1

Computational Intelligence in Software Cost Estimation: Evolving Conditional Sets of Effort Value Ranges

Efi Papatheocharous and Andreas S. Andreou
Department of Computer Science, University of Cyprus, Cyprus

1. Introduction

In the area of software engineering a critical task is to accurately estimate the overall project costs for the completion of a new software project and efficiently allocate the resources throughout the project schedule. The numerous software cost estimation approaches proposed are closely related to cost modeling and recognize the increasing need for successful project management, planning and accurate cost prediction. Cost estimators are continually faced with problems stemming from the dynamic nature of the project development process itself. Software development is considered an intractable procedure and inevitably depends highly on several complex factors (e.g., specification of the system, technology shifting, communication, etc.). Normally, software cost estimates increase proportionally to development complexity rising, whereas it is especially hard to predict and manage the actual related costs. Even for well-structured and planned approaches to software development, cost estimates are still difficult to make and will probably concern project managers long before the problem is adequately solved.

During a system's life-cycle, one of the most important tasks is to effectively describe the necessary development activities and estimate the corresponding costs. This estimation, once successful, allows software engineers to optimize the development process, improve administration and control over the project resources, reduce the risks caused by contingencies and minimize project failures (Lederer & Prasad, 1992). Subsequently, a commonly investigated approach is to accurately estimate some of the fundamental characteristics related to cost, such as effort and schedule, and identify their inter-associations. Software cost estimation is affected by multiple parameters related to technologies, scheduling, manager and team member skills and experiences, mentality and culture, team cohesion, productivity, project size, complexity, reliability, quality and many more. These parameters drive software development costs either positively or negatively and are considerably very hard to measure and manage, especially at an early project development phase. Hence, software cost estimation involves the overall assessment of these parameters, even though for the majority of the projects, the most dominant and popular metric is the effort cost, typically measured in person-months.

Recent attempts have investigated the potential of employing Artificial Intelligence-oriented methods to forecast software development effort, usually utilising publicly available

datasets (e.g., Dolado, 2001; Idri et al., 2002; Jun & Lee, 2001; Khoshgoftaar et al., 1998; Xu & Khoshgoftaar, 2004) that contain a wide variety of cost drivers. However, these cost drivers are often ambiguous because they present high variations in both their measure and values. As a result, cost assessments based on these drivers are somewhat unreliable. Therefore, by detecting those project cost attributes that decisively influence the course of software costs and similarly define their possible values may constitute the basis for yielding better cost estimates. Specifically, the complicated problem of software cost estimation may be reduced or decomposed into devising and evolving bounds of value ranges for the attributes involved in cost estimation using the theory of conditional sets (Packard, 1990). These ranges may then be used to attain adequate predictions in relation to the effort located in the actual project data. The motivation behind this work is the utilization of rich empirical data series of software project cost attributes (despite suffering from limited quality and homogeneity) to produce robust effort estimations. Previous work on the topic has suggested high sensitivity to the type of attributes used as inputs in a certain Neural Network model (MacDonell & Shepperd, 2003). These inputs are usually discrete values from well-known and publicly available datasets. The data series indicate high variations in the attributes or factors considered when estimating effort (Dolado, 2001). The hypothesis is that if we manage to reduce the sensitivity of the technique by considering indistinct values in terms of ranges, instead of crisp discrete values, and if we employ an evolutionary technique, like Genetic Algorithms, we may be able to address the effect of attribute variations and thus provide a near-to-optimum solution to the problem. Consequently, the technique proposed in this chapter may provide some insight regarding which cost drivers are the most important. In addition, it may lead to identifying the most favorable attribute value ranges for a given dataset that can yield a 'secure' and more flexible effort estimate, again having the same reasoning in terms of ranges. Once satisfactory and robust value ranges are detected and some confidence regarding the most influential attributes is achieved, then cost estimation accuracy may be improved and more reliable estimations may be produced.

The remainder of this work is structured as follows: Section 2 presents a brief overview of the related software cost estimation literature and mainly summarizes Artificial Intelligence techniques, such as Genetic Algorithms (GA) exploited in software cost estimation. Section 3 encompasses the description of the proposed methodology, along with the GA variance constituting the method suggested, a description of the data used and the detailed framework of our approach. Consequently, Section 4 describes the experimental procedure and the results obtained after training and validating the genetic evolution of value ranges for the problem of software cost estimation. Finally, Section 5 concludes the chapter with a discussion on the difficulties and trade-offs presented by the methodology in addition to suggestions for improvements in future research steps.

2. Related work

Traditional model-based approaches to cost estimation, such as COCOMO, Function Point Analysis (FPA) and SLIM, assume that if we use some independent variables (i.e., project characteristics) as inputs and a dependent variable as the output (namely development effort), the resulted complex I/O relationships may be captured by a formula (Pendharkar et al., 2005). In reality, this is never the case. In COCOMO (Boehm, 1981), one of the most popular models for software cost estimation, the development effort is calculated using the estimated delivered source instructions and an effort adjustment factor, applied to three

distinct levels (basic, intermediate and advanced) and two constant parameters. COCOMO was revised in newer editions (Boehm et al., 1995; Boehm et al., 2000), using software size as the primary factor and 17 secondary cost factors. The revised model is regression-based and involves a mixture of three cost models, each corresponding to a stage in the software life-cycle namely: Applications Composition, Early Design and Post Architecture. The Application Composition stage involves prototyping efforts; the Early Design stage includes only a small number of cost drivers as there is not enough information available at this point to support fine-grained cost estimation; the Post Architecture stage is typically applied after the software architecture has been defined and provides estimates for the entire development life-cycle using effort multipliers and exponential scale factors to adjust for project, platform, personnel, and product characteristics.

Models based on Function Points Analysis (FPA) (Albrecht & Gaffney, 1983) mainly involve identifying and classifying the major system components such as external inputs, external outputs, logical internal files, external interface files and external inquiries. The classification is based on their characterization as 'simple', 'average' or 'complex', depending on the number of interacting data elements and other factors. Then, the unadjusted function points are calculated using a weighting schema and adjusting the estimations utilizing a complexity adjustment factor. This is influenced by several project characteristics, namely data communications, distributed processing, performance objective, configuration load, transaction rate, on-line data entry, end-user efficiency, on-line update, complex processing, reusability, installation ease, operational ease, multiple sites and change facilitation.

In SLIM (Fairley, 1992) two equations are used: the software productivity level and the manpower equation, utilising the Rayleigh distribution (Putnam & Myers, 1992) to estimate project effort schedule and defect rate. The model uses a stepwise approach and in order to be applicable the necessary parameters must be known upfront, such as the system size - measured in KDSI (thousand delivered source instructions), the manpower acceleration and the technology factor, for which different values are represented by varying factors such as hardware constraints, personnel experience and programming experience. Despite being the forerunner of many research activities, the traditional models mentioned above, did not produce the best possible results. Even though many existing software cost estimation models rely on the suggestion that predictions of a dependent variable can be formulated if several (in)dependent project characteristics are known, they are neither a silver bullet nor the best-suited approaches for software cost estimation (Shukla, 2000).

