Chapter 2
Using Evolutionary Based Approaches to Estimate Software Development Effort

Filomena Ferrucci
University of Salerno, Italy

Carmine Gravino
University of Salerno, Italy

Rocco Oliveto
University of Salerno, Italy

Federica Sarro
University of Salerno, Italy

ABSTRACT

Software development effort estimation is a critical activity for the competitiveness of a software company; it is crucial for planning and monitoring project development and for delivering the product on time and within budget. In the last years, some attempts have been made to apply search-based approaches to estimate software development effort. In particular, some genetic algorithms have been defined and some empirical studies have been performed with the aim of assessing the effectiveness of the proposed approaches for estimating software development effort. The results reported in those studies seem to be promising. The objective of this chapter is to present a state of the art in the field by reporting on the most significant empirical studies undertaken so far. Furthermore, some suggestions for future research directions are also provided.

INTRODUCTION

Several factors characterise the costs of the software development (such as general costs, hardware, human resources, etc.). Nevertheless, it is widely recognised that the main factor is the "effort", meant as the amount of time spent to complete the project, expressed in terms of person-hours or man-months. So, the competitiveness of a software company heavily depends on the ability of its project managers to accurately predict in advance the effort required

DOI: 10.4018/978-1-61520-809-8.ch002

to develop software systems. Indeed, significant over or under-estimates can be very expensive for a company. Moreover, effort estimation is a critical basic activity for planning and monitoring software project development and for delivering the product on time and within budget.

Several methods have been proposed in order to estimate software development effort. Many of them determine the prediction exploiting some relevant factors of the software project, named cost drivers. These methods, named data-driven, exploit data from past projects, consisting of both factor values that are related to effort and the actual effort to develop the projects, in order to estimate the effort for a new project under development (Briand, Emam, Surmann, Wiekzorek, and Maxwell, 1999; Briand, Langley, and Wiekzorek, 2000; Shepperd and Schofield, 2000). In this class, we can find some widely used techniques, such as Linear and Stepwise Regression, Classification and Regression Tree, and Case-Based Reasoning (Briand and Wieczorek, 2002).

In the last years, some attempts have been made to apply search-based approaches to estimate software development effort. In particular, genetic algorithms (Goldberg, 1989) have been defined and assessed by some empirical studies (Burgess and Lefley, 2001; Chiu and Huang, 2007; Conte, Dunsmore, and Shen, 1986; Dolado, 2000; Lefley and Shepperd, 2003; Shan, Mckay, Lokan, and Essam, 2002; Shukla, 2000). The results reported in those studies seem to be promising.

Goal of the Chapter

The objective of this chapter is to report on the most significant empirical studies undertaken so far with the aim of assessing the effectiveness of search-based approaches for estimating software development effort. Furthermore, we provide some suggestions for future research directions.

Organization of the Chapter

The rest of the chapter is organised as follows. Section 2 introduces the problem of estimating development effort and briefly presents the widely employed estimation techniques as well as the validation methods and evaluation criteria used to assess an estimation technique. Section 3 provides a description of evolutionary based approaches for estimating development effort and reports on case studies performed to assess their effectiveness. Future research directions are instead described in Section 4.

BACKGROUND: ESTIMATING SOFTWARE DEVELOPMENT EFFORT

The prediction of software development effort plays a crucial role for the competitiveness of a software company and it is very important not only for the company that produces the software but also for its customers. Several benefits can be derived from an accurate estimate of software project development effort. Among them (Briand and Wieczorek, 2002):

- The possibility of defining the appropriate software costs, thus obtaining the contracts for the development of the software projects;
- The possibility of suitably planning/monitoring the project and allocate resources adequately, thus ensuring time to market and adequate software quality.

Software development effort can be influenced by several factors, among them the size of the software is the main factor. Other factors are the skill and the experience of the subjects involved in the projects, the complexity of the

software, the non functional requirements, the adopted software development process, etc. In the last decades, several approaches have been defined, which combine, in different ways, these factors by employing modelling techniques. A widely accepted taxonomy of estimation methods classified them in Non-Model Based and Model Methods (Briand and Wieczorek, 2002). While Non-Model Based Methods mainly take into account expert judgments (thus obtaining highly subjective evaluations), Model Based Methods involve the application of some algorithms to a number of factors to produce an effort estimation. These approaches use data from past projects, characterised by attributes that are related to effort (e.g. the size), and the actual effort to develop the projects, in order to construct a model that is used to estimate the effort for a new project under development.

Widely employed Model Based estimation methods are Linear Regression (LR), Case-Based Reasoning (CBR), and Classification And Regression Tree (CART) (Briand *et al.*, 1999; Briand *et al.*, 2000; Briand and Wieczorek, 2002; Shepperd and Schofield, 2000). Other novel approaches have been proposed in the literature. Any new approach must be validated by some empirical studies in order to verify its effectiveness, i.e., whether or not the predicted efforts are useful estimations of the actual development efforts. To this aim historical datasets are employed. In order to ensure strength to the validation process, it is recommended that data coming from the industrial world are employed. They can come from a single company or from several companies (cross-company datasets), such as the publicly available repository of the International Software Benchmarking Standards Group (ISBSG) that contains data from a great number of projects developed by companies around the world (ISBSG, 2009). A technique that is widely employed to validate an estimation approach is *cross-validation*. One round of cross-validation involves partitioning the dataset into two randomly complementary sets: the *training set* for model building and the *test set* (or *validation set*) for model evaluation (Briand and Wieczorek, 2002). To reduce variability, multiple rounds of cross-validation are performed using different partitions. The prediction accuracies are then averaged over the rounds. Several strategies have been proposed to obtain training and test sets. The *k-fold cross validation* suggests to partition the initial dataset of N observations in k randomly test sets of equal size, and then for each test set we have to consider the remaining observations as training set in order to build the estimation model. The *leave-one-out cross-validation* is widely used in the literature when dealing with small datasets, e.g. (Briand *et al.*, 1999). To apply the cross-validation, the original dataset of N observations is divided into N different subsets of training and validation sets, where each validation set has one project. Then, N steps are performed to get the predictions for the N validation sets.

