UNIVERSITÀ DEGLI STUDI DI SALERNO

Dottorato di Ricerca in Scienze Matematiche, Fisiche ed Informatiche

Curriculum in Sistemi e Tecnologie del Software

XI Ciclo

Search-Based Approaches for Software Development Effort Estimation

Doctoral Dissertation of

Federica Sarro

Coordinator

Prof. Patrizia Longobardi

Advisor

Prof. Filomena Ferrucci

ACKNOWLEDGEMENT

I am deeply grateful to my supervisor Professor Filomena Ferrucci for her continuous and invaluable advice, encouragement and support. I had the honor and privilege to benefit from the expertise of an extraordinarily passionate professor.

I wish to thank Dr. Carmine Gravino for all our valuable discussions and his advice which was always delivered with passion and rigor. I also wish to thank my colleagues for providing a stimulating and fun-filled environment and to my friends in Salerno and other parts of the world. They were always a great source of laughter, joy and support.

A special thanks to my mum and sister, for believing in my dreams, and to Maria Laura, whose endless love and encouragement allowed me to take this journey. I owe them everything and wish I could show them just how much I love and appreciate them.

To them and to the rest of my family I say, ‘my name is written on the first page but your names are written on every page because there wasn’t a moment of this journey that you weren’t by my side.’

Finally, I would like to dedicate this work to my father, who left us too soon. I hope that this work makes you proud.

Contents

INTRODUCTION

Effort estimation is a critical activity for planning and monitoring software project development and for delivering the product on time and within budget. Indeed, significant over or under-estimates expose a software project to several risks. As a matter of fact under-estimates could lead to addition of manpower to a late software project, making the project later (Brooks's Law), or to the cancellation of activities, such as documentation and testing, negatively impacting on software quality and maintainability. Thus, the competitiveness of a software company heavily depends on the ability of its project managers to accurately predict in advance the effort required to develop software system. However, several challenges exists in making accurate estimates, e.g., the estimation is needed early in the software lifecycle, when few information about the project are available, or several factors can impact on project effort and these factors are usually specific for different production contexts.

Several techniques have been proposed in the literature to support project managers in estimating software project development effort.

To date, expert opinion is a commonly used estimation method and is still used by software and Web companies [66]. However, relying on the expertise of the company's practitioners the results are less repeatable, being mainly based on subjective judgments [13]. Moreover, this made difficult to quantify and to determine those attributes that have been used to derive an estimate [89].

To overcome this limitation, several techniques which rely on a more formal approach have been proposed. These include the application of some algorithms to a number of factors that influence the development cost, such as the size, to produce an estimate or a model providing the estimation in an objective way. COCOMO and COCOMO II are probably the best known generic methods [13]. They are based on a regression formula, with parameters that are derived from some historical project data and current project characteristics. They are generic methods that often need to be calibrated to local data to take into account the characteristics of the specific production context. Alternatively, a software company can construct its specific model (or estimation) using an estimation technique that takes as input the information coming from past projects. Usually the employed data consist of information about some relevant factors (named cost drivers) and the effort

actually spent by the company to develop prior projects. In this class of data-driven estimation techniques we can find Linear and Stepwise Regression [13][89] and some artificial intelligence techniques, such as Classification and Regression Tree (CART), Case-Based Reasoning (CBR), and Bayesian Networks (BN) [89].

In the last years the use of search-based (SB) approaches has been suggested to be employed as an effort estimation technique. These approaches include a variety of meta-heuristics, such as local search techniques (e.g., Hill Climbing, Tabu Search, Simulated Annealing) or Evolutionary Algorithms (e.g., Genetic Algorithms, Genetic Programming). They search for suitable solutions to problems characterized by large search space, using an objective function that gives an indication of how a solution is suitable for the problem under investigation.

The generic nature of these meta-heuristics let them to be fruitful for different goals and issues, simply by redefining the solution representation and the objective function. As a matter of fact, in the last years there has been an explosion of researches on the use of SB techniques in many software engineering fields [55][56], giving rise to a very active field known as Search-Based Software Engineering (SBSE) [54]. The idea underlying the use of such techniques is based on the reformulation of software engineering problems as search or optimization problems whose goal is to find the most appropriate solutions that conform to some adequacy criteria (i.e., problem goals). In particular, the use of SB approaches in the context of effort estimation is twofold: they can be exploited to build effort estimation models or to enhance the use of other effort estimation techniques. In the first case the problem of building an estimation model is reformulated as an optimization problem where the SB method builds many possible models - exploiting past projects data - and tries to identify the best one, i.e., the one providing the most accurate estimates. In the second case, SB methods can be exploited in combination with other estimation techniques to improve critical steps of their application (e.g., features subset selection or the identification of critical parameters) aiming to obtain better estimates.

The usage reported in the literature of the SB approaches for effort estimation have provided promising results that encourage further investigations. However, they can be considered preliminary studies [2]. As a matter of fact, the capabilities of these approaches were not fully exploited, either the employed empirical analyses did not consider the more recent recommendations on how to carry out this kind of empirical assessment in the effort estimation and in the SBSE

contexts [4][6][73][135]. The main aim of the PhD dissertation is to provide an insight on the use of SB techniques for the effort estimation trying to highlight strengths and weaknesses of these approaches for both the uses above mentioned.

In particular, on the basis of the weakness highlighted in the state of the art, the research has been carried out aiming at answer the following questions:

- How the design choices characterizing the use of the SB approaches impact on the performance of these techniques?
- Are there any differences in the use of different SB techniques?
- Are the SB techniques as effective as widely used effort estimation methods?
- Are the SB techniques effective to improve the accuracy of other data-driven effort estimation techniques?

In particular, as for the first question, an often overlooked aspect of research on computational search algorithms lies in the selection and tuning of the algorithmic parameters. Let us observe that a suitable setting is usually obtained via a trial-and-error process for each new problem to be addressed. As a matter of fact, in previous work the number of solutions and iterations of search-based approaches was empirically determined carrying out a validation process with different values for these parameters and selecting the one providing the best results (see e.g., [42][43][44]). However, this practice is time consuming and it has to be repeated every time new data is used, thus limiting the adoption of search-based approaches by practitioners. To overcome this limitation we employed and assessed an heuristics originally suggested in [40] to set population size and generation number of a Genetic Algorithm. Moreover, special attention has been given to the role played by the use of different objective functions since this is the most important design choice to be made in the use of any SB technique. In particular, we experimented several objective functions based on both single and combined evaluation measures and assessed how the accuracy of GP and TS is affected by this choice. To answer the second question we designed and assessed the use of three different SB techniques, namely Hill Climbing, Tabu Search, and Genetic Programming, and compared them in terms of accuracy and cost-effectiveness.

To understand the actual effectiveness of SB effort estimation approaches, we compared them with baseline methods, such as the mean and median of effort, and some widely used effort estimation techniques, such as Manual Stepwise Regression (MSWR) [89] and Case-Based Reasoning (CBR) [122]. Indeed, if the investigated estimation method does not outperform the results achieved with these baseline methods it cannot be transferred to industry [89].

As for the last question, we employed SB techniques to configure Support Vector Regression (SVR) that is a new generation of Machine Learning algorithms that have turned out to be effective for effort estimation. Nevertheless, its prediction accuracy is heavily influenced by its parameter setting [25] and no general guidelines are available to select these parameters. Thus, we investigated the use of Tabu Search in combination with SVR to select the parameters of SVR to be employed for effort estimation.

The research has been carried out to verify the effectiveness of the proposed techniques in a quantitative and reproducible way carrying out several empirical studies carefully taking into account the biases that might affect the obtained results (i.e., threats to validity). To this end we performed several empirical research following the guidelines proposed in [4][73][135].

Preliminary results of this research have been published in [26][27][36][41][42][43][44][45][46][115].

Outline

The thesis is structured as follows. Chapter 1 provides background on software development effort estimation and search-based approaches. Chapter 2 discusses the work carried out so far on the use of search-based approaches for software development effort estimation. Chapter 3 focuses on the definition and assessment of three search-based approaches (i.e., Hill Climbing, Tabu Search and Genetic Programming) to build effort estimation models reporting results concerning with their setting, effectiveness, and comparison. Chapter 4 presents the results of the empirical analysis aiming to assess the impact of using different objective functions with both Tabu Search and Genetic Programming. Chapter 5 focuses on the use of Tabu Search to configure a machine learning technique for effort estimation, namely Support Vector Regression. Final remarks conclude the thesis.

CHAPTER 1: Background

1.1 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 [13]:

- 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 modeling techniques. A widely accepted taxonomy of estimation methods classified them in Non-Model Based and Model Methods [13]. 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, characterized 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) [13][14][15][122]. 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 ISBSG repository that contains data from a great number of projects developed by companies around the world [64]. 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 [13]. 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 (see, e.g. [14]). 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 compose the test set. Usually, about a third of the initial dataset is used as validation set.

To assess the accuracy of the derived estimations some evaluation criteria are proposed in the literature. Among them several summary measures, like *MMRE*, *MdMRE*, and *Pred(25)* [22], 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* [22] is defined as

$$MRE = \frac{|EFreal - EFpred|}{EFreal}$$

where *EFreal* and *EFpred* 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 [98].

The *Prediction at level l* [22] 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.* [22], 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.

Kitchenham *et al.* [72] suggest also the use of the Magnitude of Relative Error relative to the Estimate (EMRE). The *EMRE* has the same form of MRE, but the denominator is the estimate, giving thus a stronger penalty to under-estimates:

$$EMRE = EFreal - EFpred \setminus EFpred.$$

As with the MRE, we can also calculate the Mean of EMRE (*MEMRE*) and Median of EMRE (*MdEMRE*).

Other summary measures less frequently used are the *Balanced MMRE* (*BMMRE*), the *Mean Squared Error* (*MSE*) [22] and the *Adjusted Mean Square Error* (*AMSE*) [16]. They are defined as follows:

$$BMMRE = \sum_{i=1}^{n} \left(\frac{|EFreal_i - EFpred_i|}{min(EFreal_i, EFpred_i)} \right) \frac{100}{\mathrm{n}}$$

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (EFreal - EFpred)^2$$

$$AMSE = \sum_{i=1}^{n} \frac{|EFreal_i - EFpred_i|^2}{EFreal_i * EFpred_i}$$

where $EFReal_i$ and $EFpred_i$ 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 [72].

1.2 Search-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 search-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 [51], have been defined to estimate software development effort (e.g., [16][38][82][118]). At the same time, some other approaches have been proposed aiming to improve some existing techniques by suitably combining them with GA (e.g., [11][19][78][125]).

In the following sections we describe the three search-based we employed in our work and in the next chapter we report on the most relevant empirical studies conducted to assess their effectiveness in estimating software development effort.

1.2.1 Hill Climbing

Hill climbing starts from a randomly chosen candidate solution. At each iteration, the elements of a set of 'near neighbors' to the current solution are considered. Just what constitutes a near neighbor is

problem specific, but typically neighbors are a 'small mutation away' from the current solution. A move is made to a neighbor that improves fitness.

There are two choices: In next ascent hill climbing, the move is made to the first neighbour found to have an improved fitness. In steepest ascent hill climbing, the entire neighborhood set is examined to find the neighbor that gives the greatest increase in fitness. If there is no fitter neighbor, then the search terminates and a (possibly local) maxima has been found. Figuratively speaking, a 'hill' in the search landscape close to the random starting point has been climbed. Clearly, the problem with the hill climbing approach is that the hill located by the algorithm may be a local maxima, and may be far poorer than a global maxima in the search space. For some landscapes, this is not a problem because repeatedly restarting the hill climb at a different locations may produce adequate results (this is known as multiple restart hill climbing). Despite the local maxima problem, hill climbing is a simple technique which is both easy to implement and surprisingly effective [57][103].

1.2.2 Tabu Search

Tabu Search (TS) is a meta-heuristics search algorithm that can be used for solving optimization problems. The method was proposed originally by Glover to overcome some limitations of Local Search (LS) heuristics [50]. Indeed, while classical LS heuristics at each iteration constructs from a current solution *i* a next solution *j* and checks whether *j* is worse than *i* to determine if the search has to be stopped, a TS optimization step consists in creating from a current solution *i* a set of solutions *N(i)* (also called *neighboring solutions*) and selecting the best available one to continue the search. In particular, TS usually starts with a random solution and applies local transformations (i.e., *moves*) to the current solution *i* to create *N(i)*. When no improving neighboring solution exists, TS allows for a *climbing move*, i.e., a temporary worsening move can be performed. The search terminates when a stopping condition is met (e.g., a maximum number of iteration is reached). To determine whether a solution is worse (or better) than another an *objective function* is employed. In order to prevent loops and to guide the search far from already visited portions of the search space, some moves can be classified as *tabu* which means that are forbidden. The tabu moves can be stored in a list, named *Tabu List*, of fixed or variable length following a short-term (i.e., moves leading to already visited solutions are stored) or a long-term memory strategy (i.e., moves that have been performed several times are stored). Since tabu moves sometimes may prohibit attractive solution or

may lead to an overall stagnation of the searching process [50], the so called *aspiration criteria* can be used to revoke the tabu status of a move. A common aspiration criterion allows for a tabu move if it results in a solution which has an objective value better than the current solution.

To summarize, TS starting from a random solution, at each iteration explores a search space consisting of a set of moves. Such moves are often local transformations of the current solution and depend on the problem to be solved. Among these moves, the one that provides the best objective value and is not tabu or matches an aspiration criterion is selected to continue the search.

Thus, to tailor the TS meta-heuristics to a given problem we have to perform the following choices:

- define a representation of possible solutions and the way to generate the initial one;
- define local transformations (i.e., *moves*) to be applied to the current solution for exploring the *neighbor* solutions;
- choose a means to evaluate the neighborhood (i.e., an *objective function*), thus guiding the search in a suitable way;
- define the *Tabu list*, the *aspiration criteria*, and the *termination criteria*.

1.2.3 Genetic Algorithms

Basically Genetic Algorithms (GAs) simulate 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. Genetic Algorithms were first pioneered by John Holland in the 1960s [59]. 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 [51][54].

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 [51][54]. 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.

CHAPTER 2: Literature Review

Some investigations have been carried out so far on the use of SB approaches for effort estimation. These studies have provided promising results that encourage further investigations. However, they can be considered preliminary studies. As a matter of fact, the capabilities of SB approaches have not been fully exploited and often the empirical analyses have not taken into account the more recent recommendations on how to carry out this kind of empirical assessment in the effort estimation and in the SBSE contexts [4][6][73][135], as detailed in the follow.

2.1 Empirical studies that investigated search based approaches to estimate software development effort

Table 1 summarizes the main aspects (e.g., employed technique, dataset, validation method, and evaluation criteria) of the studies carried out so far to assess SB approaches for building effort estimation models.

First of all we observe that all the previous studies [16][38][82][118] employed Genetic Programming (GP) and no attempts have been reported on the use of other SB techniques (e.g., the ones based on local-search), although they have many similarities but also distinguishing features. Moreover, each SB technique has specific design choices that may affect the performance of the method. As an example, for GP we have to choose the solution encoding, the fitness function (i.e., objective function), the strategy for creating the initial population, the operators for mating and survival selection, the crossover and mutation operators, and the stopping criteria. The choice of the objective function is common to all the SB techniques and represents one of the most critical step since such function guides the search towards suitable solutions. In the context of effort estimation this choice should be based on a measure of model accuracy. The studies carried out so far exploited two measures as fitness function, namely MMRE [16] [82] and MSE [38] [118]. However, several measures have been proposed to evaluate effort estimation accuracy and all of them could be exploited as objective function [53]. Nevertheless, the use of multiple criteria has not been investigated although there are recommendations on the use of several different accuracy measures

to carry out a more reliable evaluation of estimation models. The existing studies have neither fully investigated the impact of the other design choices such as the stopping criterion and its impact on the method convergence.

Table 1. Summary of the empirical studies that assessed SB approaches for building effort estimation models

Reference	**Employed technique**	**Case study Dataset**	**Validation method**	**Evaluation Criteria**	**Benchmark Methods**
[16]	GP with MMRE as fitness function	Desharnais	hold-out training set: 149 test set: 15	AMSE, MMRE, BMMRE, Pred(25)	ANN, LR, CBR
[38]	GP with MSE as fitness function	Academic projects	hold-out training set: 30 test set: 16	MMRE, Pred(25)	LR, ANN
[82]	GP with MMRE as fitness function	Finnish	hold-out training set: 63 test set: 18	AMSE, MMRE BMMRE, Pred(25)	ANN, LR, CBR
[118]	GP with MSE as fitness function	ISBSG	hold-out training test: 211 test set: 212	MMRE, Pred(25) Pred(50), MSE	LR

Concerning the empirical analyses, all the studies employed only one dataset thus affecting their external validity. Moreover, a hold-out validation was applied, where the dataset is split into a training set used to build the estimation model and a test used to validate it. Unfortunately this procedure can be biased since the prediction performance may depend on how the dataset is split.

Regarding the evaluation criteria only summary measures were employed: in particular MMRE and Pred(25) in all the case studies, and in some cases also MSE, AMSE, BMMRE, and Pred(50).

As for the benchmarks, useful to understand the actual effectiveness of the proposed approach, all the case studies employed several estimation methods, such as Linear Regression (LR) and Case-Base Reasoning (CBR). However, often there is a lack of details about their application. As for

example, studies that employed LR did not state if the underlying assumptions were verified [71] while this aspect is crucial for the internal validity of the empirical study.

Finally, little attention has been given by previous studies to the random variation in results due to the non-deterministic nature of SB techniques: indeed, very few executions were performed and often only results related to the best execution were reported, thus affecting the conclusion validity of these case studies.

In the following, we provide some more details for each proposal, highlighting the validation results.

Dolado [38] 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 [16] performed a case study using the Desharnais dataset [33] to compare the use of genetic algorithms 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 minimize 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 GP 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 optimize 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 [122]. 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 [82] 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[16], while a different dataset was exploited. This dataset is refereed as "Finnish Dataset" and included 407 observations and 90 features, obtained from many organizations. 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 [82].

An evolutionary computation method, named Grammar Guided Genetic Programming (GGGP), was proposed in [118] 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 ISBSG [64] database was used to build the estimation models using GGGP and LR. The fitness function was designed to minimize 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.

2.2 Empirical studies that investigated search based approaches to improve the effectiveness of existing estimation techniques

Despite some efforts have been made to improve the estimation performance of existing estimation techniques combining them with genetic algorithms, many of the above limitations can be found also in the studies that assessed the use of SB approaches to improve existing effort estimation techniques [5][11][19][61][78][83]. As we can observe from Table 2, all the studies exploited Genetic Algorithms.

In particular, 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). 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 on MMRE. Only one setting used MSE values as fitness function.

As for the empirical studies carried out to assess the above proposals, we can observe that all the case studies employed industrial dataset and several validation methods were applied, such as k-fold cross-validation [19][61][125], leave-one-out cross-validation [11][78] and hold-out validation [83]. Almost all 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 COCOMO, CBR, CART, and LR, to less frequently used techniques (e.g. GRA, SVR, ANN and its variants).

In the following we provide a brief description for each proposal.

Table 2. Summary of the empirical studies that assessed the use of SB approaches in combination with existing estimation methods

Reference	**Employed technique**	**Case study Dataset**	**Validation Method**	**Evaluation Criteria**	**Benchmark Methods**
[11]	GA+SVR with MMRE and Pred(25) as fitness function	Desharnais and NASA	leave-one-out	MMRE, Pred(25)	SVR
[19]	GA+CBR with MMRE and Pred(25) as fitness function	Canadian Financial service and IBM DP	3-fold	MMRE, MdMRE, Pred(25)	OLSR, ANN, CART
[83]	GA+CBR with MMRE as fitness function	Desharnais, Albrecht, and two artificial datasets	hold-out	MMRE, MdMRE, Pred(25)	CBR, SVR, ANN, CART
[78]	GA+CBR with MMRE as fitness function	Albrecht, COCOMO, and ER	leave-one-out	MMRE, MdMRE, Pred(25)	COCOMO, NN, LR, GRA, CBR, CART
[61]	GA+GRA with MMRE as fitness function	Albrecht and COCOMO	3-fold	MMRE, Pred(25)	CBR, ANN, CART
[5]	GA+NN with MSE as fitness function	78 software projects	hold-out (n times) training sets: 63 test sets: 15	Student's t-test	Regression Tree NN. Back-Propagation NN, Quick Propagation NN

The first attempt to combine evolutionary approaches with an existing effort estimation technique was made by Shukla [125] 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 minimize MSE values. The proposed case study exploited as dataset information from 78 software projects, obtained from the combination of the COCOMO [9] and the Kemerer [68] 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 [127][111][108]. 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 [108]. 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 applied GA to another AI-based method such Case-Based Reasoning obtaining interesting results [19]. 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 [1][85] 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*. [22]. As a matter of fact applying the proposed approach on the IBM DP service [85] 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 [83] 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 [3][33] 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 [33] [48] and Albrecht [3] dataset the accuracies of the obtained estimates did not satisfy the threshold proposed by Conte et *al*. [22], while this was true for the results obtained applying the proposed approach on the two artificial datasets.

GA was also used to improve the accuracy of an effort estimation model built by combining social choice and analogy-based approaches [78]. In particular, voting rules were used to rank projects determining similar projects and GA 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 [3], [8], and [9]. 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.* [11] exploited the use of GAs with Support Vector Regression (SVR) [28], 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 [62] to address the problems of feature selection and SVR [28] parameters optimization 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 [33] and NASA [107][124], 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 [10][12][107][124]. 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.* [22]. 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 in [61] integrated a GA to the Grey Relational Analysis (GRA) [32] 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 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 [9] and the Albrecht [3] datasets 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.* [22].

CHAPTER 3: Using Search-based Approaches for Building Effort Estimation Models

In the last decades, several methods have been proposed to estimate software development effort, among them data-driven methods exploit data from past projects to estimate the effort for a new project under development [14][15]. These data consist of information about some relevant factors (named cost drivers) and the effort actually spent to develop the projects. Usually a data-driven method tries to explain the relation between effort and cost drivers building an estimation model (equation) that is used to estimate the effort for a new project. Linear (StepWise) Regression [13] is a well known and widely used data-driven approach. Also search-based methods have been suggested to build effort estimation models [54]. The suggestion is based on the observation that, among possible estimation models, we have to identify the best one, i.e., the one providing the most accurate estimates. Thus, the effort estimation problem can be formulated as an optimization problem that can be addressed by search-based methods. Indeed, these meta-heuristics are able to find optimal or near optimal solutions to problems characterized by large space, using an objective function that gives an indication of how a solution is suitable for the problem under investigation. Examples of search-based methods are Simulated Annealing (SA), Tabu Search (TS), Hill Climbing (HC), Genetic Algorithms (GA) and Genetic Programming (GP), which differ under several aspects including the kind of employed search (local or global).