Over the last years, computational intelligence methods have been used attaining promising results in software cost estimation, including Neural Networks (NN) (Jun & Lee, 2001; Papatheocharous & Andreou, 2007; Tadayon, 2005), Fuzzy Logic (Idri et al., 2002; Xu & Khoshgoftaar , 2004), Case Based Reasoning (CBR) (Finnie et al., 1997; Shepperd et al., 1996), Rule Induction (RI) (Mair et al., 2000) and Evolutionary Algorithms.

A variety of methods, usually evolved into hybrid models, have been used mainly to predict software development effort and analyze various aspects of the problem. Genetic Programming (GP) is reported in literature to provide promising approximations to the problem. In (Burgess & Leftley, 2001) a comparative evaluation of several techniques is performed to test the hypothesis of whether GP can improve software effort estimates. In terms of accuracy, GP was found more accurate than other techniques, but does not converge to a good solution as consistently as NN. This suggests that more work is needed towards defining which measures, or combination of measures, is more appropriate for the

particular problem. In (Dolado, 2001) GP evolving tree structures, which represent software cost estimation equations, is investigated in relation to other classical equations, like the linear, power, quadratic, etc. Different datasets were used in that study yielding diverse results, classified as 'acceptable', 'moderately good', 'moderate' and 'bad' results. Due to the reason that the datasets examined varied extremely in terms of complexity, size, homogeneity, or values' granularity consistent results were hard to obtain. In (Lefley, & Shepperd 2003) the use of GP and other techniques was attempted to model and estimate software project effort. The problem was modeled as a symbolic regression problem to offer a solution to the problem of software cost estimation and improve effort predictions. The so-called "Finnish data set" collected by the software project management consultancy organization SSTF was used in the context of within and beyond a specific company and obtained estimations that indicated that with the approaches of Least-Square Regression, NN and GP better predictions could be obtained. The results from the top five percent estimators yielded satisfactory performance in terms of Mean Relative Error (MRE) with the GP appearing to be a stronger estimator achieving better predictions, closer to the actual values more often than the rest of the techniques. In the work of (Huang & Chiu, 2006) a GA was adopted to determine the appropriate weighted similarity measures of effort drivers in analogy-based software effort estimation models. These models identify and compare the software project developed with similar historical projects and produce an effort estimate. The ISBSG and the IBM DP services databases were used in the experiments and the results obtained showed that among the applied methods, the GA produced better estimates and the method could provide objective weights for software effort drivers rather than the subjective weights assigned by experts.

In summary, software cost estimation is a complicated activity since there are numerous cost drivers, displaying more than a few value discrepancies between them, and highly affecting development cost assessment. Software development metrics for a project reflect both qualitative measures, such as, team experiences and skills, development environment, group dynamics, culture, and quantitative measures, for example, project size, product characteristics and available resources. However, for every project characteristic the data is vague, dissimilar and ambiguous, while at the same time formal guidelines on how to determine the actual effort required to complete a project based on specific characteristics or attributes do not exist. Previous attempts to identify possible methods to accurately estimate development effort were not as successful as desired, mainly because calculations were based on certain project attributes of publicly available datasets (Jun & Lee, 2001). Nevertheless, the proportion of evaluation methods employing historical data is around 55% from a total of 304 research papers investigated by Jorgensen & Shepperd in 2004 (Jorgensen & Shepperd, 2007). According to the same study, evaluation of estimation methods requires that the datasets be as representative as possible to the current or future projects under evaluation. Thus, if we wish to evaluate a set of projects, we might consider going a step back, and re-define a more useful dataset in terms of conditional value ranges. These ranges may thus lead to identifying representative bounds for the available values of cost drivers that constitute the basis for estimating average cost values.

3. The proposed cost estimation framework

The framework proposed in this chapter encompasses the application of the theory of conditional sets in combination with Genetic Algorithms (GAs). The idea is inspired by the

work presented by Packard et al. (Meyer & Packard, 1992; Packard, 1990) utilising GAs to evolve conditional sets. The term conditional set refers to a set of boundary conditions. The main concept is to evaluate the evolved value ranges (or conditional sets) and extract underlying determinant relationships among attributes and effort in a given dataseries. This entails exploring a vast space of solutions, expressed in ranges, utilising additional manufactured data than those located into a well-known database regularly exploited for software effort estimation.

What we actually propose is a method for investigating the prospect of identifying the exact value ranges for the attributes of software projects and determining the factors that may influence development effort. The approach proposed implies that the attributes' value ranges and corresponding effort value ranges are automatically generated, evaluated and evolved through selection and survival of the fittest in a way similar to natural evolution (Koza, 1992). The goal is to provide complementing weights (representing the notion of ranked importance to the associated attributes) together with effort predictions, which could possibly result in a solution more efficient and practical than the ones created by other models and software cost estimation approaches.

3.1 Conditional sets theory and software cost

In this section we present some definitions and notations of conditional sets theory in relation to software cost based on paradigms described in (Adamopoulos et al., 1998; Packard, 1990).

Consider a set of n cost attributes $\{A_1, A_2, \ldots, A_n\}$, where each A_i has a corresponding discrete value x_i. A software project may be described by a vector of the form:

$$L = \{x_1, x_2, ..., x_n\} \tag{1}$$

Let us consider a condition C_i of the form:

$$C_i : (lb_i < x_i < ub_i),\ i = 1...n \tag{2a}$$

where lb_i and ub_i are the lower and upper bounds of C_i respectively for which:

$$\forall C_i : \left| lb_i - ub_i \right| < \varepsilon \tag{2b}$$

that is, lb_i and ub_i have minimal difference in their value, under a specific threshold ε.

Consider also a conditional set S; we say that S is of length l ($\leq n$) if it entails l conditions of the form described by equations (2a) and (2b), which are coupled via the logical operators of AND and OR as follows:

$$S_{AND} = C_1 \wedge C_2 \wedge ... \wedge C_l \tag{3}$$

$$S_{OR} = C_1 \vee C_2 \vee ... \vee C_l \tag{4}$$

We consider each conditional set S as an individual in the population of our GA, which will be thoroughly explained in the next section as part of the proposed methodology. We use equations (3) and (4) to describe conditional sets representing cost attributes, or to be more precise, cost metrics. What we are interested in is the definition of a set of software projects,