Another technique that is often exploited is the *hold-out validation*, where a subset of observations is chosen randomly from the initial dataset to form the training set, and the remaining observations from the test set. Usually, about a third of the initial dataset is used as validation set.

To assess the acceptability of the derived estimations some evaluation criteria are proposed in the literature. Among them several summary measures, like *MMRE*, *MdMRE*, and *Pred(25)* (Conte, Dunsmore, and Shen, 1986), are widely employed and considered *de facto* standard evaluation criteria. They are based on the evaluation of the residuals, i.e., the difference between the actual and estimated efforts. In the following, we will report the definitions of these summary measures taking into account a validation set of n elements.

In order to take into account the error with respect to the actual effort, the *Magnitude of Relative Error* (Conte, Dunsmore, and Shen, 1986) is defined as:

$$MRE = \frac{\left|EF_{real} - EF_{pred}\right|}{EF_{real}}$$

where EF_{real} and EF_{pred} are the actual and the predicted efforts, respectively. *MRE* has to be calculated for each observation in the validation dataset. All the MRE values are aggregated across all the observations using the mean and the median, giving rise to the *Mean of MRE* (*MMRE*), and the *Median MRE* (*MdMRE*), where the latter is less sensitive to extreme values.

The *Prediction at level l* (Conte, Dunsmore, and Shen, 1986) is defined as:

$$Pred(l) = \frac{k}{n}$$

where *k* is the number of observations whose *MRE* is less than or equal to *l*, and *n* is the total number of observations in the validation set. Generally, a value of 25 for the level *l* is chosen. In other words, *Pred*(25) is a quantification of the predictions whose error is less than 25%. According to Conte *et al.* (1986), a good effort prediction model should have a $MMRE \leq 0.25$ and $Pred(25) \geq 0.75$, meaning that at least 75% of the predicted values should fall within 25% of their actual values. Other summary measures sporadically used are the *Balanced MMRE* (*BMMRE*), the *Mean Squared Error* (*MSE*) (Conte, Dunsmore, and Shen, 1986) and the *Adjusted Mean Square Error* (*AMSE*) (Burgess and Lefley, 2001). They are defined as follows:

$$BMMRE = \sum_{i=1}^{n} \left(\frac{\left|EF_{real_i} - EF_{pred_i}\right|}{min\left(EF_{real_i}, EF_{pred_i}\right)} \right) \frac{100}{n}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (EF_{real} - EF_{pred})^2$$

$$AMSE = \sum_{i=1}^{n} \frac{\left|EF_{real_i} - EF_{pred_i}\right|^2}{EF_{real_i} EF_{pred_i}}$$

where EF_{real_i} and EF_{pred_i} are the actual and the estimated efforts of the i^{th} observation of the validation set and *n* is the number of observations in the validation set.

Finally, in order to have an insight on the usefulness of a new method, its estimation accuracy is compared with the ones of other techniques. Several different benchmark methods are exploited to carry out such a comparison taking into account the above evaluation criteria. It is worth to noting that in the last years it has been widely recognized that the summary measures should be complemented by the analysis of boxplot of residuals and the comparisons among estimation techniques should be carried out by testing also the statistical significance of the absolute residuals. Such tests should be used to verify the following null hypothesis: "the considered populations of absolute residuals have identical distributions", thus allowing us to assess if the differences exist due to chance alone or due to the fact that the samples are from different populations (Kitchenham, Pickard, MacDonell, and Shepperd, 2001).

EVOLUTIONARY BASED APPROACHES FOR EFFORT ESTIMATION

On the basis of the observations in the previous section, it is clear that the problem of identifying an estimation method on the basis of historical data can be seen as the problem of finding an estimation method that minimise the residual values, i.e. the difference between the actual and predicted efforts. Thus, it can be seen as an optimisation problem and evolutionary based approaches could be exploited to address it. As a matter of fact, in the last years genetic algorithms

(GAs), which are based on the evolutionary ideas of natural selection (Goldberg, 1989), have been defined to estimate software development effort, e.g., (Burgess and Lefley, 2001; Dolado, 2000; Lefley and Shepperd, 2003; Shan *et al.*, 2002). At the same time, some other approaches have been proposed aiming to improve some existing Model Based techniques by suitably combining them with genetic algorithms, e.g., (Braga, Oliveira, and Meira, 2008; Chiu and Huang, 2007; Koch and Mitlöhner, 2009; Shukla, 2000).

In this section, we first describe the genetic approaches proposed in the literature and report on the most relevant empirical studies conducted to assess their effectiveness in estimating software development effort. Then, we summarise some studies that investigated whether the use of evolutionary based approaches can improve estimation accuracy of other techniques.

Genetic Algorithms

Basically a Genetic Algorithm (GA) simulates the evolution of natural systems, emphasising the principles of survival of the strongest, first set by Charles Darwin. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. GAs were first pioneered by John Holland in the 1960s (Holland, 1975). Then they have been extensively studied, experimented, and applied in many fields in the world of science and practice. It is important to note that GA not only provides an alternative method to solving problems, but, in several cases, it consistently outperforms other traditional methods (Goldberg, 1989; Harman and Jones, 2001).

