,\mathrm{x_1}\,\mathrm{x_2}\,\mathrm{x_3}-0.003078\,\mathrm{x_1}\,\mathrm{x_3}\,\mathrm{x_4}\\ &+ 0.003255\,\mathrm{x_2}\,\mathrm{x_3}\,\mathrm{x_4}+158.6 \end{aligned} \tag{12}$$

In Figure 4, we show the GP convergence process where the best RMS found was 30.8868 which were received at generation 297. Figure 5 shows the actual and estimated effort using GP based Albrecht data set adopted in this study. The developed model's performance were computed using number of criteria reported in Table VIII.

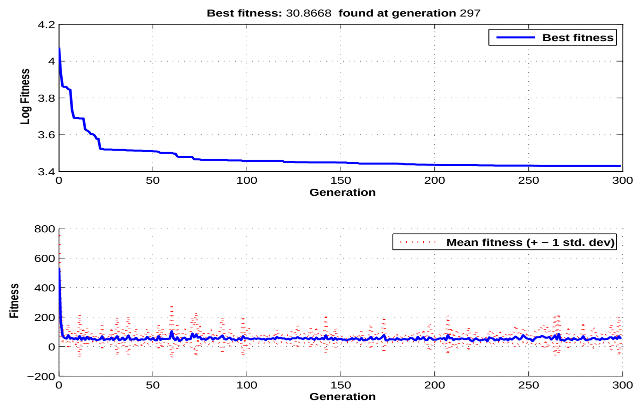

Fig. 4. GP convergence process in FP estimation case

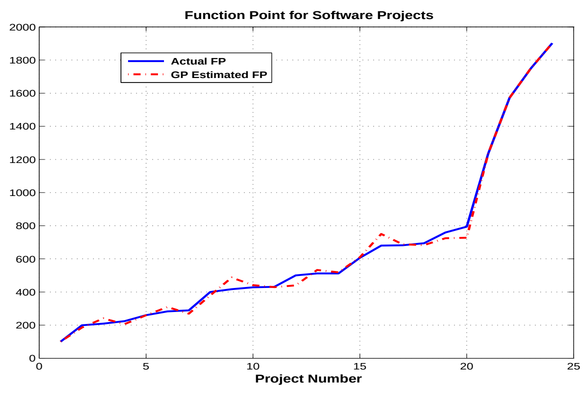

Fig. 5. Actual and estimated GP Function Points results

TABLE VII. ACTUAL AND ESTIMATED GP NUMBER OF FP

Inputs	Outputs	Files	Inquiries	FP	GP-FP
34	14	5	0	100	100.54
15	15	3	6	199	185.25
7	12	8	13	209	240.35
33	17	5	8	224	206
12	15	15	0	260	259.65
13	19	23	0	283	309.26
17	17	5	15	289	269.24
27	20	6	24	400	378.86
28	41	11	16	417	486.84
70	27	12	0	428	441.28
10	69	9	1	431	429.19
25	28	22	4	500	439.86
41	27	5	29	512	532.66
28	38	9	24	512	518.51
42	57	5	12	606	609.39
45	64	16	14	680	750.1
43	40	35	20	682	687.45
61	68	11	0	694	682.89
40	60	12	20	759	724.64
40	60	15	20	794	727.36
48	66	50	13	1235	1234.9
69	112	39	21	1572	1571.7
25	150	60	75	1750	1750.2
193	98	36	70	1902	1901.9

VI. Conclusions and Future Work

In this paper, an evolutionary software effort estimation models based Multigene Symbolic Regression Genetic Programming were developed. Two GP based models were developed; one model considered the Source Line Of Code (SLOC) as input variable to estimate the Effort (E); while the second model considered the Inputs, Outputs, Files, and User Inquiries to estimate the Function Point (FP). The proposed GP models show better performance compared to other reported models in the literature. They were tested using the Albrecht data set reported in [12]. The mathematical equation which represents both models is adequately simple and can be easily used to predict further project's effort. These types of models significant help project managers to estimate time and cost for future developments.