Some investigations have been reported in the literature on the use of search-based techniques for effort estimation. They showed some potentialities of these metaheuristics to build accurate estimation models as well as some difficulties to adopt them mainly related to the interpretation of solutions and the choice of a suitable setting. Nevertheless those previous studies mainly focused on Genetic Programming, a global search technique inspired by biological evolution [16][38][82][118]. Local search approaches, e.g., TS, have been investigated only in few preliminary studies [42][44]. In this paper we deepen the analysis of these metaheuristics from the point of view of their settings and the empirical assessment of their predictive capability. As for the setting, differently from previous works where it was adopted a time consuming trial-and-error process to set search-based approach parameters (e.g., number of moves and iterations for TS), in the present study we

employed an heuristics which relates those parameters to the problem size and assessed its effectiveness by comparing it with five configurations varying for the number of solutions and iterations exploited. Moreover, we employed as objective function the sum of squared residuals (SSR) (also named sum of squared errors of prediction (SSE)) since we compared the results achieved with GP, TS, and HC with Linear (StepWise) regression which exploits SSR to fit data.

Regarding the empirical assessment, let us observe that almost all previous studies employed only one dataset with a hold-out validation thus affecting both external and internal validity. In the present work, we assessed three search-based approaches, namely HC, TS, and GP, exploiting seven publicly available datasets (i.e., China, Desharnais, Finnish, Miyazaki, Kemerer, Maxweel, and Telecom) and performing a 3-fold cross validation. These datasets represent an interesting sample of industrial software projects containing both single- and cross-company data which vary for size, application domains, and project characteristics. Moreover, to assess the effectiveness of the three approaches we first compared them with respect to different baseline benchmarks (i.e., random search, mean and median of effort about past projects) since if they do not outperform the results achieved with simpler methods it cannot be transferred to industry [97]. Then, aiming to verify if HC, TS, and GP provide at least comparable results with respect to estimation techniques widely used in the literature and in industry, we considered as benchmark Manual StepWise Regression (MSWR).

The estimates obtained with the employed techniques were evaluated and compared by exploiting SSR and statistical tests on squared residuals [72].

The remainder of the chapter is organized as follows. In Section 3.1 we provide a description of the three applied search-based approaches, showing the employed setting. In Section 3.2 we describe the experimental method we exploited to assess and compare the prediction accuracy of HC, TS, and GP. The results of the empirical analysis are reported and discussed in Section 3.3, while Section 3.4 discusses the factors that can bias the validity of empirical studies.

3.1 Designing Search-based Approaches for building effort estimation models

Search-based methods are a set of algorithms that search for optimal or near optimal solutions to problems characterized by large space, using an objective function that indicates how a solution is suitable for the problem under investigation.

The idea of exploiting these methods to estimate software development effort is based on the observation that the effort estimation problem can be formulated as an optimization problem. As a matter of fact, among possible estimation models (equations), we have to identify the one that leads to the best predictions.

In order to have a better insight on search-based methods in our analysis we exploited three methods, HC, TS, and GP that have complementary characteristics. Indeed, HC and TS are local search-based methods, while GP follows a global search. This means that HC and TS are more exploitation oriented being designed to intensify the search in local regions, on the contrary GP is more exploration oriented allowing for a better diversification in the whole search space.

Some background on these three methods have been provided in Chapter 1, while in this section we present how we designed the HC, TS and GP for tying them to software development effort estimation.

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

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

where f_i represents the value of the *i-th* project feature and c_i is its coefficient, C represents a constant, while $op_i \in \{+, -, \cdot, \ln, \hat{}\}$ represents the *i-th* mathematical operator of the model. It is worth noting that the equations feasible for the effort estimation problem are those providing positive value for *Effort*.

The fitness function guides the search for the best estimation model. In the context of effort estimation such function should be able to determine whether a model leads to better predictions than another. In the literature several accuracy measures have been proposed to compare effort estimation models and each of them could be employed as objective function. In previous works [44][45], different designs have been experimented employing different accuracy measures revealing there was no significant difference in the results achieved with different objective functions except for Mean MRE [22] and Mean of EMRE [72] which should be avoided since they provided significantly worse predictions than other functions. Thus, in this work we employ SSR as objective function that is at the basis of MSWR, allowing a fair comparison with this benchmark. The same number of solutions and iterations characterizes all the SB approaches we employed. Let us observe that a suitable setting is usually obtained via a trial-and-error process for each new problem to be addressed. As a matter of fact, in previous work [42][44] the number of solutions and

iterations was empirically determined carrying out a validation process with different values for these parameters and selecting the one providing the best results. However, this practice is time consuming and it has to be repeated every time new data is used, thus limiting the adoption of such technique and in general of search-based approaches by practitioners. To overcome this limitation in the present paper we employed a heuristics originally suggested in [40] to set population size and generation number of a genetic algorithm employed for software clustering. The same heuristics has been successively adopted for setting GA in the context of effort estimation [19][61] and we employed it in this work also with HC and TS. In particular, given a project dataset containing V features, we set to $10V$ the number of solutions (i.e., neighbors for TS and HC), to V the Tabu List size, and stop the search after $1000V$ iterations or if the objective value of the best solution does not change in the last $100V$ iterations. Thus such heuristics allowed us to adapt the search process to the size of the problem under investigation.

Finally, let us observe that since search-based approaches do not give the same solution each time it is executed, we performed 30 runs and we retained as final results the average SSR values obtained in the 30 runs.

3.1.1 Setting of Hill Climbing and Tabu Search

As for the move employed in HC and TS to obtain a neighboring solution we applied the following steps to the current solution S:

- change each coefficient c_i of S with probability 1/2. The new coefficient c_i is calculated as follows: $c_i' = f(c_i , r)$ where $f \in \{+,-,*,/,\hat{},ln\}$ and r is randomly chosen in the range]0,1];
- change the constant factor C of S with probability 1/2, in the same way coefficients are changed;
- change each arithmetic operator op_i of S with probability 1/2.

As for the design of TS, we employed as Tabu List a short-term strategy to store the moves leading to the most recent already visited solutions and the following aspiration criteria: a tabu move is allowed only if it results in a solution with an objective function value better than the one of the best solution reached so far.

3.1.2 Setting of Genetic Programming

The initial population is generated by building random trees of fixed depth. As for the evolutionary process we employed two widely used selection operators, i.e., roulette wheel selector and tournament selector [80], whereas the crossover and mutation operators are specific for our solution encoding. In particular, we used the roulette wheel selector to choose the individuals for reproduction, while we employed the tournament selector to determine the individuals that are included in the next generation (i.e., survivals). The former assigns a roulette slice to each chromosome according to its fitness value. In this way, even if candidate solutions with a higher fitness have more chance to be selected, there is still a chance that they may be not. On the contrary, using the tournament selector only the best *n* solutions (usually *n* in [1, 10]) are copied straight into the next generation. Crossover and mutation operators were defined to preserve well-formed equations in all offspring. To this end, we used a single point crossover which randomly selects in each tree a node placed at the same depth and swaps the subtrees corresponding to the selected point. Since the two trees are cut at the same point, the trees resulting after the swapping have the same depth as compared to those of parent trees. Concerning the mutation, we employed an operator that selects a node of the tree and randomly changes the associated value. The mutation can affect internal node (i.e., operators) or leaves (i.e., coefficients) of the tree. In particular, when the mutation involves internal node, a new operator op_i' in {{+,-,*,^,ln} - op_i} is randomly generated and assigned to the node, while if the mutation involves a leaf a new coefficient c_i' in R is assigned to the node. It is worth noting that the employed mutation preserves the syntactic structure of the equation. Crossover and mutation rate were fixed to 0.5 and 0.1, respectively.

3.2 Empirical Study Design

In this section we present the design of the empirical study we carried out to assess the effectiveness of the proposed HC, TS, and GP for estimating software development effort. In particular the research questions of our study can be outlined as follows:

> **RQ1** Are there any differences in the accuracy of the predictions achieved with different settings?

RQ2 Is it possible to identify a suitable heuristics to configure the considered search-based approaches?

RQ3 Are there any differences in the accuracy of the predictions achieved using different search-based approaches?

RQ4 Do the considered search-based approaches provide significantly better prediction accuracy than those obtained by employing baseline benchmarks?

RQ5 Do the considered search-based approaches provide prediction accuracy at least comparable with those provided by MSWR?

The fact that an overlooked aspect of research on computational search algorithms lies in the selection and tuning of the algorithmic parameters motivated us to investigate RQ1 and RQ2. Let us observe that a suitable setting is usually obtained via a trial-and-error process for each new problem to be addressed. As a matter of fact, in previous work (see e.g., [16][38]) the number of moves and iterations was empirically determined carrying out a validation process with different values for these parameters and selecting the one providing the best results. However, this practice is time consuming and it has to be repeated every time new data is used, thus limiting the adoption of such technique and in general of search-based approaches by practitioners. To overcome this limitation in the present paper we employed a heuristics originally suggested in [40] to set population size and generation number of a genetic algorithm employed for software clustering. The same heuristics was successively adopted for setting genetic algorithms in the context of effort estimation [45][62][61] and in this work we extended it to work also with TS. In particular, given a project dataset containing V features, we set to $10V$ the number of iterations, to V the Tabu List size (in case of TS), and stop the search after $1000V$ iterations or if the objective value of the best solution does not change in the last $100V$ iterations (in case of GP and TS). Thus such heuristics allowed us to adapt the search process to the size of the problem under investigation. To assess the effectiveness of the proposed heuristics we compared it with respect to the use of five different configurations characterized by very small, small, medium, large, and very large values for solutions as detailed in Table 3. All configurations were allowed an identical budget of objective function evaluations (250,000), thereby ensuring that all require the same computational effort, though they may differ in parameter settings. On the other hand the "search budget" required by the heuristics is at most $10V$*$100V$ evaluations due to the two different stopping criteria employed. In particular we answered to the following questions:

RQ1a Are there any differences in the accuracy of the predictions achieved by HC configured with the settings VS, S, M, L, VL?

RQ1b Are there any differences in the accuracy of the predictions achieved by TS configured with the settings VS, S, M, L, VL?

RQ1c Are there any differences in the accuracy of the predictions achieved by GP configured with the settings VS, S, M, L, VL?

RQ2a Is the prediction accuracy obtained with the Heuristics based setting comparable with those achieved with the other settings for HC?

RQ2b Is the prediction accuracy obtained with the Heuristics based setting comparable with those achieved with the other settings for TS?

RQ2c Is the prediction accuracy obtained with the Heuristics based setting comparable with those achieved with the other settings for GP?

Table 3. Settings employed for HC, TS, and GP

Configuration	**Number of Solutions**	**Number of Iterations**	**Tabu List Size (applicable only for TS)**
Very Small	50	5000	5
Small	100	2500	10
Medium	200	1250	20
Large	500	500	50
Very Large	1000	250	100
Heuristic	10V	1000V || 100V ? BestFitness constant	V

RQ3 has been motivated by the fact that all previous work [16][38][82][118] exploited Genetic Programming (GP) to build effort estimation models. However, there exist different search-based methods that have complementary characteristics. In order to have a better insight on search-based methods in our analysis we compared three search-based methods, namely HC, TS, and GP. Indeed, HC and TS are local search-based methods, thus are more exploitation oriented, while GP follows a global search being more exploration oriented. In particular we answered to the following questions:

RQ3a Is the prediction accuracy provided by TS superior to the one provided by HC?

RQ3b Is the prediction accuracy provided by GP superior to the one provided by HC?

RQ3c Is the prediction accuracy provided by TS superior to the one provided by GP?

Once we assessed whether there are differences in using different search-based techniques, we compared their performance with respect to the ones of different baseline benchmarks (RQ3), since if they do not outperform the results achieved with these baseline methods they cannot be transferred to industry [97]. To this end we considered the following baseline techniques:

- **Random**: the same number of solutions investigated by the three search-based approaches was generated in a totally random fashion and the best one among them was selected according to the same criterion employed for HC, TS, and GP. This is a natural ``sanity check'' when using meta-heuristics search techniques.
- **Mean (Median) Effort**: the mean (median) of the past project efforts is used as predicted effort for a new project. These are popular and simple baseline benchmarks for effort estimation techniques.

In particular, to address RQ4 we answered to the following questions:

RQ4a Is the prediction accuracy provided by HC superior to the one provided by Random, Mean Effort, and Median Effort?

RQ4b Is the prediction accuracy provided by TS superior to the one provided by Random, Mean Effort, and Median Effort?

RQ4c Is the prediction accuracy provided by GP superior to the one provided by Random, Mean Effort, and Median Effort?

Once we verified the usefulness of the employed search-based techniques comparing them with baseline benchmarks, we assessed if they are also effective. Indeed, RQ5 aimed to verify if the search-based approaches that have been revealed superior to the baseline benchmarks provide prediction accuracy at least comparable with the one of the technique widely used in the literature and in industry, namely Manual StepWise Regression (MSWR). MSWR is a regression technique proposed by Mendes and Kitchenham [97] whereby an equation (i.e., the prediction model) is built and represents the relationship between independent (e.g., Team Experience, Function Points) and dependent variables (e.g., effort expressed in person/hours). This technique builds the prediction model by adding, at each stage, the independent variable with the highest association to the dependent variable, taking into account all variables currently in the model. It aims to find the set of independent variables (predictors) that best explain the variation in the dependent variable

(response). To apply MSWR we followed the guidelines provided in [76][97] including the verification of the assumptions underlying linear regression. In particular, to address RQ5 we answered to the following questions:

RQ5a Does TS provide prediction accuracy at least comparable with MSWR?

RQ5b Does GP provide prediction accuracy at least comparable with MSWR?

In the following we present the datasets, the validation method, and the evaluation criteria employed in our empirical analysis.

3.2.1 Dataset

To carry out the empirical study we exploited seven publicly available datasets included in the PROMISE repository [109], namely China, Desharnais, Finnish, Miyazaki, Kemerer, Maxwell, and Telecom. All these datasets were previously used to evaluate estimation methods (see e.g., [16][25][42][102][121][122]). Our choice was motivated by the aim to select an interesting data sample of industrial software projects representing a diversity of application domains and project characteristics. In particular, the employed datasets contain data collected from a single software company (i.e., Desharnais, Telecom, and Kemerer) or several companies (i.e., China, Finnish, Miyazaki, and Maxwell) geographically dislocated around the world (e.g., Canada, China, Finland), thus enabling us to assess the estimation technique herein employed in single- and cross-company contexts. The use of a cross-company dataset is particularly useful for companies that do not have their own data on past projects from which to obtain their estimates, or that have data on projects developed in different application domains and/or technologies.

The employed datasets include information about different features (the number varies ranging from 1 to 17). Indeed, as for the independent variables three datasets (i.e., China, Finnish, and Desharnais) contain Function Points (or their basic components) as size measure and different cost-drivers, such as manager and team experience, while one dataset (i.e., Miyazaki) contains only the components of Object Points as size measure; in all the datasets the dependent variable was the effort expressed in person-hours. As for the features, we excluded categorical variables (e.g., Language and YearEnd in Desharnais) and all the features that could not be available at the time the prediction would be made, such as the length of the code (LOC). This is important to avoid creating a false impression as to the efficacy of different prediction methods [122]. Also the number of

observations varies for the employed datasets (from 10 to 499). It is important to notice that in our analysis we excluded the observations that had missing values (i.e., four projects for Desharnais). The descriptive statistics of the employed variables for the seven datasets are shown in Table 3. A detailed description of each dataset is reported in Appendix A.

3.2.2 Validation Method and Evaluation Criteria

In order to verify whether or not a method gives useful estimations of the actual development effort a validation process is required. To this end, we performed a multiple-fold cross validation, partitioning the whole dataset into training sets, for model building, and test sets, for model evaluation. Indeed, when the accuracy of the model is computed using the same dataset employed to build the prediction model, the accuracy evaluation is considered optimistic [13][14][15]. Cross validation is widely used in the literature to validate effort estimation models when dealing with medium/small datasets (see, e.g. [14][15]). We applied a 3-fold cross validation obtaining for each dataset 3 randomly test sets and then for each test set we considered the remaining observations as training set to build the estimation model. To allow for replications of our study, the folds employed for each dataset are reported in Table 5.

To evaluate the obtained estimates we employed the sum of squared residuals (SSR) in order to employ the same strategy exploited by both search-based techniques and MSWR to fit the data (i.e., minimizing SSR). Let us recall that SSR is defined as follows:

$$SSR = \sum_{i=1}^{n} (actual_i - predicted_i)$$

where n is the number of observations, $actual_i$ and $predicted_i$ are the actual and the predicted effort for the observation i, respectively.

The analysis of SSR gives only an indication on which is the estimation method that globally gives best effort estimations. In order to establish if an estimation method provides better results than another it is necessary to test the statistical significance of the obtained results. For this reason we tested the statistical significance of the squared residuals achieved with the considered estimation methods [72][96][128]. Such an analysis aims at verifying that the estimations of one method are significantly better than the estimations provided by another method. Since (i) the squared residuals

for all the analyzed estimation methods were not normally distributed (as confirmed by the Shapiro test [110] for non-normality), and (ii) the data was naturally paired, we used the Wilcoxon test [20] setting the confidence limit at $\alpha = 0.05$ (i.e., if the p-value of the test is less than 0.05 we can reject the null hypothesis) and applying Bonferroni correction in cases it is needed.

Table 4. Descriptive statistics of the employed variables

Dataset	Variable	Min	Max	Mean	Std. Dev.
China	Input	0	9404	167.10	486.34
	Output	0	2455	113.60	221.27
	Inquiry	0	952	61.60	105.42
	File	0	2955	91.23	210.27
	Interface	0	1572	24.23	85.04
	Effort	26	54620	3921	6481
Desharnais	TeamExp	0	4	2.30	1.33
	ManagerExp	0	4	2.65	1.52
	Entities	7	386	121.54	86.11
	Transactions	9	661	162.94	146.08
	AdjustedFPs	73	1127	284.48	182.26
	Effort	546	2349	4903.95	4188.19
Finnish	HW	1	3	1.26	0.64
	AR	1	5	2.24	1.50
	FP	65	1814	763.58	510.83
	CO	2	10	6.26	2.73
	Effort	460	26670	7678.29	7135.28
Miyazaki	SCRN	0	281	33.69	47.24
	FORM	0	91	22.38	20.55
	FILE	2	370	34.81	53.56
	Effort	896	253760	13996	36601.56
Maxwell	SizeFP	48	3643	673.31	784.08
	Nlan	1	4	2.55	1.02
	T01	1	5	3.05	1.00
	T02	1	5	3.05	0.71
	T03	2	5	3.023	0.89
	T04	2	5	3.19	0.70
	T05	1	5	3.05	0.71
	T06	1	4	2.90	0.69
	T07	1	5	3.24	0.90
	T08	2	5	3.81	0.96
	T09	2	5	4.06	0.74
	T10	2	5	3.61	0.89
	T11	2	5	3.42	0.98
	T12	2	5	3.82	0.69
	T13	1	5	3.06	0.96
	T14	1	5	3.26	1.01
	T15	1	5	3.34	0.75
	Effort	583	63694	8223.2	10500
Telecom	Changes	3	377	138.06	119.95
	Files	3	284	110.33	91.33
	Effort	23.54	1,115.54	284.34	264.71
Kemerer	AdjFP	99.30	2306.80	999.14	589.59
	Effort	23.20	1,107.31	219.25	263.06

Table 5. The folds employed in our study

Dataset		Project Id
Desharnais	Fold 1 (25 observations)	11, 19, 24, 77, 01, 26, 78, 02, 05, 12, 22, 23, 35, 40, 58, 61, 68, 69, 29, 50, 13, 81, 49, 70, 65
	Fold 2 (26 observations)	10, 15, 30, 41, 42, 43, 21, 03, 47, 63, 56, 62, 74, 31, 52, 37, 57, 73, 76, 34, 27, 33, 72, 79, 54, 80
	Fold 3 (26 observations)	04, 08, 09, 14, 16, 17, 18, 25, 32, 36, 39, 45, 51, 53, 55, 59, 60, 67, 71, 06, 07, 20, 28, 46, 48, 64
Finnish	Fold 1 (13 observations)	03, 06, 37, 28, 35, 18, 01, 22, 14, 32, 36, 12, 20
	Fold 2 (12 observations)	05, 33, 15, 26, 13, 30, 02, 31, 34, 17, 21, 29
	Fold 3 (13 observations)	25, 08, 38, 10, 7, 27, 11, 23, 16, 19, 24, 09, 04
Miyazaki	Fold 1 (16 observations)	C1, D1, D2, C2, H1, K2, L1, I2, J1, J3, R2, D3, A2, P4, Q1, B1
	Fold 2 (16 observations)	G3, I1, K1, R1, B2, N1, C3, K3, L3,J2, H2, L2, F1, N2, Q2, B3
	Fold 3 (16 observations)	N3, O1, P3, E1, G1, N4, O2, P1, P2, M1, A1, A3, E2, S1, G2,T1
Maxwell	Fold 1 (21 observations)	1,6,8,12,14,15,21,22,24,26,29,30,33,34,39,44,47,52,57,58,60
	Fold 2 (20 observations)	2,5,7,16,18,20,23,25,28,32,35,36,40,42,46,48,50,51,53,55
	Fold 3 (21 observations)	3,4,9,10,11,13,17,19,27,31,37,38,41,43,45,49,54,56,59,61,62
Telecom	Fold 1 (6 observations)	1,4,5,9,10,18
	Fold 2 (6 observations)	2,5,8,11,16,17
	Fold 3 (6 observations)	3,7,12,13,14,15
China	Fold 1 (166 observations)	1,2,4,9,10,11,13,16,18,19,20,23,26,29,30,33,34,36,39,40,42,52,53,54, 56,57,58,59,60,63,66,75,78,79,87,91,93,95,96,98,105,113,114,115, 117,128,129,130,134,135,140,142,150,151,157,159,164,165,169, 171,177,179,180,185,189,192,196,208,210,211,212,213,218,220, 221,223,224,225,232,236,238,244,251,252,253,254,256,258,260,264, 265,270,272,276,277,278,279,280,281,283,284,289,293,295,297, 299,300,306,308,309,316,318,321,322,332,336,341,346,349,352, 357,358,365,368,369,370,372,373,375,376,383,384,385,389,396, 403,410,411,414,416,423,424,429,436,438,452,455,456,457,458,461, 462,466,469,471,474,477,480,483,485,487,489,493,495,497,499
	Fold 2 (167 observations)	3,5,6,7,8,12,14,15,17,25,31,32,35,38,41,44,45,49,55,62,67,68,71,73, 74,84,85,88,90,92,97,99,102,103,106,108,109,110,119,120,122,127, 133,137,138,144,145,147,148,149,152,156,160,162,163,166,168,170, 172,174,178,181,182,184,188,190,191,193,199,200,201,202,203, 204,206,209,215,216,219,222,229,233,235,240,241,242,245,246, 247,248,249,250,255,259,266,268,275,282,286,288,298,301,303, 304,307,311,312,314,315,324,326,328,333,334,335,337,339,340, 344,345,347,351,354,355,364,377,379,380,382,387,391,395,400,401, 408,412,413,415,418,419,422,426,427,433,437,439,440,442,443,445,446,447, 454,459,460,465,467,472,473,475,476,478,484,488,490,494,496
	Fold 3 (166 observations)	21,22,24,27,28,37,43,46,47,48,50,51,61,64,65,69,70,72,76,77,80,81, 82,83,86,89,94,100,101,104,107,111,112,116,118,121,123,124,125,126, 131,132,136,139,141,143,146,153,154, 155,158,161,167,173,175,176, 183,186,187,194,195,197,198,205,207,214,217,226,227,228,230,231, 234,237,239,243,257,261,262,263,267,269,271,273,274,285,287,290,291, 292,294,296,302,305,310,313,317,319,320,323,325,327,329,330,331, 338,342,343,348,350,353,356,359,360,361,362,363,366,367,371,374, 378,381,386,388,390,392,393,394,397,398,399,402,404,405,406,407,409, 417,420,421,425,428,430,431,432,434,435,441,444,448,449,450,451,453, 463,464,468,470,479,481,482,486,491,492,498
Kemerer	Fold 1 (5 observations)	5,7,9,11,14
	Fold 2 (5 observations)	4,6,8,12,13
	Fold 3 (5 observations)	1,2,3,10,15

3.3 Analysis and Interpretation of the Results

3.3.1 RQ1 and RQ2

Table 6, Table 7 and Table 8 report the mean of the 30 SSR values obtained with the 30 executions of HC, TS, and GP employing each of the considered setting (i.e., VS, S, M, L, VL, Heuristic), respectively. We can observe that there are several differences in the prediction accuracy obtained using HC with the settings VS, S, M, L, VL and in general the smallest configuration provides better results when used with HC, while we observed less variability in the results when using TS and GP and thus there is not a clear winner among the five configurations. As for the comparison between the considered heuristics and the five settings, we can observe that the results obtained by HC with Heuristic are quite worse than those obtained with the other settings, while the results obtained by TS and GP with Heuristic are better or comparable with respect to the ones obtained with the other settings.