In the computer implementation of a genetic algorithm, a crucial role is played by the solution representation. In general a solution for the problem being solved is represented by a fixed length binary string, which is called chromosome (in analogy with the biological equivalent). Each solution is evaluated using a fitness function that gives an indication of its goodness.

Despite of a number of variations, the elementary process of the genetic algorithm is the follows: (i) first a random initial population, i.e., a family of chromosome, is generated; (ii) then, a new population (i.e., generation) is created starting from the previous one by applying genetic operators (e.g., crossover, mutation) to the best chromosomes (according to the fitness value); (iii) the second step is repeated until either the fitness of the best solution has converged or a certain number of generations have been made. The chromosome that gives the best solution in the final population is taken in order to define the best approximation to the optimum for the problem under investigation.

The analysis of the process suggests that there are several key parameters that have to be determined for the application of GAs to any given optimisation problem (Goldberg, 1989; Harman and Jones, 2001). In particular, the following issues have to be addressed:

1. Defining the way for encoding a solution and the number of solutions (i.e. population size).
2. Choosing the fitness function, to measure the goodness of a solution;
3. Defining the combination of genetic operators, to explore the search space;
4. Defining the stopping criteria.

In the context of effort estimation, a solution consists of an estimation model described by an equation that combines several factors, i.e.,

$$Effort = c_1\ op_1\ f_1\ op_2 \ldots op_{2n-2}\ c_n\ op_{2n-1}\ f_n\ op_{2n}\ C \quad (1)$$

where f_i represents the value of the i^{th} factor and c_i is its coefficient, C represents a constant, while $op_i \in \{+, -, \cdot\}$ represents the i^{th} operators of the model.

Table 1. The settings of the genetic algorithms defined for effort estimation

Reference	Fitness function	Genetic operators	Size of population	Number of evolutions
(Burgess and Lefley, 2001)	MMRE	Basic crossover, mutation, and selection	1000	500
(Dolado, 2000)	MSE	Basic crossover, mutation, and selection	25	3,4,5
(Lefley and Shepperd, 2003)	MMRE	Basic crossover, mutation, and selection	1000	500
(Shan *et al.*, 2002)	MSE	Basic crossover, mutation	1000	200

As for its encoding this equation can be implemented in several ways, e.g. by a fixed binary string or a binary tree. Regarding the fitness function, the fitness value of a solution can be determined by using summary measures, usually employed to evaluate the goodness of obtained effort estimates (Briand *et al.*, 1999; Briand *et al.*, 2000; Briand and Wieczorek, 2002; Finnie, Wittig, and Desharnais, 1997; Kitchenham *et al.*, 2001; Shepperd and Schofield, 2000), such as MMRE, Pred(25), and MSE (Conte, Dunsmore, and Shen, 1986). Finally, selection, crossover, and mutation are the genetic operators widely employed to explore the search space in order to find good solutions.

Empirical Studies That Investigated Evolutionary Approaches to Estimate Software Development Effort

Table 1 reports on the settings of the genetic algorithms proposed in the literature to estimate software development effort highlighting the main GAs control parameters, such as the employed fitness functions, the population size, the genetic operators, and the maximum number of evolutions. As we can observe, to evaluate the goodness of a solution three proposals employed MSE (Dolado, 2000) as fitness function and two used MMRE. Moreover, all the proposals exploited the genetic operators originally defined by Holland (1975), namely crossover, mutation and selection operator, except for (Shan *et al.*, 2002) that did not use the selection operator. As for the population size and the evolutions number, almost all settings employed a population of 1000 chromosomes and a number of evolutions ranging from 200 to 500. Only one setting (Shukla, 2000) exploited very small values for these control parameters.

Table 2 summaries the main aspects of the empirical studies carried out to assess the proposed genetic algorithms, e.g. the employed dataset, validation method, evaluation criteria, benchmark method, and the number of runs of the genetic algorithms (i.e. the number of test performed for each validation round to ensure reliability of the results). It is worth noting that all the case studies performed a hold-out validation on industrial datasets except for one (Dolado, 2000) that employed a dataset which contains academic projects. All the industrial datasets has been widely used in effort estimation case studies; they come from a single company, e.g., the Desharnais dataset (Desharnais, 1989; Finnie *et al.*, 1997) or from multiple companies, e.g., Finnish (Burgess and Lefley, 2001) and ISBSG (ISBSG, 2009) datasets. The criteria used to evaluate the accuracy of the obtained estimates are all based on summary measures; in particular MMRE and Pred(25) are employed in all case studies, while fewer studies exploited MSE, AMSE, BMMRE, and Pred(50). In order to assess reliability of accuracy of the obtained estimates for each case study at least five test runs are performed. Finally, in order to assess the effectiveness of GAs for effort estimation, in each case study the proposed genetic-based approaches are compared with other benchmark

Table 2. Summary of the empirical studies that assessed genetic algorithms for effort estimation

Reference	Case study dataset	Validation method	Evaluation criteria	Benchmark methods	Number of test runs
(Burgess and Lefley, 2001)	Finnish Dataset (Burgess and Lefley, 2001)	hold-out validation training set: 149 test set: 15	AMSE MMRE BMMRE Pred(25)	Artificial Neural Networks Linear Regression CBR (2 Nearest Neighbours) CBR (5 Nearest Neighbours)	10
(Dolado, 2000)	46 academic projects (Conte, Dunsmore, and Shen, 1986)	hold-out validation training set: 30 test set: 16	MMRE Pred(25)	Linear Regression Artificial Neural Networks	>15
(Lefley and Shepperd, 2003)	Desharnais (Desharnais, 1989; Finnie et al., 1997)	hold-out validation training set: 63 test set: 18	AMSE MMRE BMMRE Pred(25)	Artificial Neural Networks Linear Regression CBR (2 Nearest neighbours) CBR (5 Nearest neighbours)	10
(Shan *et al.*, 2002)	423 projects from the ISBSG dataset (ISBSG, 2009)	hold-out validation: training test: 211 test set: 212	MMRE Pred(25) Pred(50) MSE	Linear Regression	5

methods, such as Linear Regression, Case-Based Reasoning, and Artificial Neural Networks.

