A GP Effort Estimation Model Utilizing Line of Code and Methodology for NASA Software Projects

Alaa F. Sheta
Computers and Systems Department
Electronics Research Institute (ERI)
Cairo, Egypt
asheta66@gmail.com

Alaa Al-Afeef
Information Technology Department
Al-Balqa Applied University (BAU)
Salt, Jordan
alaa.afeef@gmail.com

***Abstract*—There is still an urgent need of finding a mathematical model which can provide an accurate relationship between the software project effort/cost and the cost drivers. A powerful algorithm which can optimize such a relationship via developing a mathematical relationship between model variables is urgently needed. In this paper, we explore the use of GP to develop a software cost estimation model utilizing the effect of both the developed line of code and the used methodology during the development. An application of estimating the effort for some NASA software projects is introduced. The performance of the developed Genetic Programming (GP) based model was tested and compared to known models in the literature. The developed GP model was able to provide good estimation capabilities compared to other models.**

***Keywords*-Software Cost Estimation; Software Engineering; Genetic Programming; NASA Software;**

I. Introduction

A software project manager should be able to accurately estimate the overall project costs, duration, required man power and schedule [1]. He must be able to fairly distribute the resources over time such that the project could be finished on time and within budget. It was found that there are many similarities between the process of managing project resources and system modeling. In system modeling we need to develop some sort of a relationship between the system input and output such that the system function is approximated in a form of a model. The model can be used for simulation and performance evaluation of the original system under various operating conditions. In project management, the manager need to collect enough data about various attributes which affect the quality and the cost of a project. These collected data helps in developing a plan or a model for cost distribution over various phases of a project. The developed model can be calibrated in each phase of the project to meet the project goals, the quality of the product and the available resources.

A. Software effort estimation

Software effort estimation process has a similar nature since it is part of project management. In this case, the objective is to develop a sort of relationship between the expected Developed (DL) Line Of Code of a project as an input variable and the expected effort required to implement this project in man-month. A famous effort DL-E relationship [2], [3] known as the COnstructive COst MOdel (COCOMO) is give as in Equation 1.

$$E = a(DL)^b \tag{1}$$

The DL include all program instructions and formal statements [4]. 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. They are: 1) Organic 2) Semidetached and 3) Embedded. COCOMO model was first provided by Boehm [2], [5]. This model was built based on 63 software projects. The model helps is defining mathematical equations that identify the the cost, schedule and quality of a software product. The estimation accuracy is significantly improved when adopting models such as the Intermediate and Complex COCOMO models [2]. Extensions of COCOMO, such as COMCOMO II, can be found in [3].

Typical models for software effort estimation are given in Table I. These models have been derived by studying large number of completed software projects from various organizations and applications to explore how project sizes mapped into project effort.

Table I
KNOWN EFFORT ESTIMATION MODELS.

Model name	Model equation
Halstead	$E = 5.2(DL)^{1.50}$
Walston-Felix	$E = 0.7(DL)^{0.91}$
Bailey-Basili	$E = 5.5 + 0.73(DL)^{1.16}$
Doty (for DL > 9)	$E = 5.288(DL)^{1.047}$

B. Previous Work

In the past, most of the proposed models used to solve the software cost estimation modeling problem are linear in nature. It was found that dealing with a linear model makes it easier to use techniques such as least square estimation

978-1-4244-8136-1/10/$26.00 ©2010 IEEE

(LSE) or Instrumental Variable method to identify the parameters of the given model. In the other case, if the actual model is nonlinear, attempting to approximate this structure with a linear model cannot guarantee the accuracy of the model. In solving the software cost estimation problem, it is important to develop models using a small number of measurements and in the presence of measurement noise.

Recently, many questions were introduced about the applicability of using Soft Computing and Machine Learning Techniques to solve the effort and cost estimation problem for software systems. In [6], author presented a detailed study on using number of techniques such as genetic programming and neural networks to estimate software project effort. Author concluded that GP can perform well on handling such a problem. In [7], 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 [8]–[11] and provided a comparison to the work presented in [7]. In [12], author used Particle Swarm Optimization (PSO) to tune the parameters of the famous COnstructive COst MOdel (COCOMO). They also explored the advantages of Fuzzy Logic to build a set of linear models over the domain of possible software Line Of Code (LOC). The performance of the developed model was evaluated using NASA software projects data set.