M, the elements of which are vectors as in equation (1) that hold the values of the specific cost attributes used in relation with a conditional set. More specifically, the set *M* can be defined as follows:

$$M = \{L_1, L_2, ..., L_m\} \tag{5}$$

$$L_i = \{x_{i,1}, x_{i,2}, ..., x_{i,l}\}, \ i = 1...m \tag{6}$$

where *l* denotes the number of cost attributes of interest.
A conditional set *S* is related to *M* according to the conditions in equations (3) or (4) that are satisfied as follows:

$$\forall L_i : \quad x_{i,k} \ satisfies \ C_k \ , \ i = 1...m, \ \ k = 1...l \ \ (\text{AND}) \tag{7}$$

$$\begin{aligned} & x_{i,1} \ satisfies \ C_1 \ OR \ x_{i,2} \ satisfies \ C_2, ... \\ & ..., \ OR \ x_{i,l} \ satisfies \ C_l, \ \ i = 1...m, \ \ (\text{OR}) \end{aligned} \tag{8}$$

3.2 Methodology

Before proceeding to describe the methodology proposed we provide a short description of the dataset used. The dataset was obtained from the International Software Benchmarking Standards Group (ISBSG, Repository Data Release 9 - ISBSG/R9, 2005) and contains an analysis of software project costs for a group of projects. The projects come from a broad cross section of industry and range in size, effort, platform, language and development technique data. The release of the dataset used contains 92 variables for each of the projects and hosts multi-organizational, multi-application domain and multi-environment data that may be considered fairly heterogeneous (International Software Benchmarking Standards Group, http://www.isbsg.org/). The dataset was recorded following data collection standards ensuring broad acceptance. Nevertheless, it contains more than 4,000 data from more than 20 countries and hence it is considered highly heterogeneous. Therefore, data acquisition, investigation and employment of the factors that impact planning, management and benchmarking of software development projects should be performed very cautiously.
The proposed methodology is divided into three steps, namely the data pre-processing step, the application of the GA and the evaluation of the results. Figure 1 summarizes the methodology proposed and the steps followed for evolving conditional sets and providing effort range predictions. Several filtered sub-sets of the ISBSG/R9 dataset were utilized for the evolution of conditional sets, initially setting up the required conditional sets. The conditional sets are coupled with two logical operators (AND and OR) and the investigation lies with extracting the ranges of project features or characteristics that describe the associated project effort. Furthermore, the algorithm creates a random set or initial population of conditions (individuals). The individuals are then evolved through specific genetic operators and evaluated internally using the fitness functions. The evolution of individuals continues while the termination criteria are not satisfied, among these a maximum number of iterations (called generations or epochs) or no improvement in the maximum fitness value occurs for a specific number of generations. The top 5% individuals resulting in the higher fitness evaluations are accumulated into the optimum range

population, which then are advanced to the next algorithm generation (repetition). At the end, the final population produced that satisfies the criteria is used to estimate the mean effort, whereas at the evaluation step, the methodology is assessed through various performance metrics. The most successful conditional sets evolved by the GA that have small assembled effort ranges with relatively small deviation from the mean effort, may then be used to predict effort of new, unknown projects.

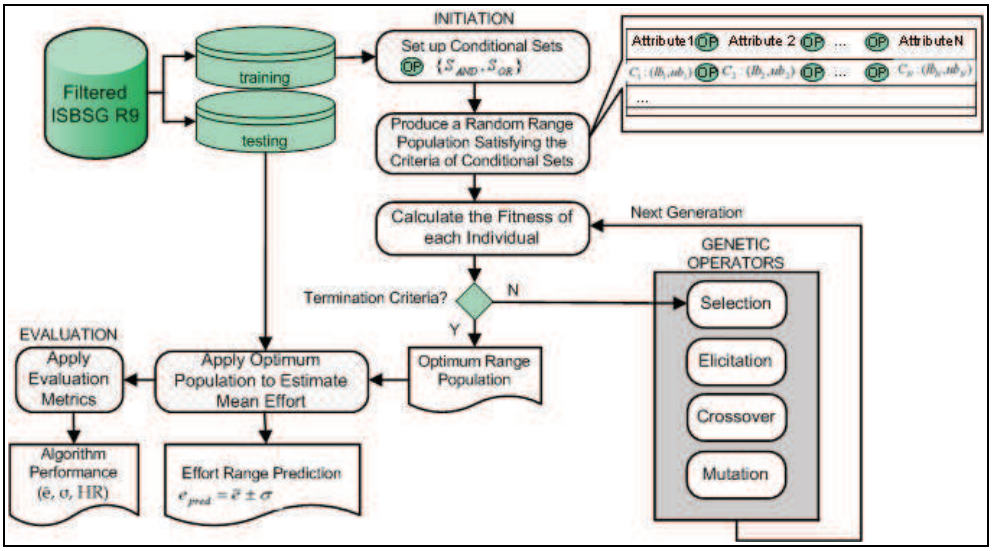


Fig. 1. Methodology followed for evolving conditional sets

3.2.1 Data pre-processing

In this step the most valuable set of attributes, in terms of contribution to effort estimation, are assembled from the original ISBSG/R9 dataset. After careful consideration of guidelines provided by the ISBSG and other research organizations, we decided to the formation of a reduced ISBSG dataset including the following main attributes: the project id (ID), the adjusted function points of the product (AFP), the project's elapsed time (PET), the project's inactive time (PIT), the project's delivery rate (productivity) in functional size units (PDRU), the average team size working on the project (ATS), the development type (DT), the application type (AT), the development platform (DP), the language type (LT), the primary programming language (PPL) and the resource level (RL) and the work effort expensed during the full development life-cycle (EFF) which will be used as a sort of output by the corresponding evolutionary algorithm. The attributes selected from the original, wider pool of ISBSG, were further filtered to remove those attributes with categorical-type data and other attributes that could not be included in the experimentation. Also, some attributes underwent value transformations, for example instead of PET and PIT we used their subtraction, normalized values for AFP and specific percentiles defining acceptance thresholds for filtering the data.

The first experiments following our approach indicated that further processing of the attributes should be performed, as the approach was quite strict and not applicable for heterogeneous datasets containing many project attributes with high deviations in their

values and measurement. Therefore, this led us to examine smaller, more compact, homogeneous and free from outlier subsets. In fact, we managed to extract three final datasets which we used in our final series of experiments. The first dataset (DS-1) contained the main attributes suggested by Function Point Analysis (FPA) to provide measurement of project software size, and included: Adjusted Function Points (AFP), Enquiry Count (EC), File Count (FC), Added Count (AC) and Changed Count (CC). These attributes were selected based on previous findings that considered them to be more successful in describing development effort after applying sensitivity analysis on the inputs with Neural Networks (Papatheocharous & Andreou, 2007). The second dataset (DS-2) is a variation of the previous dataset based on the preliminary results of DS-1, after performing normalization and removing the outliers according to the lower and upper thresholds defined by the effort box-plots. This resulted to the selection of the attributes: Normalized PDR-AFP (NAFP), Enquiry Count (EC), File Count (FC) and Added Count (AC). Finally, the third dataset (DS-3) created included the project attributes that can be measured early in the software life-cycle consisting of: Adjusted Function Points (AFP), Project's Delivery Rate (PDRU), Project's Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS) attributes in which also box-plots and percentile thresholds were used to remove outliers.