We also statistically compared the squared residuals obtained with the Heuristic based setting with those achieved with the other settings by applying the Wilcoxon test (see Tables 9, 10 and 11). For each comparison *X vs Y* (where *X* and Y can be VS, S, M, L, VL, or Heuristic) the table reports the p-value obtained with the Wilcoxon Test. It is worth noting that we applied a Bonferroni correction, i.e., 0.05/5=0.01, since we performed five tests (one for each of the other configurations, i.e., VS, S, M, L, or VL) to address research questions RQ1a, RQ1b, and RQ1c. and RQ2a, RQ2b, and RQ2c. Thus, in the table a p-value less than 0.01 means that there is a statistical significant difference between the SSR values achieved with the setting X and those obtained by employing the *Y* based setting (these cases are highlighted by bold font). We can observe that the results in terms of SSR are confirmed by the Wilcoxon Test results. Indeed, regarding the results obtained using HC with VS, S, M, L and VL we found that there is a statistical difference in 26 cases, while the difference among the predictions obtained with these configurations is less frequent when they are used with TS and GP (11 and 15 cases, respectively). As for the comparison between the employed heuristics and the five settings, we found that the square residuals obtained setting HC with Heuristic were in 5 cases significantly worse than the ones provided by another configuration (i.e., VS and S on China, VS on Finnish, VS on Maxwell, and VL on Miyazaki) and in three cases better than the other configurations (i.e., S, M and L on the Finnish dataset). This could be due to the fact that the HC

setting based on Heuristic and the settings based on the other 5 configurations just differ for the considered number of solutions and, in particular, the number of solutions considered by Heuristic is less or equals than those considered by the best configurations found except for the Maxwell dataset. Thus, we can partially positively answer to RQ2a (i.e., the predictions obtained with the heuristics based setting are comparable with those achieved with the other settings, for HC). On the other hand we found that the square residuals obtained using TS and GP with Heuristic were in 10 cases significant better than those obtained using other configurations (i.e., TS with Heuristics provided significantly better square residuals than using it with VS, S, M settings on the dataset China, and with S and VL on the Miyazaki dataset; GP with Heuristics provided significantly better square residuals than using it with all the other five configurations on the China datasets), while in the other cases no significant difference was found. Thus we can state that the considered heuristic is suitable to set TS and GP since it has allowed us to obtain comparable or superior prediction accuracy with respect to the other configurations. Moreover, if we look at the execution time obtained in 30 executions by TS and GP with all the considered configurations (see the histograms in Figure 1 and Figure 2) we can observe the use of TS and GP with the heuristics is always much faster than using them with the other settings. This is due to the fact that using the heuristics both TS and GP halted always before the maximum number of iterations (i.e., they stopped when the objective value of the best solution does not change in the last $100V$ iterations), thus performing less than the 250,000 evaluations required by the other settings on all the datasets. This allowed us to save time and computational resources without affecting the accuracy of the estimation models built with TS and GP, so we can state that the use of the heuristics has been revealed a cost-effective way to set these techniques on the considered datasets. Thus, we can positively answer to research questions RQ2b and RQ2c (i.e., the predictions obtained with the heuristics are comparable with those achieved with the other settings, for TS and GP).

Based on the above considerations we can conclude that there are differences in using the considered search-based approaches with different settings and the investigated heuristics is suitable for setting TS and GP and less useful to set HC on the considered datasets.

Table 6. Results in terms of SSR obtained employing HC with different settings

Dataset	VS	S	M	L	VL	Heuristic
China	**3.79E+10**	3.87E+10	4.28E+10	4.28E+10	4.26E+10	4.30E+10
Desharnais	**1.08E+10**	1.15E+10	1.35E+10	1.36E+10	1.19E+10	1.29E+10
Finnish	**5.50E+09**	1.07E+10	1.48E+10	1.48E+10	1.36E+10	8.79E+09
Kemerer	5.04E+06	4.29E+06	5.16E+06	5.16E+06	**4.41E+06**	4.82E+06
Maxwell	**1.95E+10**	2.44E+10	2.47E+10	2.47E+10	2.11E+10	2.79E+10
Miyazaki	4.66E+10	4.63E+10	**4.52E+10**	**4.52E+10**	4.58E+10	4.71E+10
Telecom	6.22E+05	6.27E+05	**6.18E+05**	**6.18E+05**	6.44E+05	6.51E+05

Table 7. Results in terms of SSR obtained employing TS with different settings

Dataset	VS	S	M	L	VL	Heuristic
China	1.16E+10	1.16E+10	1.18E+10	1.05E+10	1.05E+10	**1.04E+10**
Desharnais	**5.29E+08**	5.28E+08	5.30E+08	5.32E+08	5.35E+08	5.41E+08
Finnish	**7.42E+08**	7.52E+08	7.45E+08	7.49E+08	7.52E+08	7.48E+08
Kemerer	**1.18E+06**	**1.18E+06**	**1.18E+06**	**1.18E+06**	**1.18E+06**	1.19E+06
Maxwell	**1.80E+09**	**1.80E+09**	1.81E+09	1.81E+09	1.81E+09	1.81E+09
Miyazaki	**3.62E+10**	3.76E+10	3.49E+10	3.61E+10	3.82E+10	3.63E+10
Telecom	**4.27E+05**	4.27E+05	4.38E+05	4.62E+05	4.70E+05	4.56E+05

Table 8. Results in terms of SSR obtained employing GP with different settings

Dataset	VS	S	M	L	VL	Heuristic
China	1.50E+10	1.46E+10	1.46E+10	1.46E+10	1.49E+10	**1.16E+10**
Desharnais	9.52E+08	9.50E+08	9.55E+08	9.58E+08	9.59E+08	**9.51E+08**
Finnish	1.49E+09	**1.48E+09**	1.49E+09	**1.48E+09**	1.49E+09	1.53E+09
Kemerer	9.48E+05	9.46E+05	9.41E+05	9.44E+05	**9.29E+05**	9.33E+05
Maxwell	1.91E+09	1.93E+09	**1.87E+09**	1.88E+09	**1.87E+09**	2.02E+09
Miyazaki	3.51E+10	3.50E+10	3.91E+10	**3.49E+10**	3.57E+10	3.56E+10
Telecom	8.67E+05	8.65E+05	8.60E+05	8.56E+05	8.45E+05	**8.37E+05**

Table 9. Results of the Wilcoxon tests comparing different settings employed for HC

China	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
	S		-	**<0.0001**	**<0.0001**	0.110	**<0.0001**
	M			-	1	0.143	0.648
	L				-	0.143	0.648
	VL					-	0.12

Desharnais	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.462	0.239	0.042	**0.001**	0.749
	S		-	0.109	0.013	0.327	0.549
	M			-	0.507	0.042	0.113
	L				-	0.673	0.012
	VL					-	0.765

Finnish	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**<0.0001**	**<0.0001**	**<0.0001**	**0.001**	**<0.0001**
	S		-	**0.000**	**0.000**	0.451	**0.000**
	M			-	1	0.954	**<0.0001**
	L				-	0.954	**<0.0001**
	VL					-	0.27

Kemerer	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**0.001**	0.932	0.932	**0.001**	0.551
	S		-	0.029	0.029	0.514	0.379
	M			-	1	**0.001**	0.798
	L				-	**0.001**	0.798
	VL					-	0.32

Maxwell	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**<0.0001**	**<0.0001**	**<0.0001**	0.669	**0.000**
	S		-	0.29	0.29	0.017	0.111
	M			-	1	**<0.0001**	0.547
	L				-	**<0.0001**	0.547
	VL					-	0.054

Myiazaki	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.898	0.09	0.09	**0.002**	0.619
	S		-	**<0.0001**	**<0.0001**	**0.000**	0.174
	M			-	1	0.232	0.648
	L				-	0.232	0.648
	VL						**<0.0001**

Telecom	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.571	0.571	0.571	0.459	0.663
	S		-	0.728	0.728	0.486	0.514
	M			-	1	0.486	0.408
	L				-	0.486	0.408
	VL					-	0.408

Table 10. Results of the Wilcoxon tests comparing different settings employed for TS

China	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**0.000**	**0.002**	**<0.0001**	**<0.0001**	**<0.0001**
	S		-	0.684	**<0.0001**	**<0.0001**	**<0.0001**
	M			-	**<0.0001**	**<0.0001**	**<0.0001**
	L				-	**0.002**	0.704
	VL					-	0.034

Desharnais	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.58	0.35	0.405	0.405	0.417
	S		-	0.078	0.233	0.337	0.48
	M		0.078	-	0.437	0.863	0.863
	L		0.233	0.437	-	0.206	0.215
	VL		0.337	0.863	0.206	-	0.688

Finnish	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS		0.728	0.417	0.695	0.155	0.505
	S		-	0.919	0.85	0.376	0.931
	M			-	0.761	0.885	0.817
	L				-	**0.002**	0.739
	VL					-	0.477

Kemerer	**<>**	**VS**	**S**	**M**	**L**	**VL**	**TS**
	VS	-	0.629	0.689	0.977	0.065	0.065
	S		-	0.67	0.551	0.057	0.065
	M			-	0.551	0.05	0.065
	L				-	0.044	0.065
	VL					-	0.065

Maxwell	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.742	0.242	0.806	0.25	0.866
	S		-	0.614	0.806	0.57	0.994
	M			-	0.551	0.994	0.966
	L				-	0.043	0.689
	VL					-	0.785

Myiazaki	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.014	0.118	0.133	**0.007**	0.125
	S		-	0.031	0.010	0.012	**0.009**
	M			-	0.626	0.015	0.177
	L				-	**0.003**	0.191
	VL					-	**0.006**

Telecom	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.666	0.556	0.338	0.486	0.224
	S		-	0.021	0.127	0.258	0.224
	M			-	0.384	0.338	0.255
	L				-	0.408	0.965
	VL					-	0.896

Table 11. Results of the Wilcoxon tests comparing different settings employed for GP

China	**<>**	VS	S	M	L	VL	Heuristic
	VS	-	**0.000**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**
	S		-	**0.003**	**<0.0001**	**<0.0001**	**<0.0001**
	M			-	**0.001**	**0.005**	**<0.0001**
	L				-	0.047	**<0.0001**
	VL					-	**<0.0001**

Desharnais	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.431	0.601	0.875	0.815	0.44
	S		-	0.235	0.157	0.419	0.264
	M			-	0.594	0.651	0.369
	L				-	0.546	0.35
	VL					-	0.685

Finnish	**<>**	VS	S	M	L	VL	Heuristic
	VS	-	0.164	0.828	0.331	0.965	0.417
	S		-	0.114	0.523	0.202	0.523
	M		0.114	-	0.27	0.218	0.392
	L		0.523	0.27	-	0.592	0.392
	VL		0.202	0.218	0.592	-	0.369

Kemerer	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	0.262	0.164	0.222	0.164	0.514
	S		-	0.164	0.32	0.182	0.551
	M			-	0.132	0.182	0.755
	L				-	0.164	0.629
	VL					-	0.244

Maxwell	**<>**	VS	S	M	L	VL	Heuristic
	VS	-	0.833	0.927	0.994	0.123	0.25
	S		-	0.151	0.323	0.132	0.066
	M			-	0.872	0.161	0.051
	L				-	0.105	0.095
	VL					-	0.044

Myiazaki	**<>**	**VS**	**S**	**M**	**L**	**VL**	**Heuristic**
	VS	-	**0.003**	0.058	**0.001**	**0.004**	0.439
	S		-	0.663	**0.001**	**0.007**	0.98
	M			-	0.032	0.015	0.685
	L				-	**0.009**	0.576
	VL					-	0.253

Telecom	**<>**	VS	S	M	L	VL	Heuristic
	VS	-	0.571	0.632	0.459	0.542	1
	S		-	0.965	0.663	0.571	0.965
	M			-	0.76	0.571	0.728
	L				-	0.317	0.663
	VL					-	0.24

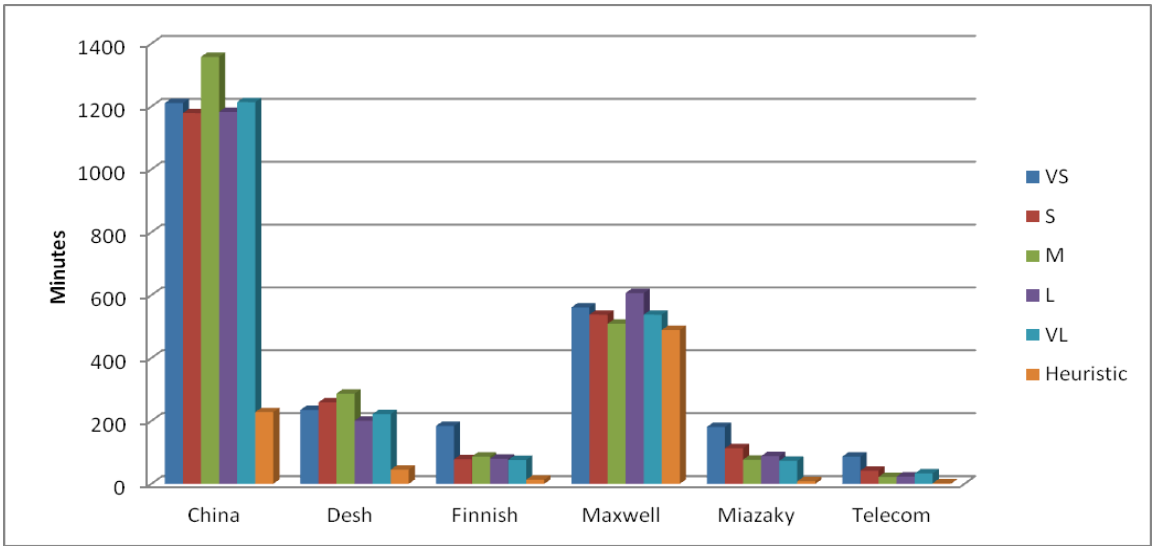

Figure 1. Execution time achieved by TS using different configurations (30 executions)

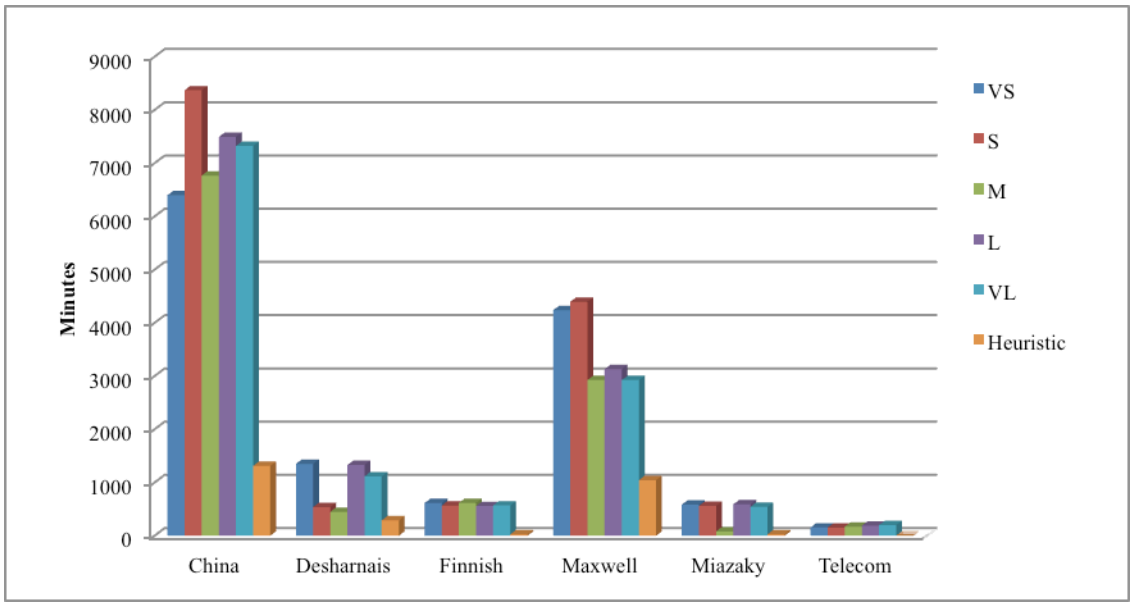

Figure 2. Execution time achieved by GP using different configurations (30 executions)

3.3.2 RQ3

Regarding the comparison among the three considered search-based approaches, the results reported in Table 12 reveal that both TS and GP provided better results in terms of SSR than HC on all the employed datasets. We also statistically compared the squared residuals obtained with the three search-based approaches by applying the Wilcoxon test, whose results are reported in Table 13. The

p-values reported in the table suggest that the difference among the achieved SSR values is statistically significant in the case of TS versus HC for all the datasets and in case of GP versus HC for all the datasets except for the Miyazaki and Telecom datasets. As for the comparison between TS and GP, we can observe that TS provided better results in terms of SSR than GP on all the employed datasets except for Kemerer and Miyazaki datasets, however this difference is significant only on the China dataset in favor of TS and on Kemerer and Miyazaki datasets in favor of GP (see Table 13). Thus, we can positively answer research question RQ3a (i.e., the accuracy provided by TS is superior to the one provided by HC) and partially positively answer RQ3b (i.e., the accuracy provided by GP is superior to the one provided by HC) and RQ3c (i.e., the accuracy provided by TS is superior to the one provided by GP).

Table 12. Results in terms of SSR obtained employing HC, TS, and GP using SSR as objective function

Dataset	**HC**	**TS**	**GP**
China	3.79E+10	**1.04E+10**	1.16E+10
Desharnais	1.29E+10	**5.41E+08**	9.51E+08
Finnish	5.5E+09	**7.48E+08**	1.53E+09
Kemerer	4.82E+06	1.19E+06	**9.33E+05**
Maxwell	1.95E+10	**1.81E+09**	2.02E+09
Miyazaki	4.58E+10	3.63E+10	**3.56E+10**
Telecom	6.51E+05	**4.56E+05**	8.37E+05

Table 13. Results of the Wilcoxon test comparing HC, TS, and GP using SSR as objective function

<	**TS vs HC**	**GP vs HC**	**TS vs GP**
China	**<0.001**	**<0.001**	**1**
Desharnais	**<0.001**	**<0.001**	0.41
Finnish	**<0.001**	**<0.001**	0.123
Kemerer	**<0.001**	**<0.001**	0.466
Maxwell	**<0.001**	**<0.001**	0.177
Miyazaki	**<0.001**	0.169	**0.027**
Telecom	**<0.001**	0.129	0.257

3.3.3 RQ4

Table 14 reports on the results achieved in terms of SSR for each employed baseline benchmarks and the mean of the 30 SSR values obtained with the 30 executions of HC, TS, and GP. Let us recall that the technique providing the less SSR value is considered better than the others. The analysis of these results suggests that the estimations obtained with TS and GP are better than those achieved by using Random Search, Mean Effort, and Median Effort on the employed datasets in terms of SSR, while HC provided better results only on 2 of 7 datasets (i.e., Miyazaki and Telecom) and worse on the other 5 (i.e., China, Desharnais, Finnish, Kemerer, and Maxwell). This is likely due to the fact that HC remained trapped in local optima, while TS and GP, being more sophisticated approaches, are able to escape from these points (see Chapter 1).

Table 14. Results in terms of SSR obtained employing HC, TS, and GP using SSR as objective function and the employed baseline benchmarks

Dataset	HC	TS	GP	Random	Mean	Median
China	3.79E+10	**1.04E+10**	1.16E+10	2.31E+10	2.11E+10	2.31E+10
Desharnais	1.29E+10	**5.41E+08**	9.51E+08	1.48E+09	1.33E+09	1.46E+09
Finnish	5.5E+09	**7.48E+08**	1.53E+09	3.23E+09	2.73E+09	2.45E+09
Kemerer	4.82E+06	1.19E+06	**9.33E+05**	1.46E+06	1.08E+06	1.09E+06
Maxwell	1.95E+10	**1.81E+09**	2.02E+09	6.87E+09	6.75E+09	7.31E+09
Miyazaki	4.58E+10	3.63E+10	**3.56E+10**	5.01E+10	7.49E+10	6.63E+10
Telecom	6.51E+05	**4.56E+05**	8.37E+05	1.79E+06	1.51E+06	1.70E+06

Table 15 reports the p-values obtained with the Wilcoxon Test to verify whether the SSR achieved by HC, TS, and GP were significantly less than those of the baseline benchmarks, for each dataset. It is worth noting that we applied a Bonferroni correction, i.e., 0.05/3=0.016, since we performed three tests (one for each baseline benchmark) to address research questions RQ4a, RQ4b, and RQ4c. Thus, in the table a p-value less than 0.016 means that the SSR values achieved with HC, (TS or GP) are significantly less than those obtained by the considered baseline benchmark. These cases are highlighted by bold font. We can observe that the results reported in Table 15 confirm those achieved in terms of SSR. Indeed, There were no difference between HC and the baseline benchmarks. On the contrary, TS and GP provided significant better squared residuals than those

achieved with the baseline benchmarks for all datasets except for Kemerer where no difference was found probably due the fact that this dataset is very small - both in terms of number of features (i.e., 1) and observations (i.e., 15) - compared to the other ones. As a matter of fact many work pointed out that in case of a small number of observations the Wilcoxon Test might be not powerful enough to confirm a statistical difference at a 0.05 significance level, even when the data seem to suggest such a difference [4]. Thus, from the above results we can conclude that we can positively answer the third research question for TS and GP (i.e., the accuracy provided by TS (and GP) is superior to the one provided by Random, Mean and Median Effort).