In this paper, an evolutionary approach, Genetic Programming (GP), is used to fit nonlinear models to a dataset of some NASA software projects, aiming to improve the prediction of software effort for NASA software projects.

In this paper, Genetic Programming is used to develop an effort estimation model for software systems due to the advantages of GP as provided in Section II-A. The theoretical foundations of genetic programming are summarized in [13].

In the following Section II, GP is introduced briefly. The experiment setup and control parameters for the application of GP in evolution of software development effort estimation programs is discussed in Section III and the developed results in Section IV. This includes data preparation, GP details and results obtained. A comparison of related developed results are presented in Section V. Section VI draws the conclusions and future work.

II. Genetic Programming

Genetic programming (GP) is an evolutionary computation (EC) technique that automatically searches for an optimal solution of a problem without requiring the user to know or specify the form or structure of the solution in advance [14], [15]. GP technique has been successfully applied to solve large number of difficult problems, such as modeling of industrial processes [16], [17], forecasting of river flow [18] and image reconstruction [19].

A. Strengths of GP

Evolutionary algorithms have been found 'experimentally' efficient in finding solutions to the Modeling problems. GP is considered one of the evolutionary algorithms that hold all advantage offered by evolutionary algorithms and adds several more. The advantages offered by GP for Modeling can be summarized as:

- GP is a global search technique that makes use of hyper plane search which, makes it less likely to get stuck in the local optimum. This is different from other techniques such as neural networks and gradient descent which are prone to local optimal values.
- GP has the benefits of variety in solution structures unlike most of the evolutionary algorithms that has fixed size solutions such as genetic algorithms or fixed architectures such as neural networks [20].
- GP can automatically eliminate unrelated attributes of the Modeling problem performing the task of feature extraction algorithm [20] in which important attributes can appear near the root while less important ones would appear deeper in the tree [21].
- GP is able to operate on portion of data to extract significant rules. There is no need to use all of the training data to develop models [20].
- GP are like white boxes that clearly sketch the relationships between attributes, as opposed to many other black box solutions like neural networks [22].
- GP has the ability to operate upon the data in its original form. No pre-processing or data transformations are usually required to apply GP for modeling task.
- GP based evolution is not affected by the data distribution [20]. This is in contrast to the neural networks which are highly dependent on the data distribution. This autonomy enables efficient discovery of unknown knowledge from the data.

B. Representation in GP

In GP, programs are usually represented as a variable sized tree structure. This type of representation allows a variety of models to be developed. A tree consists of nodes and terminals. In every terminal node, there is an operand and in every node there is a function. Trees can be easily evaluated in a recursive manner. This way we can evolve mathematical models in a very simple way such as in programming using Lisp language [23]. Such a representation is simple and has been used frequently for the data classification and modeling problems. A simple tree structure can be presented in Figure 1 as described in Equation (2).

$$E = 1.7 \cdot DL - ME \tag{2}$$

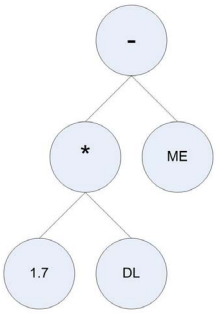

Figure 1. Example of GP tree structure

C. Preparatory Steps of GP

Before applying the Evolutionary Process, as in Figure (2), four major preparatory steps require to be specified [14], [15]:

1) The definition of the function and terminal set (primitive set) for a particular problem.
2) Fitness measure for the problem. This specifies what needs to be done.
3) The control parameters for the run (for example, population size, max generations and maximum tree depth).
4) The termination criterion which may include a maximum number of generations to be run as well as a problem-specific optimum solution.