In the following, we provide more details for each proposal, highlighting the validation results. Dolado (2000) was the first to employ an evolutionary approach in order to automatically derive equations alternative to multiple linear regression. The aim was to compare the linear equations with those obtained automatically. The proposed algorithm was run a minimum of 15 times and each run had an initial population of 25 equations. Even if in each run the number of generation varied, the best results were obtained with three to five generations (as reported in the literature, usually more generations are used) and by using MSE as fitness function. As dataset, 46 projects developed by academic students (using Informix-4GL) were exploited through a hold-out validation. It is worth noting that the main goal of Dolado work was not the assessment of evolutionary algorithms but the validation of the component-based method for software sizing. However, he observed that the investigated algorithm provided similar or better values than regression equations.

Burgess and Lefley (2001) performed a case study using the Desharnais dataset (Desharnais, 1989) to compare the use of GA for estimating software development effort with other techniques, such as LR, CBR, and ANN (Artificial Neural Networks). The comparison was carried out with respect three dimensions, namely estimation accuracy, transparency, and ease of configuration. The settings they used for the employed genetic algorithm were: an initial population of 1000, 500 generations, 10 executions, and a fitness function designed to minimise MMRE. They compared the accuracy of the analysed estimation techniques by taking into account summary statistics based on MRE, namely MMRE, Pred(25), BMMRE, and AMSE. Even if GA did not outperform the other techniques the results were promising and Burgess and Lefley suggested that a better set up of the evolutionary algorithm could improve the accuracy of the estimations. In particular they highlighted that the use of a fitness function specifically tied to optimise one particular measure could degrade the other evaluation measures. As a matter of fact GA obtained the best estimates in terms of MMRE and the worst in terms of the other summary measures AMSE, Pred(25), BMMRE.

As for the transparency of the solution, the authors highlighted that widely used techniques

such as LR and CBR allowed the user to have a deep insight on the problem making explicit any information about the contribution of each variables in the prediction model and the degree of similarity to the target project respectively. GAs also produced transparent solution because the solution is an algebraic expression, while neural networks did not make explicit any information. As for the ease of configuration, i.e. the effort required to build the prediction system, LR and CBR were easy to use because are widely used method often well supported by tool (Shepperd and Schofield, 2000). Neural networks and GA approaches required instead some effort to choose appropriate values for control parameters because different settings may be lead to different results.

Successively, Shepperd and Lefley (2003) also assessed the effectiveness of an evolutionary approach and compared it with several estimation techniques such as LR, ANN, and CBR. As for genetic algorithm setting, they applied the same choice of Burgess and Lefley (2001), while a different dataset was exploited. This dataset is refereed as "Finnish Dataset" and included 407 observations and 90 features, obtained from many organisations. After a data analysis, a training set of 149 observations and a test set of 15 observations were used for a hold-out validation. Even if the results revealed that there was not a method that provided better estimations than the others, the evolutionary approach performed consistently well. In particular the proposed approach applied on general company wide data obtained the best results in terms of AMSE, MMRE and Pred(25), while on the company specific dataset the best results were achieved only in terms of MMRE and BMMRE. However, the authors again observed that the algorithm was quite hard to configure and companies have to weigh the complexity of the algorithm against the small increases in accuracy to decide whether to use it to estimate development effort (Lefley and Shepperd, 2003).

An evolutionary computation method, named Grammar Guided Genetic Programming (GGGP), was proposed in (Shan *et al.*, 2002) to overcome some limitations of GAs, with the aim of improving the estimation of the software development effort. Indeed they proposed to use grammars in order to impose syntactical constraints and incorporate background knowledge aiming to guide the evolutionary process in finding optimal or near-optimal results. Data of software projects from the ISBSG (ISBSG, 2009) dataset was used to build the estimation models using GGGP and LR. The fitness function was designed to minimise MSE, an initial population of 1000 was chosen, the maximum number of generations was 200, and the number of executions was 5. The models were built and validated performing a hold-out validation with training and test sets of the same size. The results revealed that GPPP performed better than Linear Regression on all the exploited evaluation criteria, not just on the MSE, the criterion that was used as fitness function.

Empirical Studies That Investigated the Use of Evolutionary Based Approaches to Improve the Effectiveness of Some Existing Estimation Techniques

Some efforts have been made to improve the estimation performance of existing estimation techniques combining them with genetic algorithms. These are summarised in Table 3. As we can observe, three of the six proposed approaches combines GA with CBR, the other ones combine GA with less frequently used techniques, such as Artificial Neural Networks (ANN), Support Vector Regression (SVR) and Gray Relational Analysis (GRA). Concerning the GAs settings all the empirical studied exploited basic crossover, mutation and selection operators except for some settings (Braga *et al.*, 2008; Shukla, 2000) that employed a multi-point crossover. To evaluate the goodness of a solution two settings exploited a combination of MMRE and Pred(25) as fitness function and three settings used a fitness function based only