Table 15. Results of Wilcoxon Test comparing HC, TS, and GP using SSR as objective function and the employed baseline benchmarks

	HC vs			**GP vs**			**TS vs**		
<	**Random**	**Mean**	**Median**	**Random**	**Mean**	**Median**	**Random**	**Mean**	**Median**
China	0.839	**0.01**	0.999	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**<0.0001**	**0.005**
Desharnais	0.999	0.999	0.999	**0.008**	**0.003**	**0.006**	**0.003**	**<0.0001**	**0.012**
Finnish	0.994	0.924	0.97	**0.019**	**0.001**	**0.003**	**0.003**	**<0.0001**	**0.001**
Kemerer	0.999	0.999	0.999	0.295	0.205	0.489	**0.001**	0.556	0.623
Maxwell	0.995	0.999	0.999	**<0.0001**	**0.006**	**0.024**	**<0.0001**	**0.002**	**0.003**
Miyazaki	**<0.0001**	**<0.0001**	**0.011**	**0.001**	**<0.0001**	**0.002**	**<0.0001**	**<0.0001**	**<0.0001**
Telecom	**0.025**	**0.045**	**0.015**	**0.025**	**0.041**	**0.016**	**0.002**	**0.008**	**0.004**

3.3.4 RQ5

Table 16 report results in terms of SSR obtained by applying TS and GP (using SSR as objective function) and MSWR. We can observe that GP and TS provided better SSR values than MSWR on 3 out 7 (i.e., China, Finnish and Maxwell) and on 5 out 7 datasets (i.e., China, Desharnais, Finnish, Maxwell and Telecom), respectively. The results of the Wilcoxon test reported in Table 16 revealed that there was significant difference between GP and MSWR in three cases. In particular, we found that both GP and TS provided significant better results than MSWR on the dataset Miyazaki, while GP provided worse results than MSWR on the dataset Kemerer. This result is likely due to the fact that the Kemerer dataset is composed by only one feature therefore the linear model provided by MSWR is sufficient to explain the relation with the effort. Thus, we can positively answer the

research questions RQ5a and RQ5b (i.e., TS and GP provide at least comparable results with MSWR, respectively).

Table 16. Results in terms of SSR obtained employing TS and GP using SSR as objective function and MSWR

Dataset	TS	GP	MSWR
China	**1.04E+10**	1.16E+10	1.48E+10
Desharnais	**5.41E+08**	9.51E+08	8.55E+08
Finnish	**7.48E+08**	1.53E+09	1.55E+09
Kemerer	1.19E+06	9.33E+05	**7.51E+05**
Maxwell	**1.81E+09**	2.02E+09	2.63E+09
Miyazaki	3.63E+10	3.56E+10	**2.34E+10**
Telecom	**4.56E+05**	8.37E+05	6.48E+05

Table 17. Results of Wilcoxon Test comparing HC, TS, and GP using SSR as objective function and MSWR

< >	TS vs MSWR	GP vs MSWR
China	0.005	0.410
Desharnais	0.992	0.394
Finnish	0.400	0.739
Kemerer	0.132	**0.016**
Maxwell	0.217	0.274
Miyazaki	**<0.0001**	**0.003**
Telecom	0.728	0.338

3.4 Validity Evaluation

It is widely recognized that several factors can bias the validity of empirical studies. In this section we discuss the validity of the empirical study based on four types of threats, namely *construct*, *internal*, *conclusion*, and *external* validity. As highlighted by Kitchenham *et al.* [77], to satisfy construct validity a study has "to establish correct operational measures for the concepts being studied"'. This means that the study should represent to what extent the predictor and response variables precisely measure the concepts they claim to measure [98]. Thus, the choice of the features and how to collect them represents the crucial aspects. We evaluated the employed estimation

methods on seven publicly available datasets included in the PROMISE repository [109] previously used in many other empirical studies carried out to evaluate effort estimation methods, e.g., [16][25][45][122]. As for internal validity, biases can be introduced by the intrinsic randomness of the search-based techniques. We mitigate such a threat by executing HC, TS, and GP 30 times and using average results. Concerning the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Moreover, we used small/medium/large size datasets to mitigate the threats related to the number of observations composing the dataset. Finally, we tried to mitigate threats to the external validity employing both single- and cross-company datasets containing data about of industrial software projects that differ for size, application domains, and project characteristics.

CHAPTER 4: How the Objective Function Choice Affects the Effort Estimation Accuracy of Search-Based Approaches

As discussed in Chapter 2 the investigations carried out so far on the use of search-based approaches for effort estimation have focused on the use of Genetic Programming (GP) and have provided promising results [16][38][39][82][118]. Nevertheless, the design of these techniques deserves to be further explored and empirically analyzed also employing the more recent recommendations suggested in the effort estimation context [77][72] and in the search-based software engineering [4]. In particular, a crucial design choice is the definition of the objective function that indicates how a solution is suitable for the problem under investigation driving the search towards optimal solutions.

For the effort estimation problem the fitness function should be able to assess the accuracy of estimation models. It is worth noting that several different accuracy measures have been proposed for assessing the effectiveness/accuracy of effort prediction models. Among them the Mean Magnitude of Relative Error (MMRE) and the Prediction at level 25 (Pred(25)) represent the most widely used measures [22]. Each measure focuses the attention on a specific aspect, as a matter of fact "Pred(25) measures how well an effort model performs, while MMRE measures poor performance" [100]. Thus, the choice of the criterion for assessing predictions and establishing the best model can be a managerial issue: a project manager could prefer to use MMRE as the criterion for judging the quality of a model, while another might prefer to use another criterion, just for example Pred(25). From this point of view, search-based methods represent an opportunity since they allow a project manager to identify his/her preferred accuracy measure and explicitly use it as fitness function so that the search for the model is driven by such a criterion. Indeed, according to Harman and Clark view point, each measure that has been proposed as a means of evaluating some properties of interest can be used as fitness function [53]. It is worth noting that this is not possible for several other estimation techniques, such as Ordinary Least Squares Regression (OLSR), that have an embedded criterion (e.g., OLSR minimizes the sum of squared residuals). In the literature the studies that have reported on the use of GP for software development effort estimation were based on the use of MMRE [16][82] or Mean Square Error (MSE) [38] [118] as fitness function. Thus, we have carried out an empirical analysis to investigate how the use of different measures as

fitness function affects the overall accuracy of the estimation models built by GP. To do this, we evaluated the overall estimation accuracy by using the tools suggested in [72] to allow for a more reliable accuracy evaluation and a better comparison among different empirical analysis. These include the joint use of some different evaluation measures (e.g., MMRE, Pred(25), MdMRE), together with statistical tests [98][128].

The analysis has been carried out experimenting different objective functions based on some measures (and some combinations of these measures) proposed in the literature to evaluate the accuracy of the estimates. Preliminary empirical results based on the use of GP on the Desharnais dataset [33] were provided in [45]. The present chapter is an extension of [45] since we report and discuss the results obtained using also another technique (i.e., TS) and other publicly available datasets, namely Finnish [121], Miyazaki [102], Maxwell [87], Telecom [122], China [109], and Kemerer [68]. Moreover, we have analysed also the use of other fitness functions (i.e., SSR, a combination of MMRE and MEMRE, MEMRE and Pred(25) and a combination of MdEMRE and Pred(25)) not employed in [45].

The rest of the chapter is organized as follows. Section 4.1 describes the employed experimental method while the results are reported in Section 4.2 and empirical study validity is discussed in Section 4.3.

4.1 Empirical Study Planning

This section presents the design of the empirical study we carried out to analyze the impact of different objective functions on the accuracy of the estimation models built with GP and TS. In particular, in our analysis we defined the following research question:

- **RQ6a**: Does the employed objective function impact on the accuracy of the estimation models built with GP.
- **RQ6b:** Does the employed objective function impact on the accuracy of the estimation models built with TS.

To address them we employed the same techniques and datasets described in Section 3.1 and 3.2 respectively. Moreover, we experimented several objective functions that are described in Section

4.1.3. In the following we summarize the GP and TS settings, the validation method, and the evaluation criteria employed in the empirical analysis.

4.1.1 Setting of Genetic Programming and Tabu Search

To address research question RQ6a and RQ6b we experimented six of the accuracy measures described in Chapter 2 (i.e., SSR, MMRE, MdMRE, MEMRE, MdEMRE, Pred(25)) as objective function and analyzed the impact on the estimation accuracy of the models built with GP and TS. Moreover, the observation that different accuracy measures take into account different aspects of predictions accuracy [72][100] suggested us to investigate also the effectiveness of some combinations of those accuracy measures. In particular, we also experimented with Avg(MMRE, MEMRE), Pred(25)/MMRE, Pred(25)/MdMRE , Pred(25)/MEMRE, and Pred(25)/MdEMRE as objective functions. Table 18 summarizes the employed objective functions.

Table 18. The experimented objective functions

Employed Summary Measures	**Objective function**
Sum of Squared Residuals (SSR)	min SSR
Mean of Relative Magnitude (MMRE)	min MMRE
Median of Magnitude Relative Error (MdMRE)	min MdMRE
Prediction at Level 25 (Pred(25))	max Pred(25)/MMRE
Mean of Magnitude of Relative Error relative to Estimate (MEMRE)	min MdEMRE
Median of Magnitude of Relative Error relative to the Estimate (MdEMRE)	min MdEMRE
MMRE and MEMRE	min Avg(MMRE, MEMRE)
MMRE and Pred(25)	max Pred(25)/MdMRE
MdMRE and Pred(25)	max Pred(25)/MdMRE
MEMRE and Pred(25)	max Pred(25)/MEMRE
MdEMRE and Pred(25)	max Pred(25)/MdEMRE

The GP and TS parameters were set using the heuristics presented in the previous chapter; the employed settings for each dataset are summarized in Table 19 and Table 20. Since these techniques do not give the same solution each time they are executed, we performed 30 runs and presented the average results obtained in 30 runs on the test sets.

Table 19. The employed GP settings

Dataset	Population Size	Generation Number	Crossover Rate	Mutation Rate
China	50	<=5000	0.5	0.1
Desharnais	50	<=5000	0.5	0.1
Finnish	40	<=4000	0.5	0.1
Kemerer	10	<=1000	0.5	0.1
Miyazaki	30	<=3000	0.5	0.1
Maxwell	170	<=17000	0.5	0.1
Telecom	20	<=2000	0.5	0.1

Table 20. The employed TS setting

Dataset	Number of Moves	Number of Iterations	Tabu List Size
China	50	<=5000	5
Desharnais	50	<=5000	7
Finnish	40	<=4000	4
Kemerer	10	<=1000	1
Miyazaki	30	<=3000	3
Maxwell	170	<=17000	17
Telecom	20	<=2000	2

4.1.2 Validation Method and Evaluation Criteria

In order to verify whether or not a method gives useful estimations of the actual development effort we employed the same validation method described in Section 3.2.2 thus using the fold reported in

Table 5. Concerning the evaluation of the estimates obtained with the analyzed estimation methods, we used several summary measures, namely SSR, MMRE, MdMRE, Pred(25), MEMRE and MdEMRE [22][72]. Moreover, to establish if one of the prediction methods provides significant better estimates than the others, we tested the statistical significance of squared residuals achieved with the built models [72][98][128]. Since (i) the squared residuals for all the analysed estimation methods were not normally distributed (as confirmed by the Shapiro test [110] for non-normality), and (ii) the data were naturally paired, we used the Wilcoxon test [20] setting the confidence limit at $\alpha = 0.05$ and applying Bonferroni correction in cases it is needed.

4.2 Analysis and Interpretation of the Results

Table 21 and Table 22 report on the results obtained on the test sets in terms of summary measures related to the accuracy achieved by the models constructed by GP and TS, respectively. First of all we analyzed whether the use of a specific criterion allowed us to effectively derive models with the best value for the selected criterion. Moreover, we analyzed the impact on the overall estimation accuracy based on some different summary measures (e.g., MMRE, MdMRE, Pred(25)) as well as on the use of statistical tests on squared residuals.

We can observe that on almost all dataset all the objective functions provided the best value for the accuracy statistics for which they were specifically designed (e.g., TS optimizing SSR provided the best SSR values on all the employed datasets). However, if we consider the overall accuracy of the estimation models (i.e., considering also the other accuracy measures) we can observe that the improvements to the objective value often occur at the expense of the other measures. This is particularly evident when we used MMRE and MEMRE as objective function. Indeed, as we can observe from Figures 3 and 4 when MMRE is used as objective function, MMRE is the best and MEMRE the worst, and vice versa. This behavior seems to be mitigated employing Avg(MMRE,MEMRE) as objective function. Moreover, it appears clear that objective functions based on SSR, Pred(25), MdMRE or their combinations perform better across a wide range of accuracy statistics, indeed when used as objective function these measures are able to guide towards estimation models with better accuracy in terms of the selected criterion without degrading so much the other summary measures.

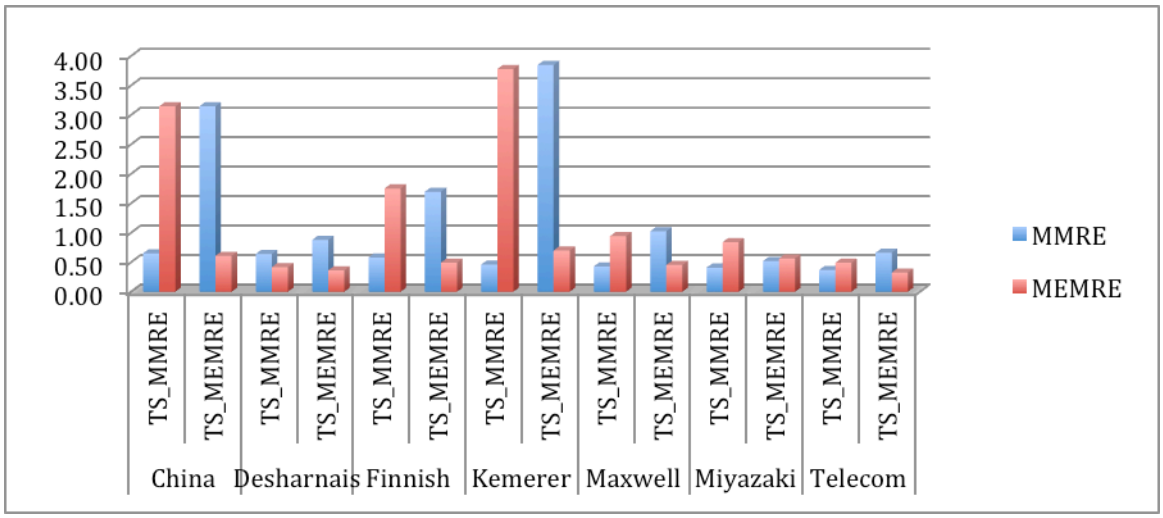

Figure 3. Comparing MMRE and MEMRE values achieved by TS using MMRE and MEMRE as objective functions.

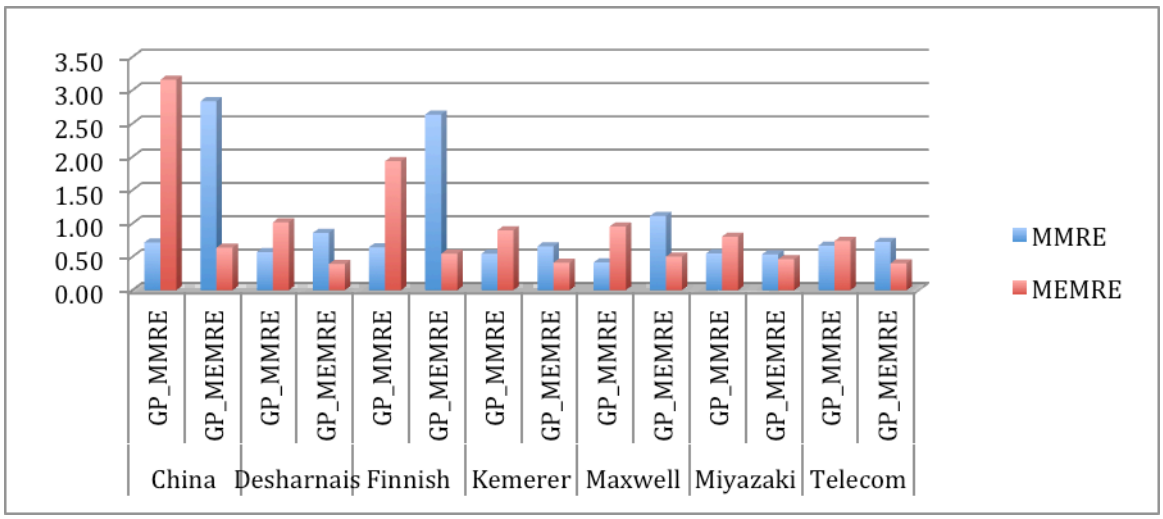

Figure 4. Comparing the MMRE and MEMRE values achieved by GP using MMRE and MEMRE as objective functions.

Table 23 and 24 report the results of the Wilcoxon Test on the square residuals obtained by TS and GP, respectively, using different objective functions. It is worth noting that we applied a Bonferroni

correction, i.e., 0.05/3=0.016, since we performed three tests (one for each baseline benchmark) to address research questions RQ6a and RQ6b. In the following we report for each dataset the objective functions that allowed us to obtain estimates better than those achieved by all the employed baseline benchmarks:

- China
  - TS_SSR, TS_MdMRE, TS_Avg(MMRE, MEMRE)
  - GP_SSR, GP_MdMRE, GP_Avg(MMRE, MEMRE), GP_Pred(25)/MdMRE
- Desharnais
  - TS_SSR, TS_MdMRE, TS_MdEMRE, TS_Pred(25), TS_Avg(MMRE, MEMRE), TS_Pred(25)/MMRE, TS_Pred(25)/MEMRE, TS_Pred(25)/MdMRE, TS_Pred(25)/MdEMRE
  - GP_SSR, GP_MdMRE, GP_MdEMRE, GP_Pred(25), GP_Avg(MMRE, MEMRE), GP_Pred(25)/MMRE, GP_Pred(25)/MdMRE, GP_Pred(25)/MdEMRE
- Finnish
  - TS_SSR, TS_MMRE, TS_MdMRE, TS_MdEMRE, TS_Pred(25), TS_Avg(MMRE, MEMRE), TS_Pred(25)/MdMRE,TS_Pred(25)/MdEMRE
  - GP_SSR, GP_Avg(MMRE, MEMRE), GP_Pred(25)/MMRE

- Maxwell
  - TS_SSR, TS_MdMRE, TS_MdEMRE, TS_ Pred(25), TS_Avg(MMRE, MEMRE), TS_ Pred(25)/MEMRE, TS_Pred(25)/MdMRE, TS_ Pred (25)/MdEMRE
  - GP_SSR, GP_Avg(MMRE, MEMRE)
- Miyazaki
  - TS_SSR, TS_MMRE, TS_MEMRE, TS_MdMRE, TS_MdEMRE, TS_Pred(25), TS_Avg(MMRE, MEMRE), TS_Pred(25)/MMRE, TS_Pred(25)/MEMRE, TS_Pred(25)/MdMRE, TS_Pred (25)/MdEMRE

  - GP_SSR, GP_MMRE, GP_MEMRE, GP_MdMRE, GP_MdEMRE, GP_Pred(25), GP_Avg(MMRE, MEMRE), GP_Pred(25)/MMRE, GP_Pred(25)/MEMRE, GP_Pred(25)/MdMRE, GP_Pred(25)/MdEMRE
- Telecom
  - TS_SSR, TS_MMRE, TS_MdMRE,TS_MdEMRE, TS_Pred(25),TS_Avg(MMRE, MEMRE), TS_Pred(25)/MMRE
  - GP_SSR, GP_Avg(MMRE, MEMRE)

We can observe that only using TS with SSR, MdMRE, MdEMRE, Avg(MMRE,MEMRE) and GP with SSR and Avg(MMRE, MEMRE) as objective functions we obtained significantly superior results with respect to all the baseline benchmarks on all the employed datasets except for Kemerer where no significant difference was found. This is probably due to the fact dataset probably due the fact that this dataset is very small – both in terms of number of features (i.e., 1) and projects (i.e., 15) - and as pointed out in the literature the Wilcoxon Test might be not powerful enough to confirm a statistical difference in case of few observations, even when the data seem to suggest such a difference [4]. Thus, we can positively answer to research questions RQ6a and RQ6b (i.e., the employed objective function impact on the accuracy of the estimation models built with TS and GP).