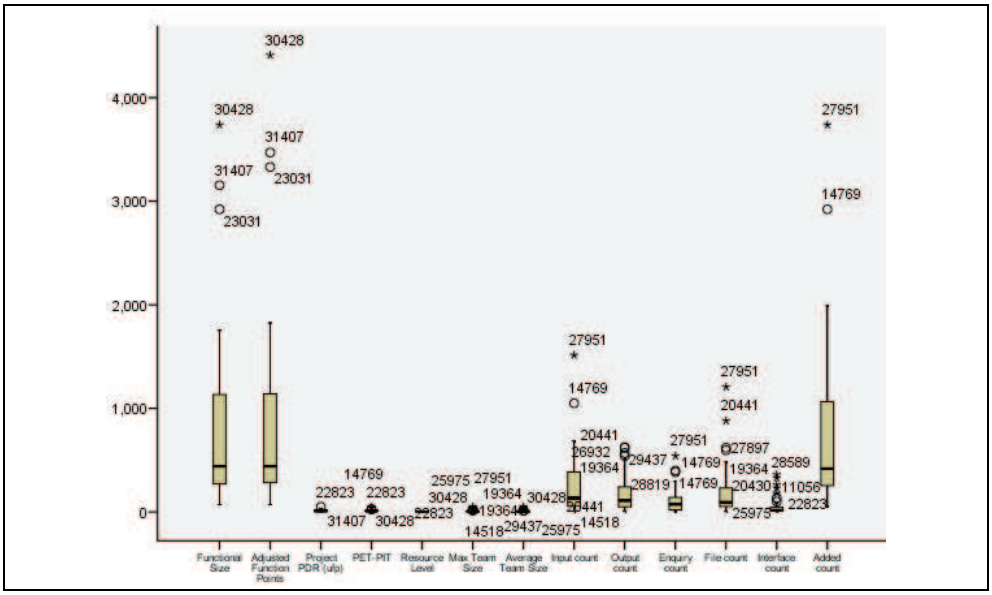


Fig. 2. Example of box-plots for the ISBSG project attributes (original full dataset)

It is noteworthy that each dataset also contained the values of the development work effort (EFF), the output attribute that we wanted to predict. As we already mentioned, the last data pre-processing step of the three datasets constructed included the cleaning of null and outlying values. The theory of box-plots was used to locate the outlying figures from the datasets and project cleaning was performed for each project variable separately. Figure 2 above shows an example of the box-plots created for each variable on the original full dataset. We decided to disregard the extreme outliers (marked as asterisks) occurring in each of the selected attributes and also exclude those projects considered as mild outliers (marked as circles), thus imposing more strict filtering associated with the output variable effort (EFF).

3.2.2 Genetic algorithm application

Genetic Algorithms (GAs) are evolutionary computational approaches that are domain-independent, and aim to find approximated solutions in complex optimization and search problems (Holland, 1992). They achieve this by pruning a population of individuals based on the Darwinian principle of reproduction and 'survival of the fittest' (Koza, 1992). The fitness of each individual is based on the quality of the simulated individual in the environment of the problem investigated. The process is characterized by the fact that the solution is achieved by means of a cycle of generations of candidate solutions that are pruned by using a set of biologically inspired operators. According to evolutionary theories, only the most suited solutions in a population are likely to survive and generate offspring, and transmit their biological heredity to the new generations. Thus, GAs are much superior to conventional search and optimization techniques in high-dimensional problem spaces due to their inherent parallelism and directed stochastic search implemented by recombination operators. The basic process of our GA operates through a simple cycle of three stages, as these were initially described by (Michalewicz, 1994):

Stage 1: Randomly create an initial population of individuals *P*, which represent solutions to the given problem (in our case, ranges of values in the form of equations (3) or (4)).

Stage 2: Perform the following steps for each generation:

2.1. Evaluate the fitness of each individual in the population using equations (9) or (10) below, and isolate the best individual(s) of all preceding populations.

2.2. Create a new population by applying the following genetic operators:

2.2.1. *Selection*; based on the fitness select a subset of the current population for reproduction by applying the roulette wheel method. This method of reproduction allocates offspring values using a roulette wheel with slots sized according to the fitness of the evaluated individuals. It is a way of selecting members from a population of individuals in a natural way, proportional to the probability set by the fitness of the parents. The higher the fitness of the individual is, the greater the chance it will be selected, however it is not guaranteed that the fittest member goes to the next generation. So, additionally, elitism is applied, where the top best performing individuals are copied in the next generation and thus, rapidly increase the performance of the algorithm.

2.2.2. *Crossover*; two or more individuals are randomly chosen from the population and parts of their genetic information are recombined to produce new individuals. Crossover with two individuals takes place either by exchanging their ranges at the crossover point (inter-crossover) or by swapping the upper or lower bound of a specific range (intra-crossover). The crossover takes place on one (or more) randomly chosen crossover point(s) along the structures of the two individuals.

2.2.3. *Mutation*; randomly selected individuals are altered randomly and inserted into the new population. The alteration takes place at the upper or lower bound of a randomly selected range by adding or subtracting a small random number. Mutation intends to preserve the diversity of the population by expanding the search space into regions that may contain better solutions.

2.3. Replace the current population with the newly formed population.

Stage 3: Repeat from stage 2 unless a termination condition is satisfied. Output the individual with the best fitness as the near to optimum solution.

Each loop of the steps is called a generation. The entire set of iterations from population initialization to termination is called a run. At the termination of the process the algorithm promotes the "best-of-run" individual.

3.2.3 Evaluation

The individuals evolved by the GA are evaluated according to the newly devised fitness functions of AND or OR, specified as:

$$F_{AND} = k + \frac{1}{\sigma} + \frac{1}{\left(\sum_{i=1}^{I}(ub_i - lb_i) * w_i\right)} \tag{9}$$

$$F_{OR} = \sum_{i=1}^{I}\left(k_i + \frac{1}{\sigma_i} + \frac{1}{ub_i - lb_i}\right) * w_i \tag{10}$$

where k represents the number of projects satisfying the conditional set, k_i the number of projects satisfying only condition C_i, and σ, σ_i are the standard deviations of the effort of the k and k_i projects, respectively.

By using the standard deviation in the fitness evaluation we promote the evolved individuals that have their effort values close to the mean effort value of either the k projects satisfying S (AND case) or either the k_i projects satisfying C_i (OR case). Additionally, the evaluation rewards individuals whose difference among the lower and upper range is minimal. Finally, w_i in equations (9) and (10) is a weighting factor corresponding to the significance given by the estimator to a certain cost attribute.