Table 3. Combinations of genetic algorithms with other estimation techniques

Reference	Employed technique	Fitness function	Genetic operators	Number of evolutions	Size of population
(Braga *et al.*, 2008)	GA + SVR	(100 − PRED(25)) + MMRE	Two-point crossover, mutation, and selection	25	500
(Chiu and Huang, 2007)	GA + CBR	(MMRE - Pred(25))	Basic crossover, mutation, and selection	1000*number of Variables	10* number of Variables
(Huang *et al.*, 2008).	GA + GRA	MMRE	Basic crossover, mutation, and selection	1000*number of Variables	10* number of Variables
(Koch and Mitlöhner, 2009)	GA + CBR	MMRE	Basic crossover, mutation, and selection	1000	2000
(Li *et al.*, 2009)	GA + CBR	MMRE	Basic crossover, mutation, and selection	1000*number of Variables	10* number of Variables
(Shukla, 2000)	GA + NN	MSE	Special MRX multi-point crossover, mutation, and selection	100	400

on MMRE. Only one setting used MSE values as fitness function. Concerning population size and evolutions number, Chiu and Huang (2007) and Huang *et al.* (2008) exploited a population composed by 10*V chromosomes, where V is the number of variables to be explored for GA. Moreover they proposed to stop the generation process not only after the execution of a fixed number of generations (e.g., 1000*V trials) but also if the solution does not change after a fixed number of generations (e.g. 100*V trials). The case study proposed in (Li, Xie, and Goh, 2009) followed this guidelines, while other case studies (Braga *et al.*, 2007; Koch and Mitlöhner, 2009; Shukla, 2000) employed instead a fixed number of evolutions and a population composed by 400, 500 and 2000 chromosomes, respectively.

Table 4 summarizes the main aspects of the empirical studies carried out to assess the above proposals. All the case studies employed industrial dataset and several validation methods were applied, such as k-fold cross-validation (Chiu and Huang, 2007; Huang *et al.*, 2008; Shukla, 2000), leave-one-out cross-validation (Braga *et al.*, 2007; Koch and Mitlöhner, 2009), and hold-out validation (Li *et al.*, 2009). Almost case studies employed the summary measures MMRE, MdMRE, and Pred(25) to evaluate the accuracy of the obtained estimates. Only one case study used a statistical significance test to evaluate the residuals, i.e. the difference between actual and predicted effort. Several estimation methods are employed as benchmark in each case study, ranging from widely used techniques, such as Constructive Cost Model (COCOMO), Case-Based Reasoning (CBR), Classification and Regression Trees (CART), and (Linear Regression) LR, to less frequently used techniques, e.g. Grey Relational Analysis (GRA), Step-wise Regression (SWR), Artificial Neural Network (ANN) and its variants. In the following we provide a brief description for each proposal.

The first attempt to combine evolutionary approaches with an existing effort estimation technique was made by Shukla (2000) applying genetic algorithms to Neural Networks (NN) predictor (namely, neuro-genetic approach, GANN) in order to improve its estimation capability. In particular, in the proposed approach the role played by NN was learning the correlation that exists between project features and actual effort and also learning any existing correlations among the predictor variables, while the GA had to minimise MSE values. The proposed case study exploited

Table 4. Summary of the empirical studies that assessed the use of estimation techniques in combination with genetic algorithms

Reference	Employed techniques	Case study dataset	Validation method	Evaluation criteria	Benchmark methods	Number of test runs
(Braga *et al.*, 2008)	GA + SVR	Desharnais (Desharnais, 1989; Finnie et al., 1997) and NASA (Oliveira, 2006; Shin and Goel, 2000)	leave-one-out cross-validation	MMRE Pred(25)	Support Vector Regression and its variants	10
(Chiu and Huang, 2007)	GA + CBR	Canadian Financial (Abran and Robillard, 1996) and IBM DP service (Matson, Barrett, and Mellichamp, 1994)	3-fold cross-validation	MMRE MdMRE Pred(25)	Ordinary Least Square Regression Artificial Neural Networks CART	-
(Huang et al., 2008)	GA + GRA	Albrecht (Albrecht and Gaffney, 1983) and COCOMO (Abran and Robillard, 1996; Boehm, 1981)	3-fold cross-validation	MMRE Pred(25)	CBR Artificial Neural Networks CART	-
(Koch and Mitlöhner, 2009)	GA + CBR + social choice	Albrecht (Albrecht and Gaffney, 1983), ERP (Bernroider and Koch, 2001), and COCOMO (Boehm, 1981)	leave-one-out cross-validation	MMRE MdMRE Pred(25)	COCOMO Basic Model Neural Networks Linear Regression Grey Relational Analysis CBR CART	-
(Li et al., 2009).	GA + CBR	Desharnais (Desharnais, 1989; Finnie et al., 1997), Albrecht (Albrecht and Gaffney, 1983), and two artificial datasets (Li et al., 2009)	hold-out validation training set and test set of the same size	MMRE MdMRE Pred(25)	CBR Support Vector Regression Artificial Neural Networks CART	-
(Shukla, 2000)	GA + NN	78 software projects from (Boehm, 1981) and (Kemerer, 1987)	10-fold cross validation training sets: 63 test sets: 15	Student's t-test	Regression Tree based NN (Srinivasan and Fisher, 1995), Back-Propagation trained NN (Rumelhart *et al.*, 1988), Quick Propagation trained NN (Parlos et al., 1994)	10

as dataset information from 78 software projects, obtained from the combination of the COCOMO (Boehm, 1981) and the Kemerer (1987) datasets and a statistical significance test was employed to assess whether the neuro-genetic approach provided significant improvement respect of common used AI-oriented methods (Parlos, Fernandez, Atyla, Muthusami, and Tsai, 1994; Rumelhart, Hinton, Williams, 1988; Srinivasan and Fisher, 1995). In particular the employed Student's t-test revealed that the mean prediction error for GANN is less to that for CARTX and less to that for Quick Propagation trained NN (Parlos *et al.*, 1994). These results showed that GANN obtained significantly better prediction than CARTX and QPNN. It is worth to nothing that the authors highlighted that the employed chromosome encoding played a crucial role in the NN predictor system and that a

number of experiments were needed to determine a suitable choose.