Table 21 Accuracy measures achieved on test sets using TS with the employed fitness functions

Dataset	Technique	MMRE	MdMRE	Pred(25)	MEMRE	MdEMRE	SSR
China	TS_SSR	1.13	0.62	0.16	1.22	0.62	1.04E+10
	TS_MMRE	0.65	0.68	0.16	3.15	1.77	1.80E+10
	TS_MdMRE	1.13	0.55	0.12	1.20	0.65	1.25E+10
	TS_Pred(25)	1.82	0.68	0.09	1.04	0.65	3.44E+10
	TS_MEMRE	3.15	1.35	0.15	0.61	0.63	1.35E+11
	TS_MdEMRE	1.60	0.64	0.17	0.87	0.54	1.75E+10
	TS_AVG(MMRE.MEMRE)	1.07	0.59	0.14	1.02	0.61	1.31E+10
	TS_Pred(25)/MMRE	0.70	0.63	0.15	2.40	1.23	1.57E+10

	TS_Pred(25)/MdMRE	1.26	0.55	0.18	1.06	0.60	1.36E+10
	TS_Pred(25)/MEMRE	2.60	0.94	0.17	0.65	0.59	9.89E+10
	TS_Pred(25)/MdEMRE	1.62	0.63	0.15	0.94	0.58	2.06E+10
Desharnais	TS_SSR	0.76	0.36	0.34	0.40	0.35	5.41E+08
	TS_MMRE	0.64	0.34	0.40	0.42	0.31	6.05E+08
	TS_MdMRE	0.64	0.29	0.46	0.49	0.30	8.07E+08
	TS_Pred(25)	0.71	0.33	0.38	0.44	0.33	8.28E+08
	TS_MEMRE	0.88	0.35	0.40	0.36	0.34	8.21E+08
	TS_MdEMRE	0.71	0.32	0.44	0.44	0.30	7.76E+08
	TS_AVG(MMRE.MEMRE)	0.60	0.33	0.43	0.44	0.35	6.40E+08
	TS_Pred(25)/MMRE	0.57	0.32	0.44	0.49	0.33	7.24E+08
	TS_Pred(25)/MdMRE	0.66	0.33	0.44	0.46	0.30	7.75E+08
	TS_Pred(25)/MEMRE	0.90	0.34	0.42	0.37	0.35	9.25E+08
	TS_Pred(25)/MdEMRE	0.69	0.34	0.42	0.44	0.29	7.42E+08
Finnish	TS_SSR	0.97	0.53	0.23	0.63	0.44	7.48E+08
	TS_MMRE	0.58	0.56	0.22	1.75	1.18	2.07E+09
	TS_MdMRE	0.89	0.37	0.26	0.76	0.51	9.62E+08
	TS_Pred(25)	1.00	0.56	0.32	0.71	0.47	9.20E+08
	TS_MEMRE	1.70	1.08	0.28	0.49	0.50	1.72E+09
	TS_MdEMRE	1.12	0.52	0.23	0.62	0.37	1.34E+09
	TS_AVG(MMRE.MEMRE)	0.74	0.51	0.31	0.68	0.52	1.03E+09
	TS_Pred(25)/MMRE	0.61	0.54	0.19	1.89	1.25	2.16E+09
	TS_Pred(25)/MdMRE	0.83	0.38	0.37	0.78	0.54	9.41E+08
	TS_Pred(25)/MEMRE	1.13	0.59	0.35	0.56	0.45	9.73E+08
	TS_Pred(25)/MdEMRE	1.02	0.60	0.40	0.64	0.45	8.80E+08
Kemerer	TS_SSR	0.63	0.42	0.33	1.45	0.70	1.19E+06
	TS_MMRE	0.46	0.45	0.40	3.78	3.34	1.35E+06
	TS_MdMRE	0.58	0.31	0.47	1.71	0.63	1.24E+06
	TS_Pred(25)	0.53	0.38	0.33	2.89	2.17	1.30E+06
	TS_MEMRE	3.85	2.69	0.20	0.70	0.64	1.00E+07
	TS_MdEMRE	2.21	1.62	0.40	1.54	0.41	7.54E+06
	TS_AVG(MMRE.MEMRE)	0.63	0.43	0.33	1.45	0.70	1.19E+06
	TS_Pred(25)/MMRE	0.46	0.43	0.40	3.81	3.35	1.35E+06
	TS_Pred(25)/MdMRE	0.58	0.32	0.47	1.73	0.66	1.24E+06
	TS_Pred(25)/MEMRE	0.59	0.36	0.47	1.59	0.67	1.22E+06
	TS_Pred(25)/MdEMRE	0.58	0.32	0.47	1.71	0.64	1.24E+06
Maxwell	TS_SSR	0.75	0.52	0.26	0.48	0.47	1.80E+09
	TS_MMRE	0.43	0.46	0.32	0.94	0.45	4.15E+09
	TS_MdMRE	0.55	0.34	0.29	0.64	0.44	3.37E+09
	TS_Pred(25)	0.67	0.49	0.34	0.68	0.49	3.21E+09

	TS_MEMRE	1.03	0.71	0.21	0.45	0.48	2.48E+09
	TS_MdEMRE	0.53	0.47	0.29	0.64	0.35	2.76E+09
	TS_AVG(MMRE.MEMRE)	0.52	0.45	0.32	0.55	0.41	2.25E+09
	TS_Pred(25)/MMRE	0.44	0.50	0.39	0.83	0.43	3.96E+09
	TS_Pred(25)/MdMRE	0.53	0.37	0.40	0.70	0.46	3.73E+09
	TS_Pred(25)/MEMRE	0.87	0.56	0.39	0.46	0.47	2.15E+09
	TS_Pred(25)/MdEMRE	0.53	0.41	0.34	0.75	0.48	3.85E+09
Miyazaki	TS_SSR	0.52	0.37	0.33	0.54	0.40	3.72E+10
	TS_MMRE	0.41	0.35	0.35	0.84	0.52	4.21E+10
	TS_MdMRE	0.46	0.36	0.27	0.68	0.40	3.95E+10
	TS_Pred(25)	0.45	0.36	0.33	0.79	0.47	4.05E+10
	TS_MEMRE	0.52	0.37	0.35	0.56	0.42	3.86E+10
	TS_MdEMRE	0.51	0.50	0.21	1.46	0.84	5.26E+10
	TS_AVG(MMRE.MEMRE)	0.49	0.34	0.35	0.59	0.41	3.89E+10
	TS_Pred(25)/MMRE	0.42	0.36	0.40	0.82	0.50	4.00E+10
	TS_Pred(25)/MdMRE	0.47	0.34	0.35	0.61	0.36	3.82E+10
	TS_Pred(25)/MEMRE	0.48	0.34	0.40	0.59	0.38	3.89E+10
	TS_Pred(25)/MdEMRE	0.48	0.35	0.42	0.59	0.36	3.78E+10
Telecom	TS_SSR	0.67	0.27	0.50	0.35	0.26	4.93E+05
	TS_MMRE	0.37	0.23	0.56	0.49	0.23	6.39E+05
	TS_MdMRE	0.70	0.22	0.56	0.39	0.30	5.71E+05
	TS_Pred(25)	0.78	0.22	0.61	0.40	0.28	5.52E+05
	TS_MEMRE	0.66	0.35	0.33	0.33	0.26	7.36E+05
	TS_MdEMRE	0.89	0.27	0.50	0.43	0.26	7.71E+05
	TS_AVG(MMRE.MEMRE)	0.40	0.22	0.56	0.41	0.21	6.43E+05
	TS_Pred(25)/MMRE	0.38	0.20	0.61	0.49	0.21	6.25E+05
	TS_Pred(25)/MdMRE	0.75	0.24	0.50	0.39	0.30	5.94E+05
	TS_Pred(25)/MEMRE	0.87	0.25	0.50	0.41	0.27	8.31E+05
	TS_Pred(25)/MdEMRE	0.80	0.23	0.56	0.38	0.25	6.45E+05

Table 22 Accuracy measures achieved on test sets using GP with the employed fitness functions

Dataset	**Technique**	**MMRE**	**MdMRE**	**Pred(25)**	**MEMRE**	**MdEMRE**	**SSR**
China	GP_SSR	1.37	0.57	0.18	0.88	0.60	1.52E+10
	GP_MMRE	0.72	0.69	0.12	3.16	1.64	2.00E+10
	GP_MdMRE	1.18	0.57	0.16	1.06	0.61	1.40E+10
	GP_Pred(25)	1.43	0.59	0.16	1.04	0.62	1.53E+10
	GP_MEMRE	2.84	1.20	0.14	0.64	0.63	5.47E+10
	GP_MdEMRE	1.81	0.62	0.16	0.77	0.59	2.16E+10
	GP_AVG(MMRE.MEMRE)	1.10	0.56	0.18	1.00	0.61	1.37E+10
	GP_Pred(25)/MMRE	0.80	0.63	0.10	2.23	1.07	1.77E+10

	GP_Pred(25)/MdMRE	1.37	0.57	0.17	1.02	0.61	1.43E+10
	GP_Pred(25)/MEMRE	2.27	0.88	0.16	0.66	0.58	3.39E+10
	GP_Pred(25)/MdEMRE	1.56	0.60	0.21	0.83	0.55	1.49E+10
Desharnais	GP_SSR	0.79	0.36	0.43	0.42	0.35	9.51E+08
	GP_MMRE	0.57	0.50	0.18	1.02	0.71	1.35E+09
	GP_MdMRE	0.75	0.34	0.43	0.44	0.35	7.87E+08
	GP_Pred(25)	0.73	0.35	0.39	0.44	0.35	7.57E+08
	GP_MEMRE	0.86	0.39	0.39	0.39	0.35	7.44E+08
	GP_MdEMRE	0.80	0.34	0.39	0.42	0.34	8.02E+08
	GP_AVG(MMRE.MEMRE)	0.67	0.35	0.39	0.46	0.40	8.29E+08
	GP_Pred(25)/MMRE	0.64	0.38	0.34	0.54	0.41	9.21E+08
	GP_Pred(25)/MdMRE	0.74	0.34	0.40	0.44	0.36	7.87E+08
	GP_Pred(25)/MEMRE	0.82	0.37	0.40	0.40	0.33	7.56E+08
	GP_Pred(25)/MdEMRE	0.79	0.35	0.38	0.42	0.35	7.81E+08
Finnish	GP_SSR	1.31	0.83	0.26	0.65	0.61	1.53E+09
	GP_MMRE	0.64	0.65	0.19	1.94	1.38	2.37E+09
	GP_MdMRE	1.73	0.87	0.23	0.56	0.52	1.53E+09
	GP_Pred(25)	1.85	0.94	0.03	0.89	0.72	2.99E+09
	GP_MEMRE	2.64	1.17	0.19	0.55	0.51	2.53E+09
	GP_MdEMRE	1.73	0.87	0.23	0.56	0.52	2.06E+09
	GP_AVG(MMRE.MEMRE)	0.97	0.67	0.25	0.78	0.58	1.43E+09
	GP_Pred(25)/MMRE	0.65	0.65	0.14	1.79	1.31	2.22E+09
	GP_Pred(25)/MdMRE	2.04	1.00	0.13	0.67	0.61	2.17E+09
	GP_Pred(25)/MEMRE	2.40	0.90	0.12	0.57	0.58	2.39E+09
	GP_Pred(25)/MdEMRE	1.97	1.04	0.18	0.63	0.54	2.55E+09
Kemerer	GP_SSR	1.08	0.90	0.00	1.08	0.90	1.02E+06
	GP_MMRE	0.55	0.44	0.27	0.90	0.80	8.03E+05
	GP_MdMRE	0.60	0.36	0.27	0.58	0.52	5.68E+05
	GP_Pred(25)	0.69	0.56	0.13	0.86	0.85	7.44E+05
	GP_MEMRE	0.66	0.36	0.33	0.41	0.28	6.10E+05
	GP_MdEMRE	0.60	0.36	0.27	0.58	0.52	5.68E+05
	GP_AVG(MMRE.MEMRE)	0.47	0.32	0.33	0.43	0.34	5.99E+05
	GP_Pred(25)/MMRE	0.54	0.39	0.27	0.71	0.65	7.13E+05
	GP_Pred(25)/MdMRE	0.62	0.42	0.20	0.88	0.91	6.77E+05
	GP_Pred(25)/MEMRE	0.59	0.33	0.33	0.41	0.33	5.61E+05
	GP_Pred(25)/MdEMRE	0.60	0.35	0.33	0.59	0.54	5.71E+05
Maxwell	GP_SSR	0.74	0.48	0.16	0.47	0.45	2.02E+09
	GP_MMRE	0.42	0.43	0.32	0.96	0.47	3.92E+09
	GP_MdMRE	0.70	0.54	0.15	0.81	0.50	4.29E+09
	GP_Pred(25)	0.92	0.53	0.16	0.63	0.52	3.96E+09

	GP_MEMRE	1.11	0.91	0.21	0.50	0.51	3.78E+09
	GP_MdEMRE	0.64	0.49	0.18	0.73	0.47	3.36E+09
	GP_AVG(MMRE.MEMRE)	0.56	0.47	0.14	0.64	0.44	2.88E+09
	GP_Pred(25)/MMRE	0.46	0.48	0.26	1.00	0.50	4.17E+09
	GP_Pred(25)/MdMRE	0.832174424	0.56	0.08	0.74	0.52	4.24E+09
	GP_Pred(25)/MEMRE	1.132274858	0.62	0.11	0.59	0.53	4.40E+09
	GP_Pred(25)/MdEMRE	0.86	0.66	0.11	0.74	0.56	4.29E+09
Miyazaki	GP_SSR	0.52	0.42	0.25	1.08	0.76	3.56E+10
	GP_MMRE	0.55	0.42	0.27	0.80	0.66	4.29E+10
	GP_MdMRE	0.51	0.33	0.40	0.51	0.36	3.61E+10
	GP_Pred(25)	0.52	0.36	0.27	0.58	0.40	3.77E+10
	GP_MEMRE	0.54	0.33	0.40	0.46	0.35	3.46E+10
	GP_MdEMRE	0.51	0.33	0.40	0.51	0.36	3.61E+10
	GP_AVG(MMRE.MEMRE)	0.54	0.33	0.40	0.48	0.35	3.63E+10
	GP_Pred(25)/MMRE	0.55	0.40	0.29	0.71	0.57	3.94E+10
	GP_Pred(25)/MdMRE	0.51	0.34	0.33	0.56	0.37	3.82E+10
	GP_Pred(25)/MEMRE	0.53	0.32	0.40	0.48	0.36	3.51E+10
	GP_Pred(25)/MdEMRE	0.51	0.33	0.40	0.51	0.36	3.62E+10
Telecom	GP_SSR	0.52	0.36	0.33	0.51	0.43	8.37E+05
	GP_MMRE	0.67	0.49	0.22	0.74	0.64	8.23E+05
	GP_MdMRE	0.81	0.51	0.22	0.49	0.47	8.03E+05
	GP_Pred(25)	0.74	0.54	0.17	0.55	0.48	7.93E+05
	GP_MEMRE	0.73	0.54	0.33	0.40	0.35	8.06E+05
	GP_MdEMRE	0.81	0.51	0.22	0.49	0.47	8.03E+05
	GP_AVG(MMRE.MEMRE)	0.66	0.46	0.39	0.45	0.33	6.39E+05
	GP_Pred(25)/MMRE	0.72	0.55	0.33	0.47	0.38	7.79E+05
	GP_Pred(25)/MdMRE	0.74	0.53	0.17	0.51	0.50	7.54E+05
	GP_Pred(25)/MEMRE	0.72	0.55	0.33	0.47	0.38	7.79E+05
	GP_Pred(25)/MdEMRE	0.79	0.52	0.22	0.50	0.44	7.66E+05

Table 23 Results of the Wilcoxon tests comparing TS objective functions on test sets

Dataset	<	**Random**	**Mean**	**Median**
China	TS_SSR	**0**	**0**	**0.005**
	TS_MMRE	**0**	**0**	0.968
	TS_MEMRE	1	0.991	1
	TS_MdMRE	**0**	**0**	**0.011**

	TS_MdEMRE	**0**	**0**	0.480
	TS_Pred(25)	0.542	**0**	1
	TS_Avg(MMRE, MEMRE)	**0**	**0**	**0**
	TS_Pred(25)/MMRE	**0**	**0**	0.281
	TS_Pred(25)/MEMRE	0.999	0.211	1
	TS_Pred(25)/MdMRE	**0**	**0**	0.108
	TS_Pred(25)/MdEMRE	0.031	**0**	0.934
Desharnais	<	**Random**	**Mean**	**Median**
	TS_SSR	**0.003**	**0**	**0.012**
	TS_MMRE	**0**	**0**	**0**
	TS_MEMRE	0.087	**0.001**	0.062
	TS_MdMRE	**0**	**0**	**0.001**
	TS_MdEMRE	**0.001**	**0**	**0.003**
	TS_Pred(25)	**0.008**	**0**	**0.007**
	TS_Avg(MMRE, MEMRE)	**0**	**0**	**0**
	TS_Pred(25)/MMRE	**0**	**0**	**0**
	TS_Pred(25)/MEMRE	**0.001**	**0.001**	**0.001**
	TS_Pred(25)/MdMRE	**0.001**	**0.001**	**0.001**
	TS_Pred(25)/MdEMRE	**0.001**	**0**	**0.002**
Finnish	<	**Random**	**Mean**	**Median**
	TS_SSR	**0.003**	**0**	**0.001**
	TS_MMRE	**0**	**0.042**	0.023
	TS_MEMRE	0.262	0.055	0.104
	TS_MdMRE	**0.006**	**0**	**0**
	TS_MdEMRE	**0.011**	**0.003**	**0.005**
	TS_Pred(25)	**0.04**	**0.002**	**0.01**
	TS_Avg(MMRE, MEMRE)	**0**	**0**	**0**

	TS_Pred(25)/MMRE	**0**	0.112	0.076
	TS_Pred(25)/MEMRE	**0.07**	**0.005**	0.021
	TS_Pred(25)/MdMRE	**0.002**	**0**	**0**
	TS_Pred(25)/MdEMRE	0.036	**0.002**	**0.012**
Kemerer	<	**Random**	**Mean**	**Median**
	TS_SSR	**0.001**	0.556	0.623
	TS_MMRE	0.189	0.725	0.918
	TS_MEMRE	0.992	0.986	0.998
	TS_MdMRE	**0.001**	0.579	0.623
	TS_MdEMRE	0.511	0.853	0.941
	TS_Pred(25)	**0.008**	0.685	0.878
	TS_Avg(MMRE, MEMRE)	**0.001**	0.556	0.623
	TS_Pred(25)/MMRE	0.147	0.725	0.909
	TS_Pred(25)/MEMRE	**0.001**	0.556	0.623
	TS_Pred(25)/MdMRE	**0.001**	0.579	0.685
	TS_Pred(25)/MdEMRE	**0.001**	0.579	0.623
Maxwell	<	**Random**	**Mean**	**Median**
	TS_SSR	**0**	**0.002**	**0.003**
	TS_MMRE	**0**	**0.004**	0.05
	TS_MEMRE	**0**	**0.014**	0.189
	TS_MdMRE	**0**	**0**	**0.001**
	TS_MdEMRE	**0**	**0**	**0.003**
	TS_Pred(25)	**0**	**0.004**	**0.014**
	TS_Avg(MMRE, MEMRE)	**0**	**0**	**0.001**
	TS_Pred(25)/MMRE	**0**	**0.002**	0.018
	TS_Pred(25)/MEMRE	**0**	**0.001**	**0.002**
	TS_Pred(25)/MdMRE	**0**	**0**	**0.004**

	TS_Pred(25)/MdEMRE	**0**	**0.001**	**0.012**
Miyazaki	**<**	**Random**	**Mean**	**Median**
	TS_SSR	**0**	**0**	**0**
	TS_MMRE	**0**	**0**	**0.001**
	TS_MEMRE	**0**	**0**	**0**
	TS_MdMRE	**0**	**0**	**0**
	TS_MdEMRE	**0.003**	**0**	**0.011**
	TS_Pred(25)	**0**	**0**	**0**
	TS_Avg(MMRE, MEMRE)	**0**	**0**	**0**
	TS_Pred(25)/MMRE	**0**	**0**	**0**
	TS_Pred(25)/MEMRE	**0**	**0**	**0**
	TS_Pred(25)/MdMRE	**0**	**0**	**0**
	TS_Pred(25)/MdEMRE	**0**	**0**	**0**
Telecom	**<**	**Random**	**Mean**	**Median**
	TS_SSR	**0.003**	**0.008**	**0.004**
	TS_MMRE	**0.004**	**0.006**	**0.002**
	TS_MEMRE	0.069	0.058	0.045
	TS_MdMRE	**0.004**	**0.007**	**0.003**
	TS_MdEMRE	**0.003**	**0.008**	**0.004**
	TS_Pred(25)	**0.007**	**0.009**	**0.006**
	TS_Avg(MMRE, MEMRE)	**0.015**	**0.013**	**0.01**
	TS_Pred(25)/MMRE	**0.004**	**0.008**	**0.002**
	TS_Pred(25)/MEMRE	**0.007**	0.023	**0.003**
	TS_Pred(25)/MdMRE	**0.003**	0.020	**0.003**
	TS_Pred(25)/MdEMRE	**0.003**	0.020	**0.003**

Table 24 Results of the Wilcoxon tests comparing GP objective functions on test sets

Dataset	<	**Random**	**Mean**	**Median**
China	GP_SSR	**0**	**0**	**0.001**
	GP_MMRE	**0**	**0.001**	0.997
	GP_MEMRE	1	0.43	1
	GP_MdMRE	**0**	**0**	**0.005**
	GP_MdEMRE	0.027	**0**	0.769
	GP_Pred(25)	**0**	**0**	0.123
	GP_Avg(MMRE, MEMRE)	**0**	**0**	**0**
	GP_Pred(25)/MMRE	**0**	**0**	0.281
	GP_Pred(25)/MEMRE	0.972	0.005	1
	GP_Pred(25)/MdMRE	**0**	**0**	0.024
	GP_Pred(25)/MdEMRE	**0**	**0**	0.061
Desharnais	<	**Random**	**Mean**	**Median**
	GP_SSR	**0.008**	**0.003**	**0.006**
	GP_MMRE	**0**	0.182	0.62
	GP_MEMRE	0.066	**0.003**	0.039
	GP_MdMRE	**0.001**	**0**	**0.002**
	GP_MdEMRE	**0.003**	**0.001**	**0.006**
	GP_Pred(25)	**0.001**	**0.001**	**0.002**
	GP_Avg(MMRE, MEMRE)	**0**	**0.001**	**0.001**
	GP_Pred(25)/MMRE	**0**	**0.004**	**0.009**
	GP_Pred(25)/MEMRE	0.023	**0.002**	**0.022**
	GP_Pred(25)/MdMRE	**0.001**	**0**	**0.001**
	GP_Pred(25)/MdEMRE	**0.003**	**0.001**	**0.006**
Finnish	<	**Random**	**Mean**	**Median**

	GP_SSR	**0.019**	**0.001**	**0.003**
	GP_MMRE	**0**	0.166	0.213
	GP_MEMRE	0.437	0.321	0.414
	GP_MdMRE	0.123	0.024	0.045
	GP_MdEMRE	0.123	0.024	0.045
	GP_Pred(25)	0.23	0.46	0.483
	GP_Avg(MMRE, MEMRE)	**0.001**	**0**	**0.001**
	GP_Pred(25)/MMRE	**0**	0.048	0.038
	GP_Pred(25)/MEMRE	0.221	0.123	0.145
	GP_Pred(25)/MdMRE	0.138	0.070	0.158
	GP_Pred(25)/MdEMRE	**0**	**0**	**0**
Kemerer	<	**Random**	**Mean**	**Median**
	GP_SSR	0.644	0.601	0.811
	GP_MMRE	0.002	0.147	0.025
	GP_MEMRE	0.066	0.091	0.147
	GP_MdMRE	**0.011**	0.101	0.022
	GP_MdEMRE	**0.011**	0.101	0.022
	GP_Pred(25)	0.082	0.239	0.705
	GP_Avg(MMRE, MEMRE)	**0.014**	0.019	0.037
	GP_Pred(25)/MMRE	**0.004**	0.101	**0.014**
	GP_Pred(25)/MEMRE	0.029	0.053	0.066
	GP_Pred(25)/MdMRE	**0.012**	0.111	0.335
	GP_Pred(25)/MdEMRE	**0.011**	0.091	0.019
Maxwell	<	**Random**	**Mean**	**Median**
	GP_SSR	**0**	**0.006**	**0.014**
	GP_MMRE	**0**	**0.004**	0.046

	GP_MEMRE	**0.001**	0.097	0.581
	GP_MdMRE	**0**	**0**	**0**
	GP_MdEMRE	**0**	0.015	0.145
	GP_Pred(25)	**0**	**0.002**	0.031
	GP_Avg(MMRE, MEMRE)	**0**	**0.002**	**0.013**
	GP_Pred(25)/MMRE	**0**	0.006	0.085
	GP_Pred(25)/MEMRE	**0.001**	0.143	0.713
	GP_Pred(25)/MdMRE	**0**	**0.021**	0.287
	GP_Pred(25)/MdEMRE	**0.003**	0.127	0.594
Miyazaki	<	**Random**	**Mean**	**Median**
	GP_SSR	**0.001**	**0**	**0.002**
	GP_MMRE	**0**	**0**	**0.001**
	GP_MEMRE	**0**	**0**	**0**
	GP_MdMRE	**0**	**0**	**0**
	GP_MdEMRE	**0**	**0**	**0**
	GP_Pred(25)	**0**	**0**	**0**
	GP_Avg(MMRE, MEMRE)	**0**	**0**	**0**
	GP_Pred(25)/MMRE	**0**	**0**	**0**
	GP_Pred(25)/MEMRE	**0**	**0**	**0**
	GP_Pred(25)/MdMRE	**0**	**0**	**0**
	GP_Pred(25)/MdEMRE	**0**	**0**	**0**
Telecom	<	**Random**	**Mean**	**Median**
	GP_SSR	**0.015**	**0.011**	**0.016**
	GP_MMRE	**0.01**	0.148	**0.015**
	GP_MEMRE	0.082	0.058	0.037
	GP_MdMRE	0.138	0.082	0.069

	GP_MdEMRE	0.138	0.082	0.069
	GP_Pred(25)	0.034	0.082	0.023
	GP_Avg(MMRE, MEMRE)	**0.013**	**0.014**	**0.013**
	GP_Pred(25)/MMRE	**0.009**	0.054	**0.01**
	GP_Pred(25)/MEMRE	0.037	0.082	0.031
	GP_Pred(25)/MdMRE	0.088	0.069	0.037
	GP_Pred(25)/MdEMRE	0.088	0.054	0.028

4.3 Validity Evaluation

The *construct*, *conclusion*, and *external* validity threats described in Section 3.4 hold also for the empirical study presented in this chapter.