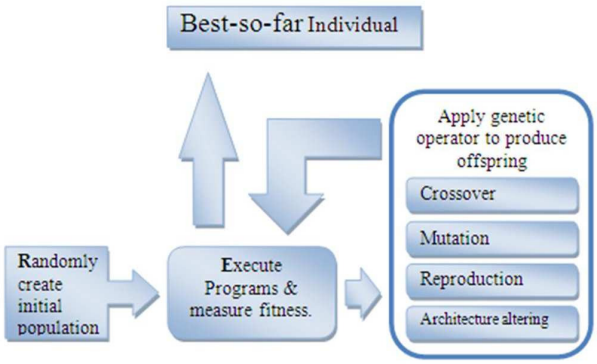

Figure 2. GP evolutionary process

D. Evaluation Criteria

In order to check the performance of the developed models, two evaluation criteria will be adopted. We compute the Variance-Accounted-For (VAF) performance criterion to measure how close the measured values to the values developed using the fuzzy models. Given that $E, \hat{E}$ are the actual effort and the estimated effort, respectively. The VAF is computed as follows:

$$VAF = [1 - \frac{var(E - \hat{E})}{var(E)}] \times 100\% \quad (3)$$

The Mean Magnitude of Relative Error (MMRE) as the main performance measure was also used in many articles [12], [24]. MMRE is defined as:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|E - \hat{E}|}{|E|} \quad (4)$$

We will also adopt these two criteria's for evaluating the cost estimation models investigated here.

III. Experiment Setup and Control Parameters

GP Setup (Table II) is adapted for modeling the problem under study. The adopted control parameters are shown in Table IV and Table V according to [14].

Table II
GP Experiment Setup for the Effort Estimation Problem

Objective	Find a function of 2 independent variable [Line Of Code (DL), Methodology(ME)] and one dependent variable [Effort (E)], in symbolic form, that fits a given Training sample of the form (DL, ME, E) data points.
Terminal set	DL, ME (the independent variables).
Function set	+, -, *
Fitness criteria	The fitness is the absolute value of the difference between the estimated values produced by GP and the target value of the effort. ($\|E^i_{Target} - E^i_{Estimated}\|$).
Raw fitness	The sum taken over the fitness cases (N) ($\sum_{i=1}^{N} E^i_{Target} - E^i_{Estimated}$)
Standardized fitness	Equals raw fitness divided by the count of fitness cases.
Hits	Number of fitness cases for which the value of the dependent variable produced by the GP comes within 0.001 of the target value.

IV. Experimental Results

Experiments have been conducted on a data set presented by Bailey and Basili [25] to explore strengthen of the developed GP based model. The dataset consist of the following variables:

- Developed Line of Code (DL)
- Methodology (ME) and
- Effort (E) in man-month.

The dataset is presented in Table III. The data was split to two sets training (i.e. 13 projects) and testing/validation (i.e. 5 projects). We used Lilgp1.1 [26] (C language package for developing genetic programming applications) to produce our results. Lilgp is well-known to be a fast, memory efficient and well documented GP tool that provides support for several features not typically found in other GP systems, such as the support of parallel processing.

Table III
SORTED NASA SOFTWARE PROJECT DATA.

Project No.	DL	ME	Effort E
1	2.1	28	5.0
2	3.1	26	7.0
3	4.2	19	9.0
4	5.0	29	8.4
5	7.8	31	7.3
6	9.7	27	15.6
7	10.5	34	10.3
8	12.5	27	23.9
9	12.8	26	18.9
10	21.5	31	28.5
11	31.1	35	39.6
12	46.2	20	96.0
13	46.5	19	79.0
14	54.5	20	90.8
15	67.5	29	98.4
16	78.6	35	98.7
17	90.2	30	115.8
18	100.8	34	138.3

A. GP Effort Model based DL

The developed GP model should be able to significantly generalize the computation of the developed effort for all projects. We run GP to develop a new software effort estimation model. The Lisp expression developed using Lilgp1.1 [26] program is presented in Equation 5 and simplified in Equation 6.

$$(*(-(+(*1.35730 DL)1.75992)(*1.36186 DL))DL) \quad (5)$$

$$E = 1.75992 \cdot DL - 4.56 \cdot 10^{-3} DL^2 \quad (6)$$

We run GP with various population sizes (i.e. 1000, ..., 9000). The convergence process for all runs were measured and the best so far curves are presented in Figure (3). It is shown that all curves convergence to the same optimal value for the fitness criteria. The rest of the tuning parameters for the Lilgp experimental setup is given in Table IV. Figures (4) show the measured and estimated GP effort.