Recently, Chiu and Huang (2007) applied GA to another AI-based method such Case-Base Reasoning obtaining interesting results. In particular, GA was adopted to adjust the reused effort obtained by considering similarity distances between pairs of software projects. As for the application of CBR, three similarity distances were considered, Euclidean, Minkowski, and Manhattan distances, and a linear equation was used to adjust the reused effort. As for the application of GA, the population included 10*V chromosomes and the generation was stopped after 1000*V trials, or when the best results did not change after 100*V trials, where V is the number of variables that GA explored. The performed case study exploited two industrial datasets (Abran and Robillard, 1996; Matson *et al.*, 1994) of 23 and 21 observations respectively and the results based on the MMRE, Pred(25) and MdMRE evaluation criteria revealed that the adjustment of the reused effort obtained by applying GA improved the estimations of CBR even if the achieved accuracy did not satisfy threshold proposed by Conte et *al*. (1986). As a matter of fact applying the proposed approach on the IBM DP service (Matson *et al.*, 1994) dataset an improvement of 58% and 126% is reached in terms of MMRE and Pred(25) respectively, but the obtained values were far enough from the proposed threshold (i.e. MMRE=0.52, Pred(25)=0.43).

Furthermore, the proposed approaches was also comparable with the models obtained by applying traditional techniques such as ordinary least square regression (OLS), CART and ANN, on both the exploited datasets.

In (Li *et al.*, 2009) it was also proposed a combination of evolutionary approach with CBR aiming at exploiting genetic algorithms to simultaneously optimize the selection of the feature weights and projects. The proposed GA worked on a population of 10*V chromosomes and explored the solution space to minimize MMRE value by considering 1000*V evolutions, where V is the number of variables. As for CBR method the authors exploited several combinations of similarity measures, K value, and solution functions. The performed case study employed a hold-out validation on two industrial datasets (Albrecht and Gaffney, 1983; Desharnais, 1989) and two artificial datasets. The obtained estimates were compared with those achieved by applying only CBR and the results showed that the use of GA can provide significantly better estimations even if there was no clear conclusion about the influence of similarity and solution functions on the method performance. It is worth to nothing that on the Desharnais (Desharnais, 1989; Finnie *et al.*, 1997) and Albrecht (Albrecht and Gaffney, 1983) dataset the accuracies of the obtained estimates did not satisfy the threshold proposed by Conte et *al*. (1986), while this was true for the results obtained applying the proposed approach on the two artificial datasets.

The GA method was also used to improve the accuracy of an effort estimation model built by combining social choice and analogy-based approaches (Koch and Mitlöhner, 2009). In particular, voting rules were used to rank projects determining similar projects and GA method was employed to find suitable weights to be associated to the project attributes. To this end, a weight between 0 and 99 was assigned to each attribute and GA started with a population of 2000 random weight vectors. By exploiting error based on summary measures, the proposed GA searched through 1000 generations an optimal assignment for the weights. The validation of the obtained weighted model was performed with a leave-one-out approach by considering as dataset those used in (Albrecht and Gaffney, 1983; Bernroider and Koch, 2001; Boehm, 1981). The accuracy of the proposed model was compared with that obtained by applying other estimation techniques, such as LR, ANN, CART, COCOMO, and GRA. The results revealed that the proposed approach provided the best value for Pred(25) but the worst MMRE value with respect to the other techniques.

Finally, we report on two case studies carried out to investigate the combination of GA with techniques not frequently employed for effort estimation. Braga *et al.* (2008) exploited the use of GAs with Support Vector Regression (SVR) (Cortes and Vapnik, 1995), a machine learning techniques based on statistical learning theory, building a regression model employed to predict effort of novel projects on the basis of historical data. In particular they exploited a GA previously used to solve classification problems (Huang and Wang, 2006) to address the problems of feature selection and SVR (Cortes and Vapnik, 1995) parameters optimisation aiming to obtain better software effort estimations. The proposed GA started with a population of 500 chromosomes and used roulette wheel selection, two-point crossover, mutation, and elitism replacement to create 25 generations. A combination of MMRE and Pred(25) is used as fitness function. To evaluate the proposed method they used two datasets, namely Desharnais (Desharnais, 1989) and NASA (Oliveira, 2006; Shin and Goel, 2000), and performed 10 runs for each dataset. The results showed that the proposed GA-based approach was able to improve the performance of SVR and outperformed some recent results reported in the literature (Braga, Oliveira, and Meira, 2007-a; Braga, Oliveira, Ribeiro, and Meira, 2007-b; Oliveira, 2006; Shin and Goel, 2000). It is worth nothing to note that the results obtained applying the proposed approach on NASA dataset satisfied the threshold proposed by Conte *et al.* (1986). On the other hand applying the same method on Desharnais dataset the obtained MMRE value is not less than 0.25, while the Pred(25) is greater than 0.75.