CHAPTER 5: Using Tabu Search to Configure Support Vector Regression for Effort Estimation

Several studies have addressed the problem of obtaining early accurate effort estimates, (e.g., [13][14][15][88][98][120][121]), many of which focusing on the proposal and evaluation of techniques to construct predictive models able to estimate the effort of a new project exploiting information (actual effort and cost-drivers) related to past projects. In particular, recent studies [24][25][26] have investigated the effectiveness of Support Vector Regression (SVR) for software effort estimation. SVR is a technique based on Support Vector Machines, a family of Machine Learning algorithms that have been successfully applied for addressing several predictive data modeling problems [31][126]. The studies reported in [24][25] showed that SVR has potential use also for software development effort estimation; indeed it outperformed the most commonly adopted prediction techniques using the Tukutuku database [94], a cross-company dataset of Web projects widely adopted in Web effort estimation studies. It was argued that the main reason for that lies in the flexibility of the method. Indeed, SVR enables the use of kernels and parameter settings allowing the learning mechanism to better suit the characteristics of different chunks of data, which is a typical characteristic of cross-company datasets. However, the setting of parameters needs to be done carefully, since an inappropriate choice can lead to over- or under-fitting, heavily worsening the performance of the method [17][69]. Nevertheless, there are no guidelines on how to best select these parameters [116][130][132] since the appropriate setting depends on the characteristics of the employed dataset. Moreover, an examination of all possible values for parameters is not computationally affordable, as the search space is too large, also due to the interaction among parameters, which cannot be separately optimized.

The issues abovementioned have motivated us to investigate the use of Tabu Search (TS) to automatically select SVR parameters [26]. TS is a meta-heuristics search technique used to address several optimization problems [50]. The TS-based approach was first investigated in [26] employing SVR in combination with different kernels and variables' preprocessing strategies, using as dataset the Tukutuku database [94]. In particular, we compared SVR configured with TS (SVR+TS) with other effort estimation techniques, namely Manual StepWise Regression (MSWR), Case-Based

Reasoning (CBR), Bayesian Networks [92], and the Mean and Median effort of the training sets. SVR+TS gave us the best results ever achieved with the Tukutuku database. However, these results were based on two random splits of only one cross-company dataset and it is widely recognized that several empirical analysis are needed to generalize empirical findings. Thus, the aim of this paper is to further investigate the combination of TS and SVR using data from several single- and cross-company datasets. Let us recall that the former represents a dataset containing data on projects from a single software company while the latter includes project data gathered from several software companies. In our analysis, we employed 13 different datasets from the PROMISE repository and also other 8 datasets obtained by splitting the Tukutuku database according to the values of its four categorical variables (see Appendix A). The choice to use datasets from the PROMISE repository is motivated due to the following points:

- Availability of datasets on industrial software projects, representing a diversity of application domains and projects´ characteristics. This is also in line with recommendation made by Kitchenham and Mendes [71].
- Availability of projects that are not Web-based, thus enabling the assessment of the effectiveness of the estimation technique employed herein when applied to different types of applications – Web, using the Tukutuku, and non-Web, using the PROMISE datasets. We would also like to point out that, in our view, Web and software development differ in a number of areas, such as: application characteristics, primary technologies used, approach to quality delivered, development process drivers, availability of the application, customers (stakeholders), update rate (maintenance cycles), people involved in development, architecture and network, disciplines involved, legal, social, and ethical issues, and information structuring and design. A detailed discussion on this issue is provided in [99].
- Availability of single- and cross-company datasets, thus enabling the assessment of the estimation technique employed herein when applied to single- and cross-company datasets. We would also like to point out that the use of a cross-company dataset is particularly useful for companies that do not have their own data on past projects from which to obtain their estimates, or that have data on projects developed in different application domains and/or technologies. To date, several studies have investigated if estimates obtained using cross-company datasets can be as accurate as the ones obtained using single-company datasets (e.g.,

[14][65][75][88][95][82][97][133]) achieving different findings (see [74] for a systematic review).

In relation to the choice of SVR kernels and pre-processing strategies, we focused our analysis on the RBF kernel and a logarithmic transformation of the variables since they provided the best results in our previous study [26].

In order to verify whether the proposed TS technique is able to make a suitable choice of SVR parameters we also compared the estimates obtained with SVR+TS with those obtained applying SVR using different strategies for parameters selection, namely:

- random SVR configurations. This means that the same number of solutions investigated for SVR+TS was generated in a totally random fashion and the best one among them was selected according to the same criteria employed for SVR+TS. This is a natural benchmark when using meta-heuristics search techniques;
- default parameters employed by the Weka tool [52];
- the Grid-search algorithm provided by LibSVM [9].

In addition, we also assessed whether the estimates provided by the proposed approach were better than those obtained using the Mean and Median effort (popular and simple benchmarks for effort estimation techniques) and those achieved with MSWR and CBR. These techniques were chosen because they are the two techniques widely used in the literature and also in industry, and the mostly employed estimation techniques [84].

Consequently, the research questions addressed in this paper are:

RQ7: Is Tabu Search able to effectively set Support Vector Regression parameters?

RQ8: Are the effort predictions obtained by using Support Vector Regression configured with Tabu Search significantly superior to the ones obtained by other techniques?

The remainder of the chapter is organized as follows. Section 5.1 first reports on the main aspects of SVR and TS and then describes the proposed approach based on TS to set-up SVR parameters. Section 5.2 presents the design of our empirical study, i.e., the datasets, the null hypotheses, the validation method, and the evaluation criteria employed to assess the prediction accuracy. Results are presented in Section 5.3, followed by a discussion on the empirical study validity in Section 5.4.

5.1 Using Support Vector Regression in combination with Tabu Search for effort estimation

In this section, we describe Support Vector Regression, Tabu Search, and how we have combined them for effort estimation.

5.1.1 Support Vector Regression

Support Vector Regression is a regression technique based on Support Vector (SV) machines, a learning approach originally introduced for linear binary classification. Linear classifiers construct a hyperplane separating the training set points belonging to the two classes. In the SV machine classifier [131][132], the hyperplane maximizes the classification margin, that is the minimum distance of the hyperplane from the training points [131], as shown in Figure 5. Choosing such optimal hyperplane requires the solution of a quadratic optimization problem subject to linear constraints, corresponding to the fact that each point of the training set must be correctly labeled. The hyperplane resulting from this optimization only depends on a subset of the training points, called *support vectors*. As an example, in Figure 5 the three points closest to the classification hyperplane are highlighted, as they represent the support vectors.

Thus, the system admits a solution only if there is a hyperplane separating the two classes in the training set (as in Figure 5), i.e., when the training set is linearly separable. Nevertheless, this can be considered too restrictive to be of any practical interest. Thus, in 1995, Cortes and Vapnik [28] defined a modified version of the approach, by introducing a parameter C to allow (but penalize) misclassifications in the training set, thus obtaining *soft margin* SVM's. The choice of the best value for C is crucial to performance, as it decides the trade-off between classification errors in the training set and model complexity [58][106].

When the SV approach is applied to a regression problem, a function has to be derived, which minimizes the deviation between observed and predicted values. To solve this problem we apply an SV approach that, rather than minimizing a function of the errors on the training set, aims at minimizing a bound on a generalized error, which also takes into account a regularization term in addition to the training error. Thus, the goal is to find a linear function that obtains an error lower than a constant ε on the training data and that at the same time is as flat as possible.

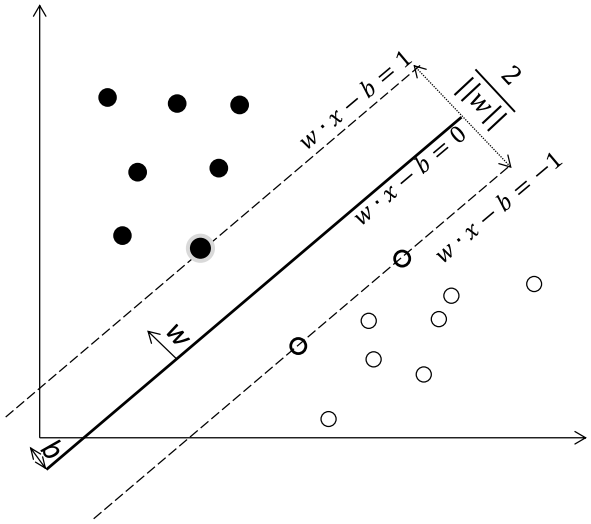

Figure 5. Hyperplane, margin and support vectors in linearly separable set

This formulation of the problem can be softened, as discussed above, by using parameter *C*, so that an error larger than the bound can be allowed on some of the points in the training set. Therefore, the parameter *C* determines the trade-off between the occurrences of errors larger than ε in the training set and the flatness of the function. On the other hand, ε controls the wideness of a tube such that points occurring inside are considered correct and only points outside the tube are evaluated as errors (see Figure 6). The two parameters are therefore strictly correlated, even if their suitable values depend on the dataset [18], so no rule of thumb exists.

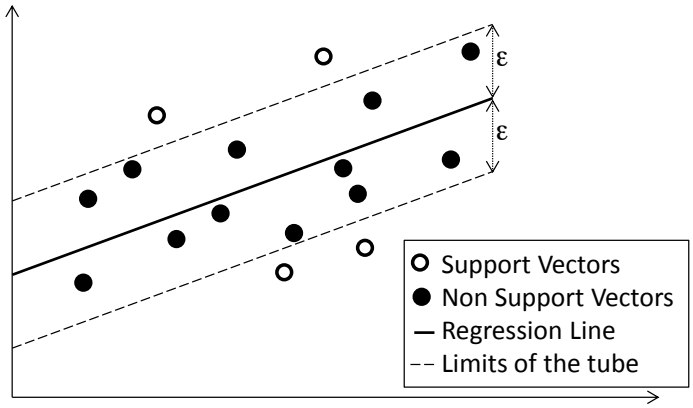

Figure 6. ε-tube in SVR

The non-linear case and the kernel choice

The SV approaches described above are conceived for the linear case. Thus, they could be not suitable for development effort estimation where the dependent variable (i.e., effort) does not necessarily linearly depend on the independent variables (i.e., cost drivers). To deal with the nonlinear case we can map the input vectors into a feature space before the linear SV approach is applied.

Mathematically, such mapping requires the substitution of dot products between the input vector $\boldsymbol{x}$ and each support vector $\boldsymbol{s}$, with a function describing their similarity in the feature space: such function k($\boldsymbol{x}$, $\boldsymbol{s}$) with two variables ($\boldsymbol{x}$ and $\boldsymbol{s}$) is called *kernel function*.

A wide variety of kernel functions has been proposed in the literature: a good overview can be found in [58]. An important kernel family is given by Radial Basis Function (RBF) where the output value only depends on the distance of the two points in the input space. In particular, the most popular kernel belonging to the RBF family is the Gaussian one, which is defined as follows:

$$k(x,s) = exp(-\gamma\ |x - s|^2),\ with\ \gamma>0\ . \qquad (1)$$

The Gaussian RBF kernel has been successfully applied in a variety of contexts, both alone (e.g., [106][124]) and in combination with SV approaches (e.g., [24][25][117]). Furthermore, Gaussian RBF kernel is usually suggested as the first choice in many practical guides (e.g., [60]) and is implemented in LibSVM, a popular library for SV approaches [17]. All the above considerations motivated our choice to use this kernel in the study reported in the present paper.

Using the Gaussian RBF kernel, a value for the kernel parameter γ needs to be selected in addition to the values for C and ε parameters. The main issue is how to set these parameters ensuring good generalization performance for a given dataset. In the following we report some existing approaches to address the problem and then describe our proposal.

SVR parameter setting

Many alternative strategies have been defined in the literature to select suitable values for SVR parameters. As pointed out in [18], many studies related to the use of SVR are based on the opinion of experts that select parameter values on the basis of their knowledge of both the approach and the application domain. Of course the reliance upon experts severely bounds the applicability of this approach. Another possibility is the use of heuristics based on noise characteristics [81]. However,

in addition to some technical limitations of these approaches, they require either an expert with a deep understanding of the problem or a statistical model for the noise. Parameters choice based on more direct information, such as the range of output values, are prone to other problems, including outliers [86].

In Grid-search approaches a certain number of parameter values are explored to identify the best option. Nevertheless, the points are chosen a-priori and do not depend on the specific case. For instance, the software library LibSVM provides a mechanism that explores a combination of 8 values for each of the parameters C, ε, and γ (in the ranges [1.0E-3, 32000], [1.0E-6, 1], and [1.0E-6, 8]) using a five-fold cross-validation on the training set [17]. Thus, a total of 512 fixed points are assessed and the one with the best cross-validation accuracy is returned. Even if Grid-searches are easy to apply, they have a main drawback: the search is performed always on the same (coarse grained) points, without taking into account the dataset to guide the search.

In [24][25] the problem was addressed in the context of effort estimation, adopting an automatic approach to explore a large number of parameter values (employing various nested cycles with small incremental steps). For each run, depending on the kernel, the number of executions ranged from some dozens to more than 4000 executions. An inner leave-one-out cross validation was performed on the training set (each cycle of execution required a number of iterations corresponding to the cardinality of the training set) and for each iteration the goodness of the solution was evaluated using a combination of effort accuracy estimation measures [1]. Thus, the setting providing the best estimation (according to the selected criterion) on the training set was chosen.

Although such optimization strategy included a quite large combination of parameter values, it proceeded by brute force, by predefined steps, and did not use any information related to the prior steps trying to improve the search. Moreover, it was computationally too expensive. Smarter optimization strategies, on the contrary, use all possible clues to focus the search in the most promising areas of parameter values for a given dataset. Among such strategies, in [26] we proposed the use of the meta-heuristics Tabu Search to search for the best parameter settings. This approach is further investigated in this paper and will be described in the next section. One of the strengths of

[1] The same combination of effort estimation measures is used as objective function in the present paper, so it will be detailed in Section 2.3.

the Tabu Search strategy is that it uses information both in a positive way, to focus the search, and in a negative way, to avoid already explored areas and loops.

5.1.2 Using Tabu Search to configure SVR

In this section we describe how we designed TS (see Chapter 1) for setting SVR parameters. Let us formulate our goal: starting from a dataset of past projects we have to identify a good solution S, represented by values for variables C, ε, and γ (see step 1 in Figure 7), so that SVR configured with those parameter values can accurately predict the unknown effort for new incoming projects (see step 2 in Figure 7). Thus, in this section we will detail step 1, whose process is illustrated in Figure 8.

An initial solution is generated by randomly choosing the values for each variable in a defined range. In particular, since the values for C, ε, and γ can vary from zero to infinity, an upper bound has usually to be chosen. To this end, we employ the same ranges of the Grid-search algorithm [60] for C, ε, and γ, respectively, and, as it is usual, we perform the search for parameter values in the logarithmic space of these ranges [60][70].

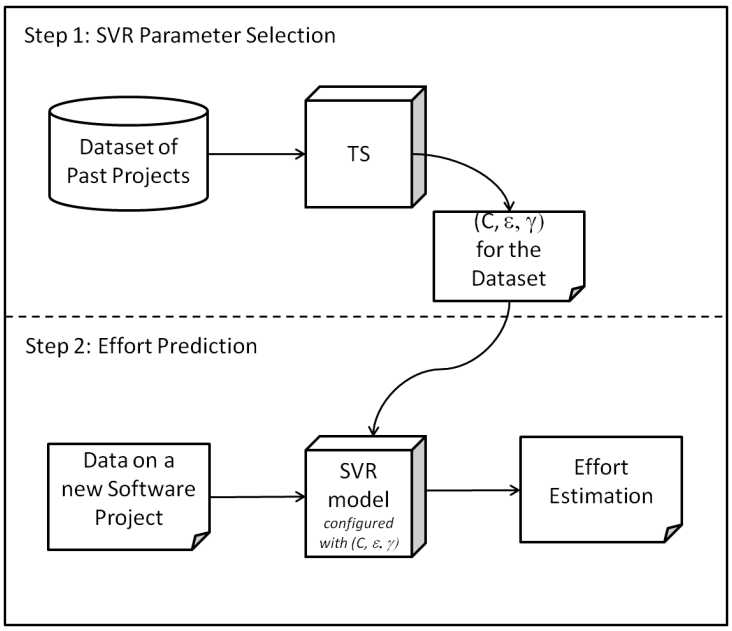

Figure 7. The two steps of applying SVR+TS: parameters identification and use

Starting from the random initial solution, at each iteration 25 moves are performed, each one according to the pseudocode provided in Figure 9 and explained herein. A parameter to be changed is selected among C, ε, and γ (with equal probability). The current value of the chosen parameter in the 80% of the cases is incremented up to its 20% adding (or subtracting, with the same probability) a random value, while in the remaining 20% of the cases the new parameter value is chosen in a totally random fashion within the specified range. The rationale for the percentage of 80% is to investigate as much as possible an actual promising solution. Indeed, once a "better" region on the space has been identified, a finer search on that region is conducted performing small changes around a potentially interesting solution (Figure 9 line 6). On the other hand, we defined also a mechanism to allow for a diversification in the search space (obtained using total random move) to escape from local optima (Figure 9 line 8).

Once all moves are performed, a set of 25 new neighboring solutions is created and the neighboring solution with the best objective function value and which is not tabu or matches an aspiration criterion is selected as current best solution and then as starting point to explore a new neighborhood in the next iteration. It is worth noting that a move is marked as tabu if it leads to a solution whose parameter values are very similar (i.e., the difference between parameter values is less than 10%) to those of a solution stored in the Tabu List. In order to allow one to revoke tabu moves, we employ the most commonly used aspiration criterion, namely we permit a tabu move if it results in a solution with an objective function value better than the one of the best solution reached so far. Moreover, if the current best solution's objective value is better than the one achieved by the best solution found so far, the latter is replaced. Finally, to avoid retracing the moves previously used, the current solution is stored in the Tabu List. Note that since only a fairly limited quantity of information is usually recorded in the Tabu List [50], we decided to employ a short-term memory of fixed length with 7 elements. The search is stopped after a fixed number of iterations is performed (i.e., 100). It is worth noting that we adopted the same choices for number of moves, Tabu List size, and iterations employed in our previous study [26]. Those numbers were empirically determined as it is usual when no guidelines are available. In particular, they were chosen for the work presented herein because our previous research showed that increasing them did not allow us to improve the estimation accuracy while wasting computation time.

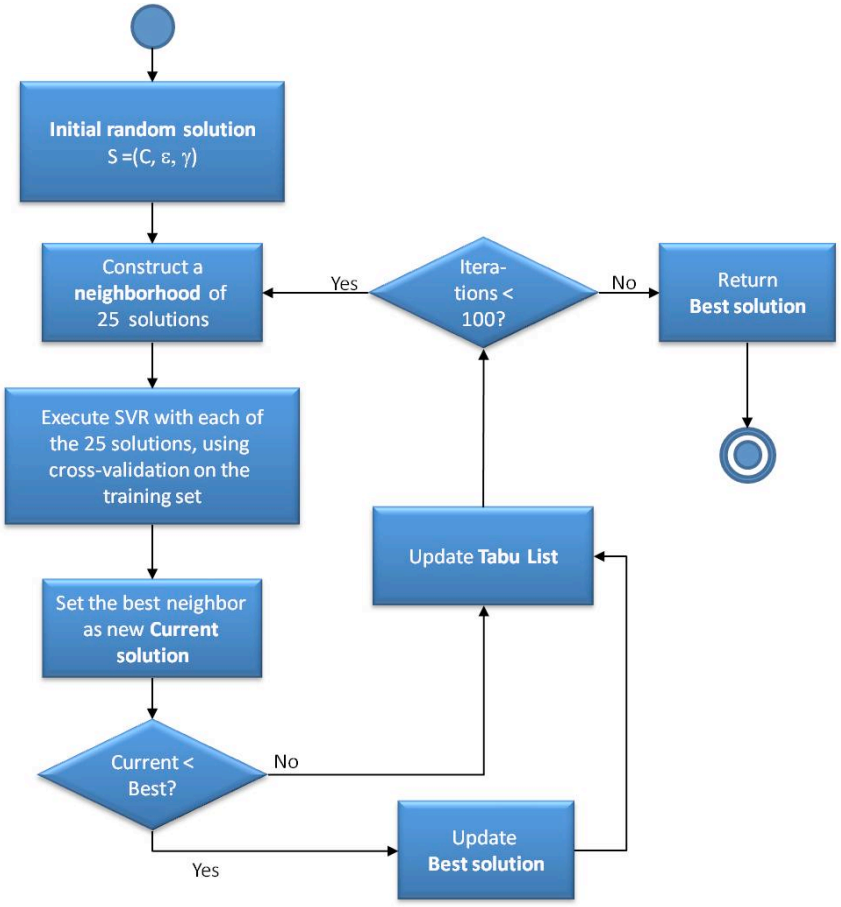

Figure 8. The proposed TS-based approach for SVR parameters selection

```
1 function applyMove(currentSolution):newSolution
2       newSolution=currentSolution
3       paramToChange= rand(C, ε,  γ)
4       p = rand(0,1)
5       if (p < 0.8) then
6               newValue = paramToChange ± rand(0, paramToChange*0.2)
7       else
8               newValue= rand(paramToChange.lowerBound, paramToChange.upperBound)
9       newSolution.paramToChange = newValue
10      return newSolution
```

Figure 9. The TS move

As for the objective function, a number of accuracy measures can be used to compare effort estimates, usually based on the residuals, i.e., the differences between predicted and actual efforts. Among them, two widely summary measures are the Mean Magnitude of Relative Error (MMRE) and the Mean Magnitude of Relative Error relative to the Estimate (MEMRE) [22][72]. Let us recall that MMRE is the Mean of MRE and MEMRE is the Mean of EMRE, where:

$$MRE = \frac{|e - \hat{e}|}{e} \qquad (2)$$

$$EMRE = \frac{|e - \hat{e}|}{\hat{e}} \qquad (3)$$

where e represents actual effort and $\hat{e}$ estimated effort. We can observe that EMRE has the same form of MRE, but the denominator is the estimate, giving thus a stronger penalty to under-estimates. In [24][26] [25] we employed as objective function, the mean of them:

$$Objective\ Function = (MMRE+MEMRE) / 2 \qquad (4)$$

The rationale was that, since MRE is more sensitive to overestimates and EMRE to underestimates, an objective function minimizing them should find better solutions. Since the present paper provides a further assessment of the technique proposed in [26], we exploited the same objective function.