The purpose of the fitness functions is to define the appropriateness of the value ranges produced within each individual according to the ISBSG dataset. More specifically, when an individual is evaluated the dataset is used to define how many records of data (a record corresponds to a project with specific values for its cost attributes and effort) lay within the ranges of values of the individual according to the conditions used and the logical operator connecting these conditions. It should be noted at this point that in the OR case the conditional set is satisfied if at least one of its conditions is satisfied, while in the AND case all conditions in S must be satisfied. Hence, k (and σ) is unique for all ranges in the AND case, while in the OR case k may have a different value for each range i. That is why the fitness functions of the two logical operators are different. The total fitness of the population in each generation is calculated as the sum of the fitness values of the individuals in P.

Once the GA terminates the best individual is used to perform effort estimation. More specifically, in the AND case we distinguish the projects that satisfy the conditional set used to train the GA, while in the OR case the projects that satisfy one or more conditions of the set. Next we find the mean effort value ($\bar{e}$) and standard deviation (σ) of those projects. If we have a new project for which we want to estimate the corresponding development effort, we first check whether the values of its attributes lay within the ranges of the best individual and that it satisfies the form of the conditional set (AND or OR). If this holds, then the effort of the new project is estimated to be:

$$e_{pred} = \bar{e} \pm \sigma \tag{11}$$

where e_{pred} is the mean value of the effort of the projects satisfying the conditional set S.

4. Experimental process

This section explains in detail the series of experiments conducted and also presents some preliminary results of the methodology. The methodology was tested on the three different datasets described in the previous section.

4.1 Design of the experiments

Each dataset was separated into two smaller sub-datasets, the first of which was used for training and the second for validation. This enables the assessment of the generalization and optimization ability of the algorithm, firstly under training conditions and secondly with new, unknown to the algorithm, data. At first, a series of initial setup experiments was performed to define and tune the parameters of the GA. These are summarized in Table 1. The values for the GA parameters were set after experimenting with different generation epochs, as well as mutation and crossover rates and various number of points of crossover. A number of control parameters were modified for experimenting and testing the sensitivity of the solution to their modification.

Category	Value	Details
Attributes set	{ S_{AND}, S_{OR} }	
Solution representation	L	
Generation size	1000 epochs	
Population size	100 individuals	
Selection		Roulette wheel based on fitness of each individual
Elitism		Best individuals are forwarded (5%)
Mutation	Ratio 0.01-0.05	Random mutation
Crossover	Ratio 0.25-0.5	Random crossover (inter-, intra-)
Termination criterion		Generations size is reached or no improvements are noted for more than 100 generations

Table 1. Genetic Algorithm main parameters

We then proceeded to produce a population of 100 individuals representing conditional sets S (or ranges of values coupled with OR or AND conditions), as opposed to the discrete values of the attributes found in the ISBSG dataset. These quantities, as shown in equations (2a) and (2b), were generated to cover a small range of values of the corresponding attributes, but are closely related to (or within) the actual values found in the original data series.

Throughout an iterative production of generations the individuals were evaluated using the fitness functions specified in equations (9) or (10) with respect to the approach adopted. As previously mentioned, this fitness was assessed based on the:

- Standard deviation

- Number of projects in *L* satisfying (7) and (8)
- Ranges produced for the attributes

Fitness is also affected by the weights given by the estimator to separate between more and less important attributes. From the fitness equations we may deduce that the combination of a high number of projects in *L*, a low standard deviation with respect to the mean effort and a small range for the cost attributes (at least the most significant) produces high fitness values. Thus, individuals satisfying these specific requirements are forwarded to the next population until the algorithm terminates. Figure 3 depicts the total fitness value of a sample population through generations, which, as expected, rises as the number of epochs increases. A plateau is observed in the range 50-400 epochs which may be attributed to a possible trapping of the GA to a local minimum. The algorithm seems to escape from this minimum with its total fitness value constantly being improved along the segment of 400-450 epochs and then stabilizing. Along the repetitions of the GA algorithm execution, the total population fitness improves showing that the methodology performs consistently well.

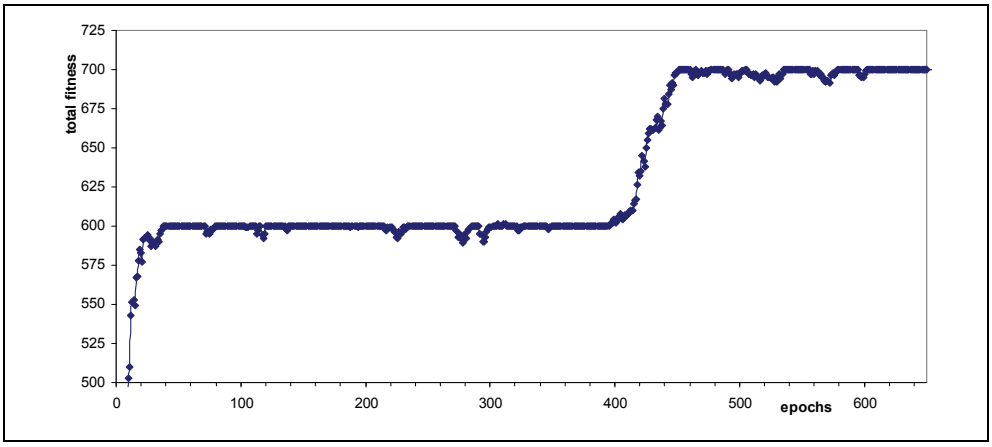


Fig. 3. Total Fitness Evolution

4.2 Experimental results

The experimental evaluation procedure was based on both the AND and OR approaches. We initially used the attributes of the datasets with equal weight values and then subsequently with combinations of different weight values. Next, as the weight values were modified it was clear that various assumptions about the importance of the given attributes for software effort could be drawn. In the first dataset for example, the Adjusted Function Point (AFP) attribute was found to have a minimal effect on development effort estimations and therefore we decided to re-run the experiments without this attribute taking part. The process was repeated for all attributes of the dataset by continuously updating the weight values and reducing the number of attributes participating in the experiments, until no more insignificant attributes remained in the dataset. The same process was followed for all the three datasets respectively, while the results summarized in this section represent only a few indicative results obtained throughout the total series of experiments.

Tables 2 and 3 present indicative best results obtained with the OR and AND approaches, respectively, that is, the best individual of each run for a given set of weights (significance)

that yield the best performance with the first dataset (DS-1). Table 4 presents the best results obtained with the AND and OR approach with the second dataset (DS-2) and Table 5 lists the best obtained results with the third attribute dataset (DS-3).