Acknowledgments

Dr. Sultan Aljahdali would like to acknowledge the financial support of King Abdulaziz City for Science and Technology and Taif University during his short research visit to Prof. Jayantha Herath with the Computer Science Department, St. Cloud State University, St. Cloud, MN 56301, USA on the Summer of 2013.

TABLE VIII. GP BASED FP MODEL: PERFORMANCE CRITERION

VAF	ED	MD	MMRE
99.59%	151.22	20.63	0.048274

References

[1] B. Boehm, *Software Engineering Economics*. Englewood Cliffs, NJ, Prentice-Hall, 1981.

[2] T. S. Group, *CHAOS Chronicles*. PhD thesis, Standish Group Internet Report, 1995.

[3] P. Suri and P. Ranjan, "Article: Comparative analysis of software effort estimation techniques," *International Journal of Computer Applications*, vol. 48, pp. 12–19, June 2012. Published by Foundation of Computer Science, New York, USA.

[4] B. Boehm, *Cost Models for Future Software Life Cycle Process: COCOMO2*. Annals of Software Engineering, 1995.

[5] L. Putnam and W. Myers, *Measures for excellence*. Yourdon Press Computing Series, 1992.

[6] L. Putnam, "A general empirical solution to the macro software sizing and estimation problem," *IEEE Transactionson Software Engineering*, vol. 4, no. 4, pp. 345–381, 1978.

[7] R. Jensen, "An improved macrolevel software development resource estimation model," in *In Proceedings of 5th Conference of International S Parametric Analysts*, pp. 88–92, 1983.

[8] J. E. Matson, B. E. Barret, and J. M. Mellinchamp, "Software develeopemnet cost estimation using function points," *IEEE Trans. Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[9] A. Sheta and S. Aljahdali, "Software effort estimation inspired by COCOMO and FP models: A fuzzy logic approach," *International Journal of Advanced Computer Science and Applications*, vol. 4, Dec. 2013.

[10] M. A. Kelly, "A methodlolgy for software cost estimation using machine learning techniques," Master's thesis, Naval Postgratuate School, Monterey, California, 1993.

[11] G. R. Finnie and G. E. Wittig, "AI tools for software development effort estimation," in *Proceedings of the 1996 International Conference on Software Engineering: Education and Practice (SE:EP '96)*, SEEP '96, (Washington, DC, USA), pp. 346–, IEEE Computer Society, 1996.

[12] C. Schofield, *An Empirical Investigation into Software Effort Estimation by Analogy*. PhD thesis, Bournemouth University, 1998.

[13] I. Attarzadeh and S. H. Ow, "A novel soft computing model to increase the accuracy of software development cost estimation," in *2010 The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, vol. 3, pp. 603–607, 2010.

[14] T. R. Benala, S. Dehuri, and R. Mall, "Computational intelligence in software cost estimation: an emerging paradigm," *SIGSOFT Software Engineering Notes*, vol. 37, pp. 1–7, May 2012.

[15] A. Martens, H. Koziolek, S. Becker, and R. Reussner, "Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms," in *Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering*, WOSP/SIPEW '10, (New York, NY, USA), pp. 105–116, ACM, 2010.

[16] F. Sarro, "Search-based approaches for software development effort estimation," in *Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement*, Profes '11, (New York, NY, USA), pp. 38–43, ACM, 2011.

[17] U. Saxena and S. P. Singh, "Software effort estimation using neuro-fuzzy approach," in *2012 CSI Sixth International Conference on Software Engineering (CONSEG)*, pp. 1–6, 2012.

[18] 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.

[19] H. Mittal and P. Bhatia, "A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers," *International Journal of Computer Science and Security*, vol. 1, no. 4, pp. 36–47, 2007.

[20] H. Mittal and P. Bhatia, "Optimization criteria for effort estimation using fuzzy technique," *CLEI Electronic Journal*, vol. 10, no. 1, pp. 1–11, 2007.

[21] M. Uysal, “Estimation of the effort component of the software projects using simulated annealing algorithm,” in *World Academy of Science, Engineering and Technology*, vol. 41, pp. 258–261, 2008.