It is worth noting that the solution we are proposing attempts to capture the necessary domain knowledge by using performance indicators as the objective function. On the other hand, it requires a meta-heuristics as robust as possible with respect to the target function characteristics, which are completely unexplored. We think that the TS strategy has these characteristics because of its capability to adapt to the input function both by concentrating search efforts on promising areas and keeping away from already visited regions by means of the Tabu List.

Finally, in order to cope with the non-deterministic nature of TS, we performed 10 executions of SVR+TS and, among the obtained configurations, we retained as final the one which provided objective value closest to the mean of the objective values obtained in the 10 executions.

5.2 Empirical Study Design

In this section, we present the design of the empirical study carried out to assess the effectiveness of the proposed approach. In particular, we present the employed datasets, the null hypotheses, the adopted validation method, and evaluation criteria. The results of the empirical analysis are discussed in Section 5.4.

5.2.1 Datasets

To carry out the empirical evaluation of the proposed technique we employed a total of 21 industry software project datasets selected both from the PROMISE repository [109] and the Tukutuku database [94]. PROMISE contains publicly available single and cross-company datasets, while the Tukutuku database contains data about Web projects (i.e., Web hypermedia systems and Web applications) developed in different companies and gathered by the Tukutuku project, which aimed to develop Web cost estimation models and to benchmark productivity across and within Web Companies.

Concerning the PROMISE repository, it is worth noting that we did not employ all the datasets that it contains, since we were interested only on the ones that can be employed for early effort estimation (i.e., datasets containing information that would be available at the early stages of a software development process), which is the managerial goal of our investigation. To this end, we avoided the use of datasets like NASA and COCOMO containing as size measures only features available once a project is completed, such as the Lines of Code (LOCs). Moreover, we pruned the remaining datasets from this kind of features, since their use could bias the results [120]. As for the categorical variables contained in some datasets, we used them as done in [79][120] to obtain different more homogenous splits from the original datasets or we excluded them from our analysis in case splitting was not possible (e.g., the resulting sub datasets were too small). As an example, we used the categorical variable “Languages” in the Desharnais dataset to split the original data into three different datasets corresponding to Languages 1, 2, and 3, respectively. After applying the above criteria, 13 PROMISE datasets were kept for our empirical analysis, namely Albrecht, China, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, MaxwellA2, MaxwellA3, MaxwellS2, MaxwellT1, Miyazaki, and Telecom. We applied the same procedure on the Tukutuku database

obtaining 8 splits since all the categorical variables (i.e., TypeProj, DocPro, ProImpr, and Metrics) were binary.

Table 8 summarizes the main characteristics of the considered datasets while further details together with the descriptive statistics of the involved features are provided in Appendix A. They represent an interesting sample of software projects, since they contain data about projects that are Web-based (i.e. the ones from Tukututku) and not Web-based (i.e., the ones from PROMISE) and include datasets that were collected from a single software company or several companies. Moreover, all the datasets contain data about industrial projects, representing a diversity of application domains and projects´ characteristics. In particular, they all differ in relation to:

- geographical locations: software projects coming from Canada, China, Finland, Japan, New Zealand, Italy, United States, etc.;
- number of involved companies;
- observation number: from 10 to 499 observations;
- number and type of features: from 1 to 27 features, including a variety of features describing the software and Web projects, such as number of entities in the data model, number of basic, logical transactions, number of developers involved in the project and their experience, number of Web page or image;
- technical characteristics: software projects developed in different programming languages and for different application domains, ranging from telecommunications to commercial information systems.

Nevertheless, note that none of these datasets are random samples of software and Web projects. Therefore the information provided in Appendix A can be useful for readers to assess whether the results we gathered can scale up to their own contexts.

In order to avoid that large differences in the ranges of the features' values can have the unwanted effect of giving greater importance to some characteristics than to others, a data preprocessing step should be applied when using SVR [17][126]. In our previous studies [24][25], we experimented different preprocessing strategies, such as normalization and logarithmic. The latter is a typical approach in the field of effort estimation [30][15][37][75], since it reduces ranges and at the same time it limits the linearity issue. It provided the best results in [24][25], thus, we adopted it in [26] and in the present paper. Moreover, we removed from the employed datasets the observations which have missing values (see Appendix A).

Table 25. Summary of the employed datasets

	Dataset	Description	Observations	Employed Features
Single-Company	Albrecht [3]	Applications developed by the IBM DP Services organization	24	4
	Desharnais [33]	Software projects derived from a Canadian software house	77	-
	Desharnais1	Projects developed with Language1	44	6
	Desharnais2	Projects developed with Language2	23	6
	Desharnais3	Projects developed with Language3	10	6
	Maxwell [87]	Software projects coming from one of the biggest commercial bank in Finland	62	-
	MaxwellA2	Projects developed for Application2 (i.e., transaction control, logistics, and order processing applications)	29	17
	MaxwellA3	Projects developed for Application3 (i.e., customer service applications)	18	17
	MaxwellS2	Projects developed in outsourcing	54	17
	MaxwellT1	Projects developed using the Telon CASE tool	47	17
	Telecom [120]	Data about enhancement projects for a U.K. telecommunication product.	18	2
Cross-Company	China [109]	Projects developed by Chinese software companies	499	5
	Finnish [121]	Data collected by the TIEKE organizations on projects from different Finnish software companies	38	4
	Kemerer [68]	Data on large business applications collected by a national computer consulting and services firm, specialized in the design and development of data-processing software	15	1
	Miyazaki [102]	Data on projects developed in 20 companies by Fujitsu Large Systems Users Group.	48	3
	Tukutuku [94]	Data about Web hypermedia systems and Web applications coming from several software companies across ten different countries.	195	-
	DocProNo	Projects that did not follow a defined and documented process.	90	15

		DocProYes	Projects that followed a defined and documented process.	105	15
		Enhancement Projects	Projects that are enhancement projects	67	15
		NewProjects	Projects that are new projects	128	15
		MetricYes	Projects whose team was part of a software metrics programme	65	15
		MetricNo	Projects whose team was not part of a software metrics programme	130	15
		ProImprYes	Projects whose team was involved in a process improvement programme	91	15
		ProImprNo	Projects whose team was not involved in a process improvement programme	104	15

5.2.2 Null Hypotheses

To address the first research question (i.e., assessing the effectiveness of TS for configuring SVR) we first verified the benefits of using a search-based approach like TS to configure SVR against a simpler approach considering random configurations (SVRrand, in the following). In this case, to be fair the same number of solutions has to be generated and compared with those achieved with the meta-heuristics search approach. Thus, we randomly generated 25100 SVR configurations ten times (within the same ranges defined for TS in Section 5.2) and the best one of these was selected based on the same criteria employed for SVR+TS but without guiding the search in any way. Moreover, we also considered the use of the default configuration (i.e., $C = 1$, $\varepsilon = 0.001$, $\gamma = 0$) provided by the Weka tool [52] (SVRweka in the following) and the Grid-search algorithm provided by LibSVM [17] (SVRgrid in the following).

As a consequence, the following null hypotheses were formulated:

Hn0: SVR+TS does not provide significant better estimates than SVRrand;

Hn1: SVR+TS does not provide significant better estimates than SVRweka;

Hn2: SVR+TS does not provide significant better estimates than SVRgrid;

which contrast with the following alternative hypotheses:

Hn0: SVR+TS provides significant better estimates than SVRrand;

Hn1: SVR+TS provides significant better estimates than SVRweka;

Hn2: SVR+TS provides significant better estimates than SVRgrid.

With regard to the second research question, we assessed whether the estimates obtained with SVR+TS were better than those obtained using the Manual StepWise Regression (MSWR) [75][97] and the Case-Based Reasoning (CBR) [119] that are two techniques widely used in the literature and also in industry (probably the most employed estimation methods).

MSWR is a statistical technique whereby a prediction model (Equation) is built and represents the relationship between independent (e.g., number of Web pages) and dependent variables (e.g., total Effort). This technique builds the model by adding, at each stage, the independent variable with the highest association to the dependent variable, taking into account all variables currently in the model. It aims to find the set of independent variables (predictors) that best explain the variation in the dependent variable (response).

Within the context of our investigation, the idea behind the use of CBR is to predict the effort of a new project by considering similar projects previously developed. In particular, the completed projects are characterized in terms of a set of *p* features (i.e., variables) and form the *case base* [119]. The new project is also characterized in terms of the same *p* features and it is referred as the *target case*. Then, the similarity between the target case and the other cases in the *p*-dimensional feature space is measured, and the most similar cases are used, possibly with adaptations, to obtain a prediction for the target case. In our empirical study we employed CBR in two ways:

i) by considering only the independent variables that are statistically correlated to the dependent variable (CBRfss in the following), and
ii) without applying any kind of selection of the variables (CBR in the following).

The key aspects of MSWR and CBR are detailed in Appendix B and C, respectively.

In addition, we also assessed whether the estimates obtained with SVR+TS were significantly better than those obtained using the mean of effort (MeanEffort in the following) and the median of effort (MedianEffort in the following). This was done because, as suggested by Mendes and Kitchenham in [97], if an estimation technique does not outperform the results achieved by using MeanEffort and MedianEffort, it cannot be transferred to industry since there would be no value in dealing with complex computations of estimation methods to predict development effort compared to simply using as estimate the mean or the median effort of its own past projects.

Thus, we formulated the following null hypotheses:

Hn3: SVR+TS does not provide significant better estimates than MSWR;

Hn4: SVR+TS does not provide significant better estimates than CBRfss;

Hn5: SVR+TS does not provide significant better estimates than CBR;

Hn6: SVR+TS does not provide significant better estimates than MeanEffort;

Hn7: SVR+TS does not provide significant better estimates than MedianEffort;

which contrast with the following alternative hypotheses:

Ha3: SVR+TS provides significant better estimates than MSWR;

Ha4: SVR+TS provides significantly better estimations than CBRfss;

Ha5: SVR+TS provides significantly better estimations than CBR;

Ha6: SVR+TS provides significantly better estimations than Mean Effort;

Ha7: SVR+TS provides significantly better estimations than Median Effort.

5.2.3 Validation Method

To assess the effectiveness of the effort predictions obtained using the techniques employed herein we exploited a multiple-fold cross validation, partitioning each original dataset into training sets, for model building, and test sets, for model evaluation. This is done to avoid optimistic predictions [13]. Indeed, cross validation is widely used in the literature to validate effort estimation models when dealing with medium/small datasets (e.g., [15]). When applying the multiple-fold cross validation, we decided to use the leave-one-out cross validation on the datasets that have less than 60 observations (i.e., Albrecht, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, Miyazaki, and Telecom). In those cases the original datasets of N observations were divided into N different subsets of training and validation sets, where each validation set had one project. On the other hand, we decided to partition the datasets having more than 60 observations (i.e., China and the 8 splits obtained from the Tukutuku database) into k=10 randomly test sets, and then for each test set to consider the remaining observations as training set to build the estimation model. This choice was

made trying to find a trade-off between computational costs and effectiveness of the validation. The 10 folds for the China datasets are given in Appendix A (Table 33) [2].

5.2.4 Evaluation Criteria

Several accuracy measures have been proposed in the literature to assess and compare the estimates achieved with effort estimation methods [22][72], e.g., Mean of MRE, Median of MRE; Mean of EMRE, Median of EMRE, and Pred(25) (i.e., Prediction at level 25%). Considering that all the above measures are based on the absolute residuals (i.e., the absolute values of differences between predicted and actual efforts) in our empirical analysis we decided to compare the employed estimation techniques in terms of the Median of Absolute Residuals (MdAR), which is a cumulative measure widely employed as the Mean of Absolute Residuals (MAR). We chose to employ MdAR since it is less sensitive to extreme values with respect to MAR [98]. The use of a single summary measure was motivated by the aim to improve the readability of the discussion on the comparison of the analyzed effort estimation methods (that is not confused by the fact that some measures have to be minimized and other maximized). Moreover, to make the comparison more reliable we used, behind this summary measure, also a statistical test. Indeed, to verify if the differences observed using the above measure were legitimate or due to chance, we checked if the absolute residuals obtained with the application of the various estimation techniques come from the same population. If they do, it means that there are no significant differences between the data values being compared. We accomplished the statistical significance test using a nonparametric statistical significance test [72], namely Wilcoxon Signed Rank test, with $\alpha = 0.05$. We decided to use the Wilcoxon test since it is resilient to strong departures from the *t*-test assumptions [21].

5.3 Results and Discussion

Table 26 reports the Median of the Absolute Residuals (MdAR) obtained with each technique for all the employed datasets. Let us recall that the results of TS+SVR reported herein were obtained

[2] We cannot report the 10 folds used for the Tukutuku datasets since the information included in the Tukutuku database are not public available, for confidence reasons.

applying on test set the final configuration provided by TS, namely the one having objective value closest to the mean of the objective values obtained in the 10 executions performed on training set. Notice that for CBR we used 1, 2, and 3 analogies and due to space constraints, only the best results are reported herein. The number of analogies used to obtain each of these best results is specified in Table 26. The details about the application of MSWR and CBR are reported in Appendix B and C, respectively.

In order to provide better readability, all the best results (i.e., the minimum MdAR values) obtained for each dataset across the employed techniques are reported in bold (see Table 26). Table 26 shows that SVR+TS provided the best MdAR values for all the datasets, except for NewProjects, where CBR provided a slightly better result. To quantify how much SVR+TS provided better results than the other employed techniques, for each dataset we calculated the ratio BestSVR/SVR+TS (AvgSVR/SVR+TS, and WorstSVR/SVR+TS, respectively) between the best (the mean, and the worst, respectively) MdAR provided by the other SVR based approaches with the MdAR of SVR+TS. Similarly, we also provided the same ratios (named BestBench/SVR+TS, AvgBench/SVR+TS, and WorstBench/SVR+TS) with respect to the other estimation techniques used as benchmarks. These results are reported in Table 27, together with the median values of these ratios obtained on all the datasets. Thus, we can observe that with respect to the other SVR techniques:

- the error (i.e., MdAR) made using the other SVR technique providing the best estimates is on median about one half (i.e., 1.48) the error made employing SVR+TS;
- the mean of the errors made using the other SVR techniques is on median about twice (i.e., 1.75) the error made employing SVR+TS;
- the error made using the other SVR technique providing the worst result is on median about twice (i.e., 2.06) the error made employing SVR+TS.

As for the comparison with the other estimation techniques used as benchmarks (i.e., MSWR, CBR, MeanEffort, and MedianEffort), the results in Table 27 suggest that:

- the error made using the technique providing the best estimates is on median about twice (i.e., 1.65) the error made employing SVR+TS;
- the mean of the errors made using the other techniques is on median about four (i.e., 3.99) times the error made employing SVR+TS;

- the error made using the technique providing the worst result is on median about nine times (i.e., 8.93) the error made employing SVR+TS.

In order to verify whether the differences observed using MdAR values were legitimate or due to chance, we employed the Wilcoxon test (α= 0.05) to assess if the absolute residuals from all the techniques used came from the same population. The results are reported in Table 28 where "Yes" in a cell means that SVR+TS is significantly superior to the technique indicated on the column (i.e., it means that the absolute residuals achieved with SVR+TS are significantly less than the ones obtained with the technique indicated on the column).

These results allowed us to state that the predictions obtained with SVR+TS were significantly superior than those obtained with SVRrand, SVRweka, SVRgrid, MSWR, CBR (with and without feature selection), MedianEffort, and MeanEffort for all PROMISE and Tukutuku datasets, except for a few cases (i.e., the China, EnhancementProjects, MetricNo, ProImprYes, and ProImprNo datasets with respect to SVRgrid, SVRweka, SVRgrid, CBR, and SVRweka approaches, respectively) where no significant difference was found.

According to these results we can reject all the null hypotheses presented in Section 5.4 (with a confidence of 95%), highlighting that SVR+TS provided significant better estimates than:

- SVRrand for all the datasets;
- SVRweka for 19 out of 21 datasets;
- SVRgrid for 19 out of 21 datasets;
- MSWR for all the datasets;
- CBR for 20 out of 21 datasets;
- CBRss for all the datasets;
- Mean Effort for all the datasets;
- Median Effort for all the datasets.

Thus, we conclude that we can positively answer our research questions RQ7 and RQ8, i.e., Tabu Search was able to effectively set Support Vector Regression parameters and the effort predictions obtained by using the combination of Tabu Search and Support Vector Regression were significantly superior to the ones obtained by other techniques. Note that these results confirm and extend those previously obtained and detailed in [26], thus supporting the usefulness of TS for configuring SVR. Indeed, TS has allowed us to improve the accuracy of the obtained estimates with respect to the use of random configurations, the use of a default configuration, and the use of Grid-

Table 26. Accuracy in terms of MdAR

	Dataset	SVR + TS	SVRrand	SVRweka	SVRgrid	MSWR	CBRfss	CBR	Mean	Median
Single-Company	PROMISE repository									
	Albrecht	**1**	4	3	8	4	5 (k=2)	5 (k=2)	14	7
	Desharnais1	**270**	826	1268	803	1142	1457 (k=3)	1738 (k=2)	2316	1589
	Desharnais2	**490**	668	665	2283	1078	1479 (k=3)	1015 (k=3)	3089	1579
	Desharnais3	**48**	522	302	1013	303	576 (k=3)	483 (k=3)	1007	812
	MaxwellA2	**663**	1752	2533	1993	2159	4250 (k=1)	3221 (k=2)	5598	4166
	MaxwellA3	**1394**	2697	4515	2449	3084	3639 (k=1)	2236 (k=3)	6005	5598
	MaxwellS2	**1489**	1747	2193	1926	1978	16225 (k=2)	2552 (k=3)	5473	3473
	MaxwellT1	**1207**	1852	2488	1981	3005	2857 (k=4)	2906 (k=3)	4462	3196
	Telecom	**20**	80	74	80	59	41 (k=2)	41 (k=2)	156	168
Cross-Company	China	**1032**	1335	1090	1082	1138	1326 (k=3)	1326 (k=3)	2833	1329
	Finnish	**1145**	1776	2318	3194	2602	4170 (k=3)	4919 (k=3)	5909	4774
	Kemerer	**14**	61	66	65	52	56 (k=3)	56 (k=3)	125	74
	Miyazaki	**1175**	2197	2483	1739	1936	1976 (k=3)	1976 (k=2)	8338	2832
	TUKUTUKU repository									
	DocProNo	**26**	41	33	43	38	64 (k=2)	40 (k=2)	287	61
	DocProYes	**33**	54	63	56	63	64 (k=2)	65 (k=2)	533	81
	Enhancement Projects	**17**	24	24	18	24	50 (k=3)	27 (k=3)	192	25
	MetricNo	**36**	57	53	57	65	60 (k=2)	50 (k=2)	532	90
	MetricYes	**12**	24	15	15	14	21 (k=1)	24 (k=2)	189	164
	NewProjects	47	49	60	57	56	57 (k=3)	43 (k=3)	548	87
	ProImprYes	**27**	29	48	36	39	34 (k=3)	28 (k=3)	164	53
	ProImprNo	**41**	52	52	56	49	56 (k=3)	54 (k=1)	646	84

-search algorithm for parameter selection provided by LibSVM. Moreover, we want to stress that the analysis showed that SVR outperformed the two techniques that are to date the most widely and successfully employed prediction techniques in Software Engineering (e.g., [30][13][15][75][95][97][119]), namely MSWR and CBR. In addition, note that SVR+TS outperformed all the other techniques both for single- and cross- company datasets and for both Web-based and not Web-based applications datasets.

Table 27. A comparison between SVR+TS and the other techniques

	Dataset	BestSVR / SVR + TS	AvgSVR / SVR + TS	WorstSVR / SVR + TS	BestBench / SVR + TS	AvgBench / SVR + TS	WorstBench / SVR + TS
	PROMISE repository						
Single Company	Albrecht	3.00	5.00	8.00	4.00	7.00	14.00
	Desharnais1	2.97	3.58	4.70	4.23	6.11	8.58
	Desharnais2	1.36	2.46	4.66	2.07	3.36	6.30
	Desharnais3	6.29	12.76	21.10	6.31	13.25	20.98
	MaxwellA2	2.64	3.16	3.82	3.26	5.85	8.44
	MaxwellA3	1.76	2.31	3.24	1.60	2.95	4.31
	MaxwellS2	1.17	1.31	1.47	1.33	3.99	10.90
	MaxwellT1	1.53	1.75	2.06	2.37	2.72	3.70
	Telecom	3.70	3.90	4.00	2.05	4.65	8.40
Cross Company	China	1.05	1.13	1.29	1.10	1.54	2.75
	Finnish	1.55	2.12	2.79	2.27	3.91	5.16
	Kemerer	4.36	4.57	4.71	3.71	5.48	8.93
	Miyazaki	1.48	1.82	2.11	1.65	2.90	7.10
	Tukutuku repository						
	DocProNo	1.27	1.50	1.65	1.46	3.77	11.04
	DocProYes	1.64	1.75	1.91	1.91	4.88	16.15
	Enhancement Projects	1.06	1.29	1.41	1.41	3.74	11.29
	MetricNo	1.47	1.55	1.58	1.39	4.43	14.78
	MetricYes	1.25	1.50	2.00	1.17	6.87	15.75
	NewProjects	1.04	1.18	1.28	0.91	3.37	11.66
	ProImprYes	1.07	1.40	1.78	1.04	2.36	6.07
	ProImprNo	1.27	1.30	1.37	1.20	4.34	15.76
	Median	1.48	1.75	2.06	1.65	3.99	8.93

Table 28. Comparison of the absolute residuals using Wilcoxon test (p-values are reported between brackets) for PROMISE and Tukutuku datasets

	Dataset	SVR rand	SVR Weka	SVR grid	MSWR	CBR fss	CBR	Median effort	Mean effort
Single-Company	Albrecht	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Desharnais1	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Desharnais2	Yes (<0.01)	Yes (0.046)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Desharnais3	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (0.04)	Yes (<0.01)	Yes (<0.01)	Yes (0.012)	Yes (<0.01)
	MaxwellA2	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	MaxwellA3	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (0.015)	Yes (0.012)	Yes (0.018)	Yes (<0.01)	Yes (<0.01)
	MaxwellS2	Yes (0.047)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	MaxwellT1	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Telecom	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
Cross-Company	China	Yes (<0.01)	Yes (<0.01)	No (0.40)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Finnish	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Kemerer	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Miyazaki	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	DocProNo	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	DocProYes	Yes (<0.01)	Yes (<0.01)	Yes (0.025)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	Enhancement projects	Yes (0.014)	No (0.065)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	NewProjects	Yes (<0.01)	Yes (0.046)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (0.035)	Yes (<0.01)	Yes (<0.01)
	MetricsYes	Yes (<0.01)	Yes (<0.01)	Yes (0.011)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (0.012)	Yes (<0.01)
	MetricsNo	Yes (<0.01)	Yes (0.013)	No (0.111)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)
	ProImprYes	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	Yes (<0.01)	No (0.344)	Yes (<0.01)	Yes (<0.01)
	ProImprNo	Yes (<0.01)	No (0.123)	Yes (0.012)	Yes (<0.01)	Yes (<0.01)	Yes (0.027)	Yes (<0.01)	Yes (<0.01)

5.4 Validity Assessment

We mitigated the construct validity threats arising from the choice of the features and their collection by evaluating the employed estimation methods on publicly available datasets from the PROMISE repository. These datasets have been previously used in many other empirical studies carried out to evaluate effort estimation methods [109]. With respect to the Tukutuku datasets, the size measures and cost drivers used in the Tukutuku database, and therefore in our study, have been obtained from the results of a survey investigation [96], using data from 133 on-line Web forms aimed at giving quotes on Web development projects. In addition, these measures and cost drivers have also been confirmed by an established Web company and a second survey involving 33 Web companies in New Zealand. Consequently, it is our belief that the variables identified are measures that are meaningful to Web companies and are constructed from information their customers can provide at a very early stage in the project development. As for data quality, to identify effort guesstimates from more accurate effort data, companies were asked on how their effort data was collected (see Table 29). At least for 93.8% of Web projects in the Tukutuku database, effort values were based on more than just guesstimates.