Attribute Weights / Ranges				Evaluation Metrics		
FC	**AC**	**CC**	**EC**	**ē**	**σ**	**HR**
0.1 [11, 240]	0.4 [1, 1391]	0.1 [206, 1739]	0.4 [14, 169]	3014.2	1835.1	81/179
0.3 [11, 242]	0.1 [1, 1363]	0.4 [60, 1498]	0.2 [1, 350]	3125.5	1871.5	81/184
0.3 [11, 242]	**0.4 [1, 1391]**	**0.1 [1616, 2025]**	**0.2 [14, 268]**	**3204.5**	**1879.2**	**81/187**
0.2 [19, 298]	0.4 [1, 1377]	0.1 [1590, 3245]	0.3 [14, 268]	3160.3	1880.7	81/178
0.2 [11, 240]	0.2 [1, 1377]	0.4 [46, 573]	0.2 [1, 350]	3075.1	1857.2	79/183
0.2 [3, 427]	**0.4 [1, 1377]**	**0.2 [46, 579]**	**0.2 [1, 347]**	**3254.5**	**1857**	**83/191**

Table 2. Indicative Results of conditional sets using the OR approach and DS-1

Evaluation metrics were used to assess the success of the experiments, based on (i) the total mean effort, (ii) the standard deviation and, (iii) the hit ratio. The hit ratio (given in equation (12)) provides a complementary piece of information about the results. It basically assesses the success of the best individual evolved by the GA on the testing set. Recall that the GA results in conditional set of value ranges which are used to compute the mean effort and standard deviation of the projects satisfying the conditional set. Next, the number of projects n in the testing set that satisfy the conditional set is calculated. Of those n projects we compute the number of projects b that have additionally a predicted effort value satisfying equation (11). The latter may be called the "hit-projects". Thus, equation (12) essentially calculates the ratio of hit-projects in the testing set:

$$hit\ ratio(HR) = \frac{b}{n} \tag{12}$$

The results are expressed in a form satisfying equations (3)-(8). A numerical example could be a set of range values produced to satisfy equations (2a) and (2b) coupled with the logical operator of AND as follows:

$$S_{AND} = [1700, 2000] \wedge [16, 205] \wedge \ldots \wedge [180\ 200] \tag{13}$$

The projects that satisfy equation (7) are then accumulated in set L (numbers represent project IDs):

$$L=\{1827, 1986, 1987,\ldots,1806\} \tag{14}$$

Using L the $\bar{e}$, σ and HR figures may be calculated. The success of the experiments is a combination of the aforementioned metrics. Finally, we characterize an experiment as successful if its calculated standard deviation is adequately lower than the associated mean effort and achieves a hit ratio above 60%.

Indicative results of the OR conditional sets are provided in Table 2. We observe that the OR approach may be used mostly for comparative analysis of the cost attributes by evaluating their significance in the estimation process, rather the estimation itself, as results indicate low performance. Even though the acceptance level of the hit ratio is better than average, the high value of the standard deviation compared to the mean effort (measured in person days) indicates that the results attained are dispersed and not of high practical value. The total mean effort of the best 100 experiments was found equal to 2929 and the total standard deviation equal to 518. From these measures the total standard error was estimated at 4.93, which is not satisfactory, but at the same time it cannot be considered bad. However, in terms of suggesting ranges of values for specific cost attributes on which one may base an estimation, the results do not converge to a clear picture. It appears that when evaluating different groups of data in the dataset we attain large dissimilarities, suggesting that clustered groups of data may be present in the series. Nevertheless, various assumptions can be drawn from the methodology as regards to which of the attributes seem more significant and to what extent. The selected attributes, namely Added Count (AC), File Count (FC), Changed Count (CC) and Enquiry Count (EC) seem to have a descriptive role over effort as they provide results that may be considered promising for estimating effort. Additionally, the best results of Table 2 (in bold) indicate that the leading factor is Added Count (AC), with its significance being ranked very close to that of the File Count (FC).

Attribute Weights / Ranges			**Evaluation Metrics**		
FC	**AC**	**CC**	**ē**	**σ**	**HR**
0.1 [22, 223]	0.2 [187, 504]	0.7 [9, 195]	3503	1963.6	3/4
0.5 [22, 223]	0.3 [114, 420]	0.2 [9, 197]	3329.4	2014.2	3/4
0.2 [14, 156]	**0.4 [181, 489]**	**0.4 [9, 197]**	**3778.8**	**2061.4**	**3/4**
0.4 [22, 223]	**0.4 [167, 390]**	**0.2 [9, 195]**	**3850.3**	**2014.3**	**3/4**
0.2 [14, 154]	0.8 [35, 140]	0 0	2331.2	1859.4	12/16
0.7 [14, 152]	0.3 [35, 141]	0 0	2331.2	1859.4	12/16

Table 3. Indicative Results of conditional sets using the AND approach with DS-1

On the other hand, the AND approach (Table 3) provides more solid results since it is based on a more strict method (i.e. satisfy all ranges simultaneously). The results indicate again some ranking of importance for the selected attributes. To be specific, Added Count (AC)

and File Count (FC) are again the dominant cost attributes, a finding which is consistent with the OR approach. We should also note that the attribute Enquiry Count (EC) proved rather insignificant in this approach, thus it was omitted from Table 3. Also, the fact that the results produced converge in terms of producing similar range bounds shows that the methodology may provide empirical indications regarding possible real attribute ranges. A high hit ratio of 75% was achieved for nearly all experiments in the AND case for the specified dataset, nevertheless this improvement is obtained with fewer projects, as expected, satisfying the strict conditional set compared to the more loose OR case. This led us to conclude that the specific attributes can provide possible ranges solving the problem and providing relatively consistent results.

The second dataset (DS-2) used for experimentation included the Normalized AFP (NAFP) and some of the previously investigated attributes for comparison purposes. The dataset was again tested using both the AND and OR approaches. The first four rows of the results shown in Table 4 individuals were obtained with the AND approach and the last two results with the OR approach. The figures listed in Table 4 show that the method 'approves' more individuals (satisfying the equations) because the ranges obtained are wider. Conclusively, the values used for effort estimation result to increase of the total standard error. The best individuals (in bold) were obtained after applying box-plots in relation to the first result shown, while the rest two results did not use this type of filtering. It is clear from the lowering of the value of the standard deviation that after box-plot filtering on the attributes some improvement was indeed achieved. Nevertheless, the HR stays quite low, thus we cannot argue that the ranges of values produced are robust to provide new effort estimates.