Chiu and Huang (2007) integrated a genetic algorithm to the Grey Relational Analysis (GRA) (Deng, 1989) method to build a formal software estimation method. Since GRA is a problem-solving method that is used to deal with similarity measures of complex relations, the GA method was adopted in the GRA learning process to find the best fit of weights for each software effort driver in the similarity measures. To this end the weights of each effort driver were encoded in a chromosome and the MMRE was the value to be optimized. A case study was performed by exploiting the COCOMO (Boehm, 1981) and the Albrecht datasets (Albrecht and Gaffney, 1983) and the experimental results showed that when GA was applied to the former dataset the accuracies of the obtained estimates outperformed those obtained using CBR, CART, and ANN, while on Albrecht dataset all the exploited methods achieved a comparable accuracy. In both cases the accuracy obtained applying the proposed approach did not satisfy the thresholds proposed by Conte *et al.* (1986).

FUTURE RESEARCH DIRECTIONS

The results so far obtained using genetic algorithms for effort estimation are comparable with those provided by other widely used estimation techniques, in terms of the summary measures adopted for the comparison (such as MMRE and Pred(25)). The analysis of the case studies reported in the previous sections has also highlighted that genetic algorithms can be suitably combined with existing techniques to improve their estimation accuracy. Thus, further investigations deserve to be carried out to analyze the usefulness of genetic algorithms in the context of effort estimation. In particular, as highlighted also by (Burgess and Lefley, 2001; Lefley and Shepperd, 2003; Li *et al.*, 2009; Shukla, 2000) specific empirical studies should be performed to understand the role played by different configurations of the evolutionary based approaches to improve the obtained estimates. These configurations could be based on:

- Several fitness functions possibly not employed in the previous works. Indeed, the choice of fitness function could influence the achieved results (Harman, 2007), particularly when the measure used by the

algorithm to optimise the estimates is the same used to evaluate the accuracy of them (Burgess and Lefley, 2001). An example of novel fitness function might be a combination of Pred(25) and MMRE. Such a combination can be achieved maximising the ration between these two metrics.

- Aggregated fitness or Pareto optimality to consider a multi-objective optimisation (Harman, 2007);
- An interactive optimisation approach where users can influence the evaluation of fitness (Harman, 2007).
- Other combinations of genetic operators, to better explore the solution space.

It is important to stress that future investigations should use statistical significance tests for the comparisons to give a deeper insight on the significance of the results. It is worth noting that empirical studies carried out so far have been rarely used such tests.

At the same time other combinations of evolutionary algorithms with existing estimation techniques could be explored. For example, a genetic algorithm could be exploited to optimise a crucial step in the application of the CBR technique, namely the feature subset selection. Moreover, other evolutionary algorithms should be investigated (Harman, 2007), such as Tabu Search (Glover and Laguna, 1997), never used for estimating software development effort, and Simulated Annealing (Metropolis, Rosenbluth, Rosenbluth, Teller and Teller, 1953), employed only in a very preliminary case study (Uysal, 2008). The estimation process should also combine two heuristic methods. In particular, a global search can be performed to find a set of good estimation models. To this end a Genetic Algorithm (Goldberg, 1989) or a Particle Swarm Optimization algorithm (Kennedy and Eberhart, 1995) can be used. Then, this set is refined in a second step in order to achieve the best estimation model. Such a step can be performed by using a local search, e.g., a Simulated Annealing algorithm (Metropolis et al., 1953).

Finally, the observation that so far there is no study that took into account new emerging development environments, such as model-driven development, agile techniques, web applications, suggests the need to apply evolutionary approaches in these contexts.

REFERENCES

Abran, A., & Robillard, P. N. (1996). Function Points Analysis: An Empirical Study of its Measurement Processes. *IEEE Transactions on Software Engineering*, *22*(12), 895–910. doi:10.1109/32.553638

Albrecht, S. J., & Gaffney, J. R. (1983). Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. *IEEE Transactions on Software Engineering*, *9*(6), 639–648. doi:10.1109/TSE.1983.235271

Bernroider, E., & Koch, S. (2001). ERP Selection Process in Midsize and Large Organizations. *Business Process Management Journal*, *7*(3), 251–257. doi:10.1108/14637150110392746

Boehm, B. W. (1981). *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice-Hall.

Braga, P. L. Oliveira, A. L. I. & Meira, S. R. L. (2008). A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation. In *Proceedings of the 23rd ACM symposium on Applied computing*, Galé in Fortaleza, Ceará, Brazil, (pp. 1788-1792).

Braga, P. L., Oliveira, A. L. I., & Meira, S. R. L. (2007-a). Software Effort Estimation Using Machine Learning Techniques with Robust Confidence Intervals. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence*, Patras, Greece, (vol. 1, pp. 181-185).

Braga, P. L., Oliveira, A. L. I., Ribeiro, G. H. T., & Meira, S. R. L. (2007-b). Bagging Predictors for Estimation of Software Project Effort. In *Proceedings of the 20th IEEE International Joint Conference on Neural Networks*, Orlando, FL, (pp. 1595-1600).

Briand, L., El Emam, K., Surmann, D., Wiekzorek, I., & Maxwell, K. (1999). An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques. In *Proceedings of the 21st International Conference on Software Engineering*, Los Angeles, CA, (pp. 313–322).

Briand, L., Langley, T., & Wiekzorek, I. (2000). A Replicated Assessment and Comparison of Common Software Cost Modeling Techniques. In *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, (pp. 377–386).

Briand, L. C., & Wieczorek, I. (2002). *Software Resource Estimation* (pp. 1160–1196). Encyclopaedia of Software Engineering.

Burgess, C. J., & Lefley, M. (2001). Can Genetic Programming Improve Software Effort Estimation: A Comparative Evaluation. *Information and Software Technology*, *43*(14), 863–873. doi:10.1016/S0950-5849(01)00192-6

Chiu, N. H., & Huang, S. (2007). The Adjusted Analogy-based Software Effort Estimation based on Similarity Distances. *Journal of Systems and Software*, *80*(4), 628–640. doi:10.1016/j.jss.2006.06.006

Conte, D., Dunsmore, H., & Shen, V. (1986). *Software Engineering Metrics and Models*. San Francisco, CA: The Benjamin/Cummings Publishing Company, Inc.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. doi:10.1007/BF00994018

Deng, J. (1989). Introduction to Grey System Theory. *Journal of Grey System*, *1*, 1–24.