B. GP Effort Model based DL and ME

GP was used to find the model structure which describe the relationship between the effort and both the developed line of code and the methodology. We run GP was various population sizes to explore the possibility of having a good model structure which better estimate the software effort. the tuning parameters for the GP evolutionary process is presented in Table V. The Lisp expression developed using Lilgp1.1 program is presented in Equation 7 and simplified in Equation 8. The convergence process for GP is presented in Figure (5). GP convergence to the best possible model with a good prediction capabilities. Figures (6) show the measured and estimated GP effort.

$$(-(+DL\,DL)(*(*(*(*0.022970.02588)ME)DL)ME)) \quad (7)$$

$$E = 2 \cdot DL - 0.59 \cdot 10^{-3} ME^2 \cdot DL \quad (8)$$

Table IV
LILGP EXPERIMENTAL SETUP FOR DL BASED MODEL

Parameter	Value
Max generations	100
Max tree depth	5
Max tree nodes	11
Initial tree depth	2-4
Crossover rate	0.8
Reproduction rate	0.1
Mutation rate	0.1
selection method	fitness_overselect

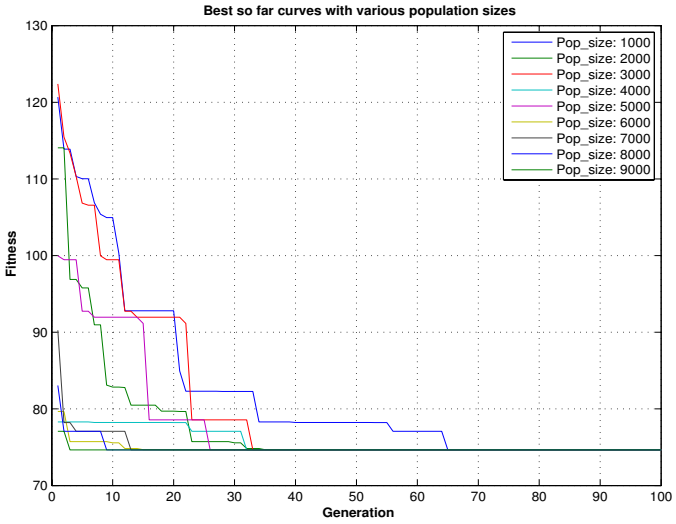

Figure 3. Convergence of GP with various population sizes for DL

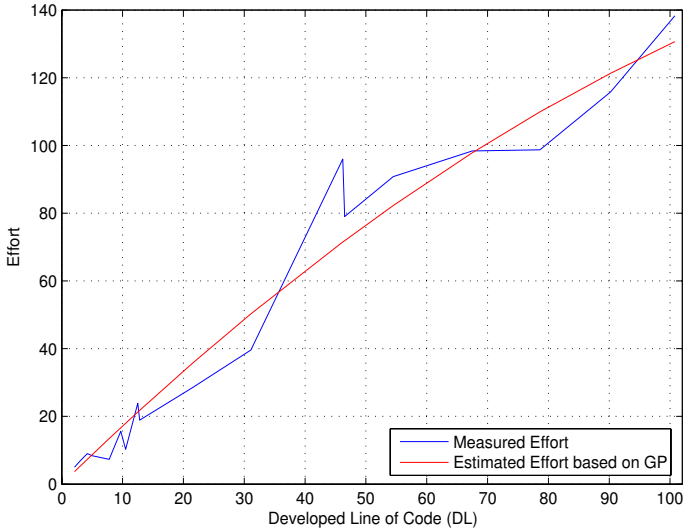

Figure 4. Measured and Estimated effort Using GP Based DL Model

Table V
LILGP EXPERIMENTAL SETUP FOR DL-ME BASED MODEL

Parameter	Value
Max generations	100
Max tree depth	5
Max tree nodes	13
Initial tree depth	2-5
Crossover rate	0.8
Reproduction rate	0.1
Mutation rate	0.1
selection method	fitness_overselect