[22] P. S. Sandhu, M. Prashar, P. Bassi, and A. Bisht, “A model for estimation of efforts in development of software systems,” in *World Academy of Science, Engineering and Technology*, vol. 56, pp. 148–152, 2009.

[23] A. Sheta, “Software effort estimation and stock market prediction using takagi-sugeno fuzzy models,” in *Proceedings of the 2006 IEEE Fuzzy Logic Conference, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21*, pp. 579–586, 2006.

[24] A. Sheta, D. Rine, and A. Ayesh, “Development of software effort and schedule estimation models using soft computing techniques,” in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong, 1-6 June*, pp. 1283–1289, 2008.

[25] S. Aljahdali and A. Sheta, “Software effort estimation by tuning COOCMO model parameters using differential evolution,” in *2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, pp. 1–6, 2010.

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

[27] O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, “COCOMO based effort estimation for iterative and incremental software development,” *Software Quality Journal*, vol. 11, pp. 265–281, 2003.

[28] J. Moses, “Measuring effort estimation uncertainty to improve client confidence,” *Software Quality Control*, vol. 10, pp. 135–148, Sept. 2002.

[29] J. Wu and X. Cai, “A software size measurement model for large-scale business applications,” in *Proceedings of the 2008 International Conference on Computer Science and Software Engineering - Volume 02*, CSSE ’08, (Washington, DC, USA), pp. 39–42, IEEE Computer Society, 2008.

[30] R. Rask, P. Laamanen, and K. Lyytinen, “A comparison of albrecht’s function point and symons’ mark ii metrics,” in *Proceedings of the thirteenth international conference on Information systems*, ICIS ’92, (Minneapolis, MN, USA), pp. 207–221, University of Minnesota, 1992.

[31] S. Furey, “Why we should use function points [software metrics],” *IEEE Software*, vol. 14, pp. 28, 30–, Mar. 1997.

[32] A. J. Albrecht, “Measuring application development productivity,” in *Proceedings of the Joint SHARE, GUIDE, and IBM Application Developments Symposium*, pp. 83–92, 1979.

[33] A. J. Albrecht and J. E. Gaffney, “Software function, source lines of code, and development effort prediction: A software science validation,” *IEEE Transactions on Software Engineering*, vol. 9, no. 6, pp. 639–648, 1983.

[34] J. Koza, “Evolving a computer program to generate random numbers using the genetic programming paradigm,” in *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, La Jolla,CA, 1991.

[35] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, 1992.

[36] J. R. Koza, ed., *Genetic Algorithms and Genetic Programming at Stanford 2003*. Stanford, California, 94305-3079 USA: Stanford University Bookstore, June 2003.

[37] J. R. Koza, “Human-competitive results produced by genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 11, pp. 251–284, Sept. 2010.

[38] A. Sheta and A. Al-Afeef, “A GP effort estimation model utilizing line of code and methodology for NASA software projects,” in *2010 10th International Conference on Intelligent Systems Design and Applications (ISDA)*, pp. 290–295, 2010.

[39] H. Faris, M. Al-kasassbeh, and A. Sheta, “Predicting stock abundance of the barents sea capelin using genetic programming,” *International Review on Computers and Software (IRECOS)*, vol. 7, no. 4, 2012.

[40] H. Faris and A. Sheta, “Identification of the tennessee eastman chemical process reactor using genetic programming,” *International Journal of Advanced Science and Technology*, vol. 50, pp. 121–140, Jan. 2013.

[41] H. Faris, A. Sheta, and E. Oznergiz, “Modeling hot rolling manufacturing process using soft computing techniques,” *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 8, 2013.

[42] A. Sheta, R. Hiary, H. Faris, and N. Ghatasheh, “Optimizing thermostable enzymes production using multigene symbolic regression genetic programming,” *World Applied Sciences*, vol. 22, no. 4, pp. 485–493, 2013.

[43] D. P. Searson, D. E. Leahy, and M. J. Willis, “GPTIPS : An open source genetic programming toolbox for multigene symbolic regression,” in *Proceedings of the International Multi-conference of Engineers and Computer Scientists*, vol. 1, (Hong Kong), pp. 77–80, 17-19 Mar. 2010.