Desharnais, J. M. (1989). *Analyse Statistique de la Productivitie des Projets Informatique a Partie de la Technique des Point des Function*. Unpublished Masters Thesis, University of Montreal, Montreal, Canada.

Dolado, J. J. (2000). A Validation of the Component-based Method for Software Size Estimation. *IEEE Transactions on Software Engineering*, *26*(10), 1006–1021. doi:10.1109/32.879821

Finnie, G. R., Wittig, G. E., & Desharnais, J. M. (1997). A Comparison of Software Effort Estimation Techniques: Using Function Points with Neural Networks, Case-based Reasoning and Regression Models. *Journal of Systems and Software*, *39*(3), 281–289. doi:10.1016/S0164-1212(97)00055-1

Glover, F., & Laguna, M. (1997). *Tabu Search*. Boston: Kluwer Academic Publishers.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Harman, M. (2007). The Current State and Future of Search Based Software Engineering. In Proceedings of Future of Software Engineering, Minneapolis, MN, (pp. 342-357).

Harman, M., & Jones, B. F. (2001). Search based software engineering. *Information and Software Technology*, *43*(14), 833–839. doi:10.1016/S0950-5849(01)00189-6

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press.

Huang, C. L., & Wang, C. J. (2006). A GA-based Feature Selection and Parameters Optimization for Support Vector Machines. *Expert Systems with Applications*, *31*(2), 231–240. doi:10.1016/j.eswa.2005.09.024

Huang, S. J., Chiu, N. H., & Chen, L. W. (2008). Integration of the Grey Relational Analysis with Genetic Algorithm for Software Effort Estimation. *European Journal of Operational Research*, *188*(3), 898–909. doi:10.1016/j.ejor.2007.07.002

ISBSG. (2009). *International Software Benchmarking Standards Group*. Retrieved from www.isbsg.org

Kemerer, C. F. (1987). An Empirical Validation of Software Cost Estimation Models. *Communications of the ACM*, *30*(5), 416–429. doi:10.1145/22899.22906

Kennedy, J., & Eberhart, R. C. (1995). Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks,* Beijing, China, (pp. 1942–1948).

Kitchenham, B., Pickard, L. M., MacDonell, S. G., & Shepperd, M. J. (2001). What Accuracy Statistics Really Measure. *IEEE Software*, *148*(3), 81–85. doi:10.1049/ip-sen:20010506

Koch, S., & Mitlöhner, J. (2009). Software Project Effort Estimation with Voting Rules. *Decision Support Systems*, *46*(4), 895–901. doi:10.1016/j.dss.2008.12.002

Lefley, M., & Shepperd, M. J. (2003). Using Genetic Programming to Improve Software Effort Estimation based on General Data Sets. In *Proceedings of Genetic and Evolutionary Computation Conference*, Chicago, (pp. 2477–2487).

Li, Y. F., Xie, M., & Goh, T. N. (2009). A Study of Project Selection and Feature Weighting for Analogy based Software Cost Estimation. *Journal of Systems and Software*, *82*(2), 241–252. doi:10.1016/j.jss.2008.06.001

Matson, J. E., Barrett, B. E., & Mellichamp, J. M. (1994). Software Development Cost Estimation using Function Points. *IEEE Transactions on Software Engineering*, *20*(4), 275–287. doi:10.1109/32.277575

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., & Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, *21*(6), 1087–1092. doi:10.1063/1.1699114

Oliveira, A. L. I. (2006). Estimation of Software Project Effort with Support Vector Regression. *Neurocomputing*, *69*(13-15), 1749–1753. doi:10.1016/j.neucom.2005.12.119

Parlos, A. G., Fernandez, B., Atyla, A., Muthusami, J., & Tsai, W. (1994). An Accelerated Algorithm for Multilayer Preceptor Networks. *IEEE Transactions on Neural Networks*, *5*(3), 493–497. doi:10.1109/72.286921

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning representations by back-propagating errors. In Neurocomputing: foundations of research, (pp. 696-699).

Shan, Y., Mckay, R. I., Lokan, C. J., & Essam, D. L. (2002). Software Project Effort Estimation using Genetic Programming. In *Proceedings of International Conference on Communications Circuits and Systems*, Chengdu, China, (Vol. 2, pp. 1108–1112).

Shepperd, M., & Schofield, C. (2000). Estimating Software Project Effort using Analogies. *IEEE Transactions on Software Engineering*, *23*(11), 736–743. doi:10.1109/32.637387

Shin, M., & Goel, A. L. (2000). Empirical Data Modeling in Software Engineering using Radical Basis Functions. *IEEE Transactions on Software Engineering*, *26*(6), 567–576. doi:10.1109/32.852743

Shukla, K. K. (2000). Neuro-Genetic Prediction of Software Development Effort. *Information and Software Technology*, *42*(10), 701–713. doi:10.1016/S0950-5849(00)00114-2

Srinivasan, K., & Fisher, D. (1995). Machine Learning Approaches to Estimating Software Development Effort. *IEEE Transactions on Software Engineering*, *21*(2), 126–137. doi:10.1109/32.345828

Uysal, M. (2008). Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm. In *Proceedings of World Academy of Science* (pp. 258–261). Buenos Aires, Argentina: Engineering and Technology.