Attribute Weights / Ranges				**Evaluation Metrics**		
NAFP	**AC**	**FC**	**EC**	**ē**	**σ**	**HR**
0.25 [2, 134]	0.25 [215, 1071]	0.25 [513, 3678]	0.25 [4, 846]	11386.7	9005.4	9/58
0.25 **[7, 80]**	**0.25** **[34, 830]**	**0.25** **[88, 1028]**	**0.25** **[37, 581]**	**2861.7**	**2515.9**	**7/66**
0.25 **[1,152]**	**0.25** **[22, 859]**	**0.25** **[58, 3192]**	**0.25** **[20, 563]**	**3188.6**	**2470.9**	**3/221**
0.25 **[1, 156]**	**0.25** **[34, 443]**	**0.25** **[122, 2084]**	**0.25** **[37, 469]**	**3151.6**	**2377.9**	**4/139**
0.25 [1, 36]	0.25 [449, 837]	0.25 [23, 1014]	0.25 [7, 209]	4988.5	8521.2	10/458
0.25 [1, 159]	0.25 [169, 983]	0.25 [78, 928]	0.25 [189, 567]	4988.5	8521.2	10/458

Table 4. Indicative Results of conditional sets using the AND and OR approaches with DS-2

The purpose of the final dataset (DS-3) used in the experiments is to test whether a selected subset of attributes that can be measured early in the development life-cycle can provide adequately good predictions. Results showed that the attributes of Adjusted Function Points

(AFP), Project Delivery Rate (PDRU), Project Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS) may provide improvements for selecting ranges with more accurate effort estimation abilities. For these experiments only the AND approach is presented as the results obtained were regarded to be more substantial. In the experiments conducted with this dataset (DS-3) we tried to impose even stricter ranges, after the box-plots and outlier's removal in the initial dataset, by applying an additional threshold to retain the values falling within the 90% (percentile). This was performed for the first result listed in Table 5, whereas the threshold within the 70% percentile was also applied for the second result listed on the same table. We noticed that this led to a significant optimization of the results. Even though very few individuals are approved, satisfying the equations, the HR is almost always equal to 100%. The obtained ranges are more clearly specified and in addition, sound predictions can be made regarding effort since the best obtained standard deviation of effort falls to 74.9 which also constitutes one of the best predictions yielded by the methodology. This leads us to conclude that when careful removal of outliers is performed the proposed methodology may be regarded as achieving consistently successful predictions, yielding optimum ranges that are adequately small and suggesting effort estimations that lay within reasonable mean values and perfectly acceptable deviation from the mean.

Attribute Weights / Ranges					**Evaluation Metrics**		
AFP	**PDRU**	**PET**	**RL**	**ATS**	**ē**	**σ**	**HR**
0.2 [48, 1207]	0.2 [1, 7]	0.2 [3, 12]	0.2 [1, 3]	0.2 [2, 5]	2657.6	913.7	3/3
0.2 **[57, 958]**	**0.2** **[2, 13]**	**0.2** **[5, 10]**	**0.2** **[2, 4]**	**0.2** **[1, 6]**	**2131.0**	**74.9**	**2/2**
0.2 [133, 1409]	0.2 [1, 25]	0.2 [7, 21]	0.2 [2, 4]	0.2 [2, 10]	2986.6	1220.0	5/5
0.2 **[173, 1131]**	**0.2** **[1, 20]**	**0.2** **[2, 20]**	**0.2** **[2, 4]**	**0.2** **[1, 7]**	**2380.0**	**434.5**	**2/3**
0.25 **[189, 1301]**	**0.25** **[2, 26]**	**0** **0**	**0.25** **[1, 3]**	**0.25** **[2, 11]**	**2477.8**	**838.2**	**5/5**
0.25 [693, 1166]	0.25 [2, 23]	0 0	0.25 [1, 3]	0.25 [1, 4]	2478.0	565.6	2/2

Table 5. Indicative Results of conditional sets using the AND approach with DS-3

5. Conclusions

In this approach we aimed at addressing the problem of large variances found in available historical data that are used in software cost estimation. Project data is expensive to collect, manage and maintain. Therefore, if we wish to lower the dependence of the estimation to

the need of gathering accurate and homogenous data, we might consider simulating or generating data ranges instead of real crisp values.

The theory of conditional sets was applied in the present work with Genetic Algorithms (GAs) on empirical software cost estimation data. GAs are ideal for providing efficient and effective solutions in complex problems; there are, however, several trade-offs. One of the major difficulties in adopting such an approach is that it requires a thorough calibration of the algorithm's parameters. We have tried to investigate the relationship between software attributes and effort, by evolving attribute value ranges and evaluating estimated efforts. The algorithm promotes the best individuals in the reproduced generations through a probabilistic manner. Our methodology attempted to reduce the variations in performance of the model and achieve some stability in the results. To do so we approached the problem from the perspective of minimizing the differences in the ranges and the actual and estimated effort values to decisively determine which attributes are the most important in software cost estimates.

We used the ISBSG repository containing a relatively large quantity of data; nevertheless, this data suffers from heterogeneity thus presents low quality level from the perspective of level of values. We formed three different subsets selecting specific cost attributes from the ISBSG repository and filtering out outliers using box-plots on these attributes. Even though the results are of average performance when using the first two datasets, they indicated some importance ranking for the attributes investigated. According to this ranking, the attributes Added Count (AC) and File Count (FC) were found to lay among the most significant cost drivers for the ISBSG dataset. The third dataset included Adjusted Function Points (AFP), Project Delivery Rate (PDRU), Project Elapsed Time (PET), Resource Level (RL) and Average Team Size (ATS). These attributes may be measured early in the software life-cycle, thus this dataset may be regarded more significant than the previous two from a practical perspective. A careful and stricter filtering of this dataset provided prediction improvements, with the yielded results suggesting small value ranges and fair estimates for the mean effort of a new project and its deviation. There was also an indication that within different areas of the data, significantly different results may be produced. This is highly related to the scarcity of the dataset itself and supports the hypothesis that if we perform some sort of clustering in the dataset we may further minimize the deviation differences in the results and obtain better effort estimates.

Although the results of this work are at a preliminary stage it became evident that the approach is promising. Therefore, future research steps will concentrate on ways to improve performance, examples of which may be: (i) Pre-processing of the ISBSG dataset and appropriate clustering into groups of projects that will share similar value characteristics. (ii) Investigation of the possibility of reducing the attributes in the dataset by utilizing a significance ranking mechanism that will promote only the dominant cost drivers. (iii) Better tuning of the GA's parameters and modification/enhancement of the fitness functions to yield better convergence. (iv) Optimization of the trial and error weight factor assignment used in the present work by utilizing a GA. (v) Experimentation with other datasets containing selected attributes again proposed by a GA. Finally, we plan to perform a comparative evaluation of the proposed approach with other well established algorithms, like for example the COCOMO model.

6. References

Adamopoulos, A.V.; Likothanassis, S.D. & Georgopoulos, E.F. (1998). A Feature Extractor of Seismic Data Using Genetic Algorithms, *Signal Processing IX: Theories and Applications*, *Proceedings of EUSIPCO-98, the 9th European Signal Processing Conference*, Vol. 2, pp. 2429-2432, Typorama, Greece.

Albrecht, A.J. & Gaffney J.R. (1983). 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.

Boehm, B.W. (1981). *Software Engineering Economics*, Prentice Hall, New Jersey.

Boehm, B.W.; Clark, B.; Horowitz, E.; Westland, C.; Madachy, R.J. & Selby R.W. (1995). Cost Models for Future Software Life Cycle Processes: COCOMO 2.0, *Annals of Software Engineering*, Vol.1, pp. 57-94, Springer, Netherlands.

Boehm, B.W.; Abts, C. & Chulani, S. (2000). Software Development Cost Estimation Approaches – A Survey, *Annals of Software Engineering*, Vol.10, No. 1, pp. 177-205, Springer, Netherlands.