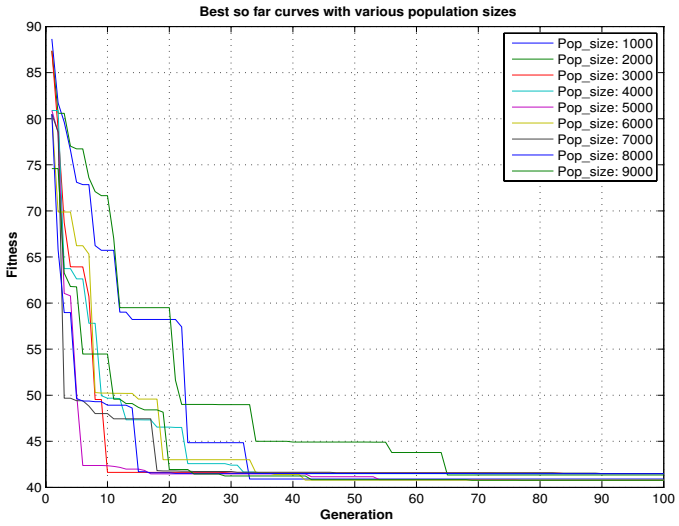

Figure 5. Convergence of GP with various population sizes for DL-ME

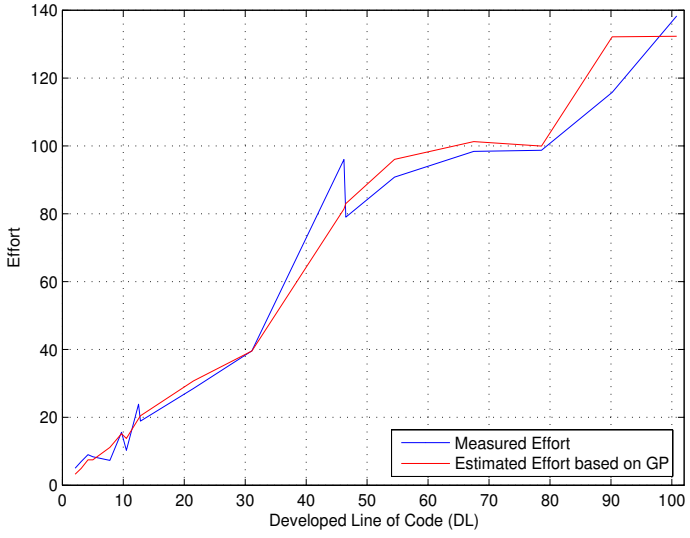

Figure 6. Measured and Estimated effort Using GP Based DL-ME Model

Table VI
THE COMPUTED PERFORMANCE OF THE DEVELOPED GP MODELS

Model	VAF	MMRE
DL based Model	96.5538	0.0052
DL-ME based Model	98.2346	0.0039

V. COMPARISON WITH OTHER MODELS

The computed performance of the developed models is presented in Table VI. The computed VAF is high in the case of the DL-ME based model and better than the case of the DL based model. This is an evidence that the inclusion in the ME as a variable in the modeling process of the effort enhance the model capabilities to better estimate the effort. Thus, GP was able to better find the function f which related the E and both DL and ME, $E = f(DL, ME)$.

In [12], authors presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. A comparison between COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided. In Table VII, we show the MMRE criteria computed overall data set. It is shown that the GP and the COCOMO based PSO models have almost similar properties. The FL model is the model found to provide the minimum MMRE since it consists of three linear models with various membership functions. This gives an advantage of the FL model over other effort estimation models.

VI. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a new model structure to estimate the software effort for projects sponsored by NASA using genetic programming. The performance of the developed GP model was tested on NASA software projects data presented in [25]. The developed software effort estimation model based GP was capable of providing good effort estimation as compared to other known model in the literature such as Halstead, Walston-Felix, Bailey-Basili and Doty models. the consideration of other attributes such as the Methodology while developing the effort most significantly improves the model prediction capabilities. GP was able to provide an advanced mathematical function utilizing the DL and ME such that the computed effort is more accurate.

REFERENCES

[1] C. F. Kemere. An empirical validation of software cost estimation models. *Communication ACM*, 30:416–429, 1987.

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

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

Table VII
THE COMPUTED MMRE CRITERION FOR ALL MODELS BASED DL ONLY

Model	Fuzzy Model	GP Model	COCOMO based PSO	Walston-Felix Model	Bailey-Basili Model	Halstead Model	Doty Model
MMRE	0.0046	0.0052	0.0074	0.0822	0.0095	0.1479	0.1848

[4] Tim Menzies, Dan Port, Zhihao Chen, Jairus Hihn, and Sherry Stukes. Validation methods for calibrating software effort models. In *Proceedings of the 27th international conference on Software Engineering (ICSE'05)*, pages 587–595, New York, NY, USA, 2005. ACM Press.

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

[6] Martin Lefley and Martin J. Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In *GECCO'03: Proceedings of the 2003 international conference on Genetic and evolutionary computation*, pages 2477–2487, Berlin, Heidelberg, 2003. Springer-Verlag.

[7] A. F. Sheta. Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects. *Journal of Computer Science*, 2(2):118–123, 2006.

[8] Harish Mittal and Pradeep Bhatia. A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers. *International Journal of Computer Science and Security*, 1(4):36–47, 2007.

[9] Harish Mittal and Pradeep Bhatia. Optimization criteria for effort estimation using fuzzy technique. *CLEI ELECTRONIC JOURNAL*, 10(1):1–11, 2007.

[10] Mitat Uysal. Estimation of the effort component of the software projects using simulated annealing algorithm. In *World Academy of Science, Engineering and Technology*, volume 41, pages 258–261, 2008.

[11] Parvinder S. Sandhu, Manisha Prashar, Pourush Bassi, and Atul Bisht. A model for estimation of efforts in development of software systems. In *World Academy of Science, Engineering and Technology*, volume 56, pages 148–152, 2009.

[12] Alaa Sheta, David Rine, and Aladdin 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*, pages 1283–1289, 2008.

[13] W. B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

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

[15] Riccardo Poli, William B. Langdon, and Nicholas Freitag McPhee. *A field guide to genetic programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[16] A. Hussian, A. Sheta, M. Kamel, M. Telbany, and A. Abdelwahab. Modeling of a winding machine using genetic programming. In *Proceedings of the Congress on Evolutionary Computation (CEC2000)*, pages 398–402, 2000.

[17] A. Sheta and J. Gertler. Modeling the dynamics of an automotive engine using genetic programming. In *Proceedings of the International Symposium on Engineering of Natural and Artificial Intelligent Systems (ENAIS2001), American University in Dubai, U.A.E.*, 2000.

[18] A. Sheta and A. Mahmoud. Forecasting using genetic programming. In *Proceedings of the 33 rd Southern Symposium on System Theory, March 19-20, Athens, Ohio, USA*, pages 343–347, 2001.

[19] Alaa. S. Al-Afeef. Image reconstructing in electrical capacitance tomography of manufacturing processes using genetic programming. Master's thesis, Al-Balqa Applied University, July 2010.

[20] J. K. Kishore, L. M. Patnaik, V. Mani, and V. K. Agrawal. Application of genetic programming for multicategory pattern classification. *IEEE Transactions on Evolutionary Computation*, 4(3):242–258, September 2000.

[21] Sean Luke. Code growth is not caused by introns. In Darrell Whitley, editor, *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*, pages 228–235, Las Vegas, Nevada, USA, 8 July 2000.

[22] S. E. Rouwhorst and A. P. Engelbrecht. Searching the forest: Using decision trees as building blocks for evolutionary search in classification databases. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, volume 1, pages 633–638, La Jolla Marriott Hotel La Jolla, California, USA, 6-9 July 2000. IEEE Press.

[23] Stuart C. Shapiro (Editor). *Encyclopedia of Artificial Intelligence*. John Wiley, 2 edition, January 1992. 1792 pages.

[24] Alaa 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*, pages 579–586, 2006.

[25] J. W. Bailey and V. R. Basili. A meta model for software development resource expenditure. In *Proceedings of the International Conference on Software Engineering*, pages 107–115, 1981.

[26] Douglas Zongker and Bill Punch. lilgp 1.01 user's manual. Technical report, Michigan State University, USA, 26 March 1996.