Burgess, C.J. & Lefley, M. (2001). Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation, *Information and Software Technology*, Vol. 43, No. 14, pp. 863-873, Elsevier, Amsterdam.

Dolado, J.J. (2000). A Validation of the Component-Based Method for Software Size Estimation, *IEEE Transactions on Software Engineering*, Vol. 26, No. 10, pp. 1006-1021, IEEE Computer Press, Washington D.C..

Dolado, J.J. (2001). On the Problem of the Software Cost Function, *Information and Software Technology*, Vol. 43, No. 1, pp. 61-72, Elsevier, Amsterdam.

Fairley, R.E. (1992). Recent Advances in Software Estimation Techniques, *Proceedings of the 14th International Conference on Software Engineering*, pp. 382-391, ACM, Melbourne, Australia.

Finnie, G.R.; Wittig, G.E. & Desharnais, J.-M. (1997). Estimating software development effort with case-based reasoning, *Proceedings of the 2nd International Conference on Case-Based Reasoning Research and Development ICCBR*, pp.13-22, Springer.

Holland, J.H. (1992). Genetic Algorithms, *Scientific American*, Vol. 267, No. 1, pp. 66–72, New York.

Huang, S. & Chiu, N. (2006). Optimization of analogy weights by genetic algorithm for software effort estimation, *Information and Software Technology*, Vol. 48, pp. 1034-1045, Elsevier.

Idri, A.; Khoshgoftaar, T.M. & Abran, A. (2002). Can Neural Networks be Easily Interpreted in Software Cost Estimation?, *Proceedings of the 2002 IEEE World Congress on Computational Intelligence*, pp. 1162-1167 IEEE Computer Press, Washington D.C..

International Software Benchmarking Standards Group (ISBSG), Estimating, Benchmarking & Research Suite Release 9, ISBSG, Victoria, 2005.

International Software Benchmarking Standards Group, http://www.isbsg.org/

Jorgensen, M. & Shepperd, M. (2007). A Systematic Review of Software Development Cost Estimation Studies, *IEEE Transactions on Software Engineering*, Vol. 33, No. 1, pp. 33-53, IEEE Computer Press, Washington D.C..

Jun, E.S. & Lee, J.K. (2001). Quasi-optimal Case-selective Neural Network Model for Software Effort Estimation, *Expert Systems with Applications*, Vol. 21, No. 1, pp. 1-14 Elsevier, New York.

Khoshgoftaar, T.M.; Evett, M.P.; Allen, E.B. & Chien, P. (1998). An Application of Genetic Programming to Software Quality Prediction *Computational Intelligence in Software Engineering*, *Series on Advances in Fuzzy Systems – Applications and Theory*, Vol. 16, pp. 176-195, World Scientific, Singapore.

Koza, J.R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Massachusetts.

Lederer, A.L. & Prasad, J. (1992). Nine Management Guidelines for Better Cost Estimating, *Communications of the ACM*, Vol. 35, No. 2, pp. 51-59, ACM, New York.

Lefley, M. & Shepperd, M.J. (2003). Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets, *Proceedings of GECCO*, pp. 2477-2487.

MacDonell, S.G. & Shepperd, M.J. (2003). Combining Techniques to Optimize Effort Predictions in Software Project Management, *Journal of Systems and Software*, Vol. 66, No. 2, pp. 91-98, Elsevier, Amsterdam.

Mair, C; Kadoda, G.; Lefley, M.; Phalp, K.; Schofield, C.; Shepperd, M. & Webster, S. (2000). An investigation of machine learning based prediction systems, *Journal of Systems Software*, Vol. 53, pp. 23–29, Elsevier.

Meyer, T.P. & Packard, N.H. (1992). Local Forecasting of High-dimensional Chaotic Dynamics, *Nonlinear Modeling and Forecasting*, Addison-Wesley.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, Berlin.

Packard, N.H. (1990). A Genetic Learning Algorithm for the Analysis of Complex Data, *Complex Systems*, Vol. 4, No. 5, pp. 543-572, Illinois.

Pendharkar, P.C.; Subramanian, G.H. & Rodger, J.A. (2005). A Probabilistic Model for Predicting Software Development Effort, *IEEE Transactions on Software Engineering*, IEEE Computer Press, Vol. 31, No. 7, pp. 615-624, Washington D.C..

Papatheocharous, E. & Andreou, A. (2007). Software Cost Estimation Using Artificial Neural Networks with Inputs Selection, *Proceedings of the 9th International Conference on Enterprise Information Systems*, pp. 398-407, Madeira, Portugal.

Putnam, L.H. & Myers, W. (1992). *Measures for Excellence, Reliable Software on Time, Within Budget*, Yourdan Press, New Jersey.

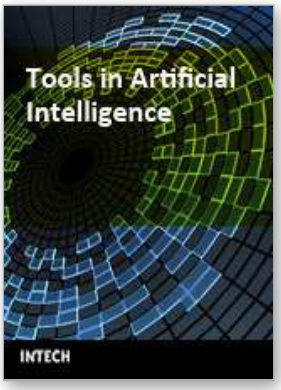
Shepperd, M. & Kadoda, G. (2001). Comparing Software Prediction Techniques Using Simulation, *IEEE Transactions on Software Engineering*, Vol. 27, No. 11, pp. 1014-1022, IEEE Computer Press, Washington D.C..

Shepperd, M.J.; Schofield, C. & Kitchenham, B. A. (1996). Effort estimation using analogy, *Proceedings of the 18th International Conference on Software Engineering*, pp. 170-178, Berlin.

Shukla, K.K. (2000). Neuro-genetic Prediction of Software Development Effort, *Information and Software Technology*, Vol. 42, No. 10, pp. 701-713, Elsevier, Amsterdam.

Tadayon, N. (2005). Neural Network Approach for Software Cost Estimation. *Proceedings of the International Conference on Information Technology: Coding and Computing*, pp. 815-818, IEEE Computer Press, Washington D.C..

Xu, Z. & Khoshgoftaar, T.M. (2004). Identification of Fuzzy Models of Software Cost Estimation, *Fuzzy Sets and Systems*, Vol. 145, No. 1, pp. 141-163, Elsevier, New York.



Tools in Artificial Intelligence

Edited by Paula Fritzsche

ISBN 978-953-7619-03-9
Hard cover, 488 pages
Publisher InTech
Published online 01, August, 2008
Published in print edition August, 2008

This book offers in 27 chapters a collection of all the technical aspects of specifying, developing, and evaluating the theoretical underpinnings and applied mechanisms of AI tools. Topics covered include neural networks, fuzzy controls, decision trees, rule-based systems, data mining, genetic algorithm and agent systems, among many others. The goal of this book is to show some potential applications and give a partial picture of the current state-of-the-art of AI. Also, it is useful to inspire some future research ideas by identifying potential research directions. It is dedicated to students, researchers and practitioners in this area or in related fields.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821