Table 29. How effort data was collected

Data Collection Method	# Projects	% Projects
Hours worked per project task per day	81	41.5
Hours worked per project per day/week	40	20.5
Total hours worked each day or week	62	31.8
No timesheets (guesstimates)	12	6.2

In relation to the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Moreover, we used medium size datasets to mitigate the threats related to the number of observations composing the dataset.

As for the external validity, let us observe that both PROMISE and Tukutuku datasets comprise data on projects volunteered by individual companies, and therefore they do not represent random samples of projects from a defined population. This means that we cannot conclude that the results of this study promptly apply to other companies different from the ones that volunteered the data

used here. However, we believe that companies that develop projects with similar characteristics to those included in the Tukutuku and PROMISE database may be able to apply our results to their software projects. However, the adoption of this technique by industry may require to build and calibrate the initial model, prior to its use for effort estimation. This also applies to most effort estimation techniques investigated to date in the literature, and some examples of how to bridge the gap between research and practice are given in [93].

CONCLUSIONS

The main aim of this PhD dissertation was to provide an insight on the use of the search-based (SB) techniques for the effort estimation trying to highlight strengths and weaknesses of these approaches for building effort estimation models and enhancing existing effort estimation techniques.

In particular, the research has been carried out aiming at answer the following questions:

- How the design choices characterizing the use of SB approaches impact on the performance of these techniques?
- Are there any differences in the use of different SB techniques?
- Are SB techniques as effective as widely used effort estimation methods?
- Are SB techniques effective to improve the accuracy of other data-driven effort estimation techniques?

To this end we defined and employed three search-based approaches, namely Hill Climbing (HC), Tabu Search (TS), and Genetic Programming (GP) for software development effort estimation and analyzed their effectiveness with respect to both baseline benchmarks (i.e., Random, Mean Effort, and Median Effort) and widely used estimation methods (i.e., MSWR). The experimentation was performed by considering different settings and objective functions, exploiting seven publicly available datasets (i.e., China, Desharnais, Finnish, Miyazaki, Maxwell, Kemerer, Telecom) and using a 3-fold cross-validation.

The results, evaluated in terms of Sum of Squared Residuals (SSR) and statistical tests, have highlighted that HC is the worst of the three techniques, while TS and GP provided significantly better results than all the baseline benchmarks and can offer estimation models more accurate than those obtained by applying MSWR on almost all the datasets. However, apart from the accuracy, other factors can be relevant for practitioners and crucial for the adoption of a predictive technique in industry, namely transparency of solutions, ease of use, and required resources [16]. As for transparency of solutions, all the proposed search-based approaches and MSWR build an estimation model (i.e., an equation) that makes clear the weight and the contribution of each employed feature, allowing for making easy the solution interpretation. Differently, other techniques (e.g., CBR) give no indication on the contribution of specific features, thus limiting the understating by the user [16].

As for the ease of use, let us observe that to properly apply MSWR a practitioner should have a good statistical background. As a matter of fact, a recent study by Kitchenham *et al.* [71] pointed out that in the literature often invalid regression models have been reported because Linear Regression was applied without taking care of good statistical practices. On the contrary, the use of search-based approaches do not require specific knowledge to be applied since there are no assumptions to be verified on the employed data and also because they are supported by automated tools. However, for using search-based approaches some design choices have to be made and generally different choices may lead to different results. As an example for TS many parameters should be settled (e.g., tabu list size, number of moves and iterations) and such setting in the literature has been generally obtained after a trial-and-error process depending on the problem under investigation. However, in this thesis we have employed an heuristics to set TS and GP which revealed to be effective on all the considered datasets. These findings can allow a practitioner to employ TS or GP overcoming the difficulties related to the set up of these techniques [16].

Concerning the resources required to apply the considered estimation techniques, we can observe that a practitioner has to apply MSWR manually executing the selection process at most n times (n is the number of dataset features), making its use especially time consuming and more error-prone in case of datasets containing a lot of features. On the other hand, the automated tools provided for search-based approaches require in general a longer time to be executed with respect to MSWR, since their execution time depends on both dataset size and the explored search space (i.e., number of moves and iterations).

Finally, we can say that a search-based approach can represent a flexible technique for project managers giving them the possibility to choose the preferred criterion to drive the search for the estimation model. We have empirically analyzed the impact of several existing evaluation criteria as objective function and observed that some of them have the effect to degrade a lot of other criteria. Thus, project managers should be informed on this effect and should take care to select the evaluation criterion as objective function. On the other hand, the research community working in the area of effort estimation is still searching for a suitable and reliable accuracy criterion to assess and compare estimation models (e.g. see [123]). When it will be identified it can be easily exploited as objective function with a search-based technique thanks to the flexibility of these approaches and in case more than one criterion will be identified it could be interesting to investigate multi-objective

optimization approaches (e.g., the ones based on Pareto optimality) [55] to simultaneously optimize several criteria. Moreover, multi-objective optimization has proved to be able to provide decision support for other early-stage software project management activities (see e.g., [47][112]), so its use deserves to be further investigated also in the effort estimation context.

As for the use of search-based approaches to improve other existing estimation techniques, we have assessed whether Support Vector Regression configured by using the proposed Tabu Search approach can be effective to estimate software development effort exploiting 21 datasets (both single- and cross- company datasets related to both Web-based and not Web-based applications). The results of the empirical analysis have highlighted the goodness of TS for configuring SVR. Indeed, SVR+TS provided significant better estimates than SVR configured with simpler approaches, such as random configuration, default configuration provided by the Weka tool, and the Grid-search algorithm provided by LibSVM. Moreover, SVR+TS allowed us to obtain significantly better effort estimates than the ones obtained using MSWR and CBR, two techniques widely employed both in academic and industrial contexts. Many studies reported in the literature show the ability of SVR to construct accurate predictive models in different contexts [18]. Nevertheless, those studies are usually based on the opinion of experts that select SVR parameter values on the basis of their knowledge of both the approach and the application domain [18]. Of course the reliance upon experts severely bounds the practical applicability of this approach in the software industries. The approach investigated in the present paper does not only address the problem to find a suitable SVR setting for effort estimation but it also allows practitioners of software industries to effectively use it without requiring to be an expert in the field of those techniques. These observations together with the results presented in this paper suggest SVR+TS among the techniques that are suitable for software development effort estimation in industrial world.

The good results herein reported concerning the ability of TS to configure SVR motivated us to add to our agenda of future work the use of a TS to improve other estimation techniques, such as Extreme Learning Machine [63], and the use of the proposed approach in other field of Software Engineering, such as fault prediction where some preliminary studies provided encouraging results [36][115].

REFERENCES

[1] Abran, A., Robillard, P.N., Function points analysis: an empirical study of its measurement, 22 (12): 895-910, (1996).

[2] Wasif Afzal, Richard Torkar: On the application of genetic programming for software engineering predictive modeling: A systematic review. Expert Syst. Appl. 38(9): 11984-11997 (2011).

[3] Albrecht AJ, Gaffney JE, Software function, source lines of code, and development effort prediction: A software science validation. IEEE Transactions on Software Engineering 9 (6): 639-648, (1983).

[4] Arcuri, A., Briand, L., A Practical Guide for Using Statistical Tests to Assess Randomized Algorithms in Software Engineering, in Procs of International Conference on Software Engineering, (2011).

[5] Bailey, J.W., Basili, V.R., A Meta Model for Software Development Resource Expenditure, in Procs of Conference on Software Engineering, pp. 107-115, (1981).

[6] Basili, V.R., The Role of Experimentation in Software Engineering: Past, Current and Future, in Procs of International Conference on Software Engineering, (1996).

[7] Basili, V.R., Shull, F., Lanubile, F., Building knowledge through families of experiments, IEEE Transactions on Software Engineering 25(4) (1999), pp. 435–437.

[8] Bernroider, E., Koch, S., ERP selection process in midsize and large organizations, Business Process Management Journal 7(3) (2001), pp. 251–257.

[9] Boehm, B. W., "Software Engineering Economics", Prentice-Hall, Englewood Cliffs, NJ, 1981.

[10] Braga, P.L., Oliveira, A.L.I., Meira, S.R.L, Software Effort Estimation Using Machine Learning Techniques with Robust Confidence Intervals, In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence 1(29-31) (ICTAI 2007), pp.181-185.

[11] Braga, P.L., Oliveira, A.L.I., Meira, S.R.L., A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation, in Procs of the ACM Symposium on Applied computing, pp. 1788-1792, (2008).

[12] Braga, P.L., Oliveira, A.L.I., Ribeiro, G.H.T., and Meira, S.R.L, Bagging predictors for estimation of software project effort, IEEE International Joint Conference on Neural Networks, pp. 1595-1600, (2007).

[13] Briand, L., Wieczorek, I., Software Resource Estimation, Encyclopedia of Software Engineering, pp. 1160–1196, (2002).

[14] Briand, L., Emam, K. El., Surmann, D., Wiekzorek, I., Maxwell, K., An assessment and comparison of common software cost estimation modeling techniques, in Proceedings of International Conference on Software Engineering, IEEE press, 1999, pp. 313–322.

[15] Briand, L., Langley, T., Wiekzorek, I., A replicated assessment and comparison of common software cost modeling techniques, in Proceedings of International Conference on Software Engineering, IEEE press, 2000, pp. 377–386.

[16] Burgess C. J., Lefley M., Can genetic programming improve software effort estimation? A comparative evaluation, Information and Software Technology, pp.863–873, 43, (2001).

[17] Chang, C.-C., Lin, C.-J., LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, (2001).

[18] Cherkassky V, Ma Y, Practical selection of SVM parameters and noise estimation for SVM Regression. Neural Networks, 17(1): 113-126, (2004).

[19] Chiu, N.H., Huang, S., The adjusted analogy-based software effort estimation based on similarity distances, Journal of Systems and Software, pp. 628–640, 80(4) (2007).

[20] Cohen, J., Statistical power analysis for the behavioral sciences, 2nd edition edn. Lawrence Earlbaum Associates, (1988).

[21] Conover, W.J., Practical nonparametric statistics, 3rd ed. Wiley, New York, (1998).

[22] Conte, D., Dunsmore, H., Shen, V., Software engineering metrics and models, The Benjamin/Cummings Publishing Company, Inc., (1986).

[23] Cook, R.D., Detection of influential observations in linear regression, Technometrics, 19: 15-18, (1977).

[24] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E.. Applying support vector regression for web effort estimation using a cross-company dataset. Procs. Empirical Software Engineering and Measurement, pp. 191-202, (2009).

[25] Corazza A., Di Martino S., Ferrucci F., Gravino C., Mendes E., Investigating the use of Support Vector Regression for web effort estimation, Empirical Software Engineering, pp. 211-243,16(2), (2011).

[26] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E., How Effective is Tabu Search to Configure Support Vector Regression for Effort Estimation? in Procs of PROMISE 2010: 4.

[27] Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Sarro, F., Mendes, E., Using Tabu Search to Configure Support Vector Regression for Effort Estimation, Journal of Empirical Software Engineering, http://dx.doi.org/10.1007/s10664-011-9187-3.

[28] Cortes, C., Vapnik, V., Support-vector networks, Machine Learning, 20(3) (1995), pp.273–297.

[29] COSMIC. 2007. Web site, http://www.cosmicon.com

[30] Costagliola, G, Di Martino, S., Ferrucci, F., Gravino, C., Tortora, G., Effort estimation modeling techniques: a case study for web applications. Procs. International Conference on Web Engineering, pp. 9-16, (2006).

[31] Cristianini N, Shawe-Taylor J, An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press, (2000).

[32] Deng, J., Introduction to grey system theory, The Journal of Grey System 1 (1989), pp. 1–24.

[33] Desharnais, J.M., Analyse statistique de la productivitie des projets informatique a partie de la technique des point des function, Unpublished Masters Thesis, University of Montreal, 1989.

[34] Di Geronimo, L., Ferrucci, F., Murolo, A., Sarro, F., A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites, in Proceedings of the 5th Conference on Software Testing, Verification and Validation, Workshop on SBST, pp. 785-793, (2012).

[35] Di Martino, S., Ferrucci, F., Maggio, V., Sarro, F., Towards Migrating Genetic Algorithms for Test Data Generation to the Cloud, in Software Testing in the Cloud: Perspectives on an

Emerging Discipline, Scott Tilley and Tauhida Parveen (Editors), IGI Global, DOI: 10.4018/978-1-4666-2536-5, ISBN13: 9781466625365, (2012).

[36] Di Martino S., Ferrucci F., Gravino C., Sarro F., A Genetic Algorithm to configure Support Vector Machines for Predicting Fault-Prone Components, in Proceedings of the 12th International Conference on Product-Focused Software Development and Process Improvement (PROFES 2011), LNCS Springer vol. 6759, pp. 247-261, ISBN: 978-3-642-21842-2.

[37] Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., Comparing Size Measures for Predicting Web Application Development Effort: A Case Study. Procs. Empirical Software Engineering and Measurement, pp. 324–333, (2007).

[38] Dolado, J. J. , A validation of the component-based method for software size estimation, IEEE TSE, pp. 1006–1021, 26(10), (2000).

[39] Dolado, J.J., On the problem of the software cost function. Information and Software Technology 43(1), 61-72, (2001).

[40] Doval, D., Mancordis, S., B. Mitchell, S., Automatic clustering of software system using a genetic algorithm. In: Proceedings of the 9th International Workshop Software Technology and Engineering Practice, p. 73, (1998).

[41] Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Using Evolutionary Based Approaches to Estimate Software Development Effort, in Evolutionary Computation and Optimization Algorithms in Software Engineering: Applications and Techniques, M. Chis, IGI Global, ISBN13: 9781615208098, (2010).

[42] Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Using Tabu Search to Estimate Software Development Effort, in Procs of IWSM/MENSURA 2009. LNCS, Springer, vol. 5891, pp. 307-320, (2009).

[43] Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Estimating Software Development Effort Using Tabu Search,, in Procs of the 12th International Conference on Enterprise Information Systems, pp. 236-241, 1, (2010).

[44] Ferrucci, F., Gravino, C., Oliveto, R., Mendes, E., Sarro, Investigating Tabu Search for Web Effort Estimation, in Procs of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE Computer Society, pp.350-357, (2010).

[45] Ferrucci, F., Gravino, C., Oliveto, R., Sarro, F., Genetic Programming for Effort Estimation: an Analysis of the Impact of Different Fitness Functions, in Procs of the 2nd International Symposium on SBSE, IEEE Computer Society, pp. 89-98, (2010).

[46] Ferrucci, F., Gravino, C., Sarro, F., How Multi-Objective Genetic Programming is Effective for Software Development Effort Estimation?, 3rd International Symposium on Search Based Software Engineering (SSBSE 2011), Lecture Notes in Computer Science vol. 6956, pp. 274-275.

[47] Ferrucci, F., Harman, M., Ren, J., Sarro, F., Not Going to Take this Anymore: Multi-Objective Overtime Planning for Software Engineering Projects. ICSE 2013, to appear.

[48] Finnie, G. R., Wittig, G. E., Desharnais, J.-M., 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): pp. 281–289, (1997).

[49] Foss, T., Stensrud, E., Kitchenham, B., Myrtveit, I., A simulation study of the model evaluation criterion MMRE, IEEE TSE, 29(11): 985-995, (2003).

[50] Glover, F., Laguna, M., Tabu Search, Kluwer Academic Publishers, Boston, (1997).

[51] Goldberg, D.E., Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[52] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Wit-ten, I.H., The WEKA Data Mining Software: An Update; SIGKDD Explorations, 11(1), (2009).

[53] Harman, M., Clark, J.A., Metrics Are Fitness Functions Too, IEEE METRICS, pp. 58-69, 2004.

[54] Harman, M., Jones, B.F., Search-based software engineering, Information and Software Technology, 43(14): 833–839, (2001).

[55] Harman, M., The Current State and Future of Search-based Software Engineering, in Procs of Future of Software Engineering, pp. 342-357, (2007).

[56] Harman, M., The relationship between search based software engineering and predictive modeling. PROMISE 2010: 1

[57] Harman, M., Hierons, R., Proctor, M., A new representation and crossover operator for search-based optimization of software modularization, in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1351–1358, Morgan Kaufmann Publishers.

[58] Hofmann, T., Schölkopf, B., Smola, A.J., Kernel Methods in Machine Learning, Annals of Statistics, 36: 1171-1220, (2008).

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

[60] Hsu, C-W, Chang, C-C, Lin, C-J, A Practical Guide to Support Vector Classification, available at http://www.csie.ntu.edu.tw/\~{}cjlin/papers/guide/guide.pdf, (2010).

[61] Huang, S.J., Chiu, N.H., Chen, L.W., Integration of the grey relational analysis with genetic algorithm for software effort estimation, European Journal of Operational Research, pp. 898-909, 188(3), (2008).

[62] Huang C.-L., Wang, C.-J., A GA-based feature selection and parameters optimization for support vector machines, Expert Systems with Applications 31(2) (2006), pp. 231–240.

[63] Huang, G.B., Zhou, H., Ding, X., Zhang, R., Extreme learning machine for regression and multi-class classification, IEEE Transactions on Systems, Man, and Cybernetics—Part b: Cybernetics, 42 (2012) 513-529.

[64] ISBSG: www.isbsg.org

[65] Jeffery, R., Ruhe, M., Wieczorek, I., A Comparative Study of Two Software Development Cost Modeling Techniques using Multi-organizational and Company-specific Data. Information and Software Technology, 42: 1009-1016, (2000).

[66] Jørgensen, M., A review of studies on expert estimation of software development effort, Journal of Systems and Software, pp. 37-60, 70(1-2), (2004).

[67] Kampenes, V., Dybå, T., Hannay, J.E., Sjøberg, D.I.K., A Systematic Review of Effect Size in Software Engineering Experiments, Information and Software Technology, pp.1073-1086, 4(11-12), (2007).

[68] Kemerer, C. F., An empirical validation of software cost estimation models, Communications of ACM, 30(5): 416–429, (1987).

[69] Keerthi, S., Efficient tuning of SVM hyper-parameters using radius/margin bound and iterative algorithms. IEEE Transactions on Neural Networks, 13(5): 1225–1229, (2002).

[70] Keerthi, S., Lin, C.-J., Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel, Neural Computation 15: 1667–1689, (2003).

[71] Kitchenham, B., Mendes, M., Why comparative effort prediction studies may be invalid, in Procs of PROMISE 2009: 4.

[72] Kitchenham, B., Pickard, L. M., MacDonell, S. G., Shepperd, M. J., What accuracy statistics really measure, IEE Procs Software, pp. 81-85, 148 (3), (2001).

[73] Kitchenham, B.A., Pfleeger, S.L., Pickard, L.M., Jones, P.W., Hoaglin, D.C, El Emam, K., Rosenberg, J., Preliminary Guidelines for Empirical Research in Software Engineering, IEEE TSE, pp. 721-734, 8 (2002).

[74] Kitchenham, B.A., Mendes, E., Travassos, G.H., Cross versus Within-Company Cost Estimation Studies: A Systematic Review. IEEE Transactions on Software Engineering 33(5): 316-329, (2007).

[75] Kitchenham, B.A., Mendes, E., A Comparison of Cross-company and Single-company Effort Estimation Models for Web Applications. Procs. Evaluation & Assessment in Software Engineering, pp. 47-55, (2004).

[76] Kitchenham, B.A., A Procedure for Analyzing Unbalanced Datasets. IEEE Transactions on Software Engineering 24(4): 278-301, (1998).

[77] Kitchenham, B.A., Pickard, L., Peeger, S., Case studies for method and tool evaluation. IEEE Software 12(4): 52-62, (1995).

[78] Koch, S., Mitlöhner, J., Software project effort estimation with voting rules, Decision Support Systems, pp. 895- 901, 46(4), (2009).

[79] Kocaguneli E, Gay G, Menzies T, Yang Y, Keung JW, When to use data from other projects for effort estimation. Procs. IEEE/ACM international conference on Automated Software Engineering, pp. 321-324, (2010).

[80] Koza, J.R., Genetic Programming. MIT Press (1992)

[81] Kwok, J.T., Tsang, I.W., Linear dependency between ε and the input noise in ε-support vector regression. IEEE Transactions on Neural Networks 14(3): 544-553, (2003).

[82] Lefley, M., Shepperd, M.J., Using genetic programming to improve software effort estimation based on general data sets, in Procs of GECCO, pp. 2477–2487, (2003).

[83] Li, Y.F., Xie, M., Goh, T.N., A study of project selection and feature weighting for analogy based software cost estimation, Journal of Systems and Software, pp.241-252, 82(2), (2009).

[84] Mair, C., Shepperd, M., The consistency of empirical comparisons of regression and analogy-based software project cost estimation. Procs ISESE, pp. 509-518, (2005).

[85] Matson, J.E., Barrett, B.E., Mellichamp, J.M., Software development cost estimation using function points, IEEE Transactions on Software Engineering 20 (4) (1994), pp. 275–287.

[86] Mattera, D., Haykin S., Support vector machines for dynamic reconstruction of a chaotic system. In B. Scholkopf, J. Burges, & A. Smola (Eds.), Advances in kernel methods: Support vector machine. Cambridge, MA, 1999, MIT Press.

[87] Maxwell, Applied Statistics for Software Managers. Software Quality Institute Series, Prentice Hall, 2002.

[88] Maxwell K., Wassenhove, L.S., Dutta, S., Performance Evaluation of General and Company Specific Models in Software Development Effort Estimation. Management Science, 45(6): 787-803, (1999).

[89] Mendes, E., Web Cost Estimation and Productivity Benchmarking, ISSSE, pp. 194-222, (2008).

[90] Mendes, E., Kitchenham, B., Further comparison of cross-company and within-company effort estimation models for web applications, in: Procs of International Software Metrics Symposium, IEEE press, pp. 348-357, 10 (47), (2004).

[91] Mendes, E., Mosley, N., Counsell, S., Investigating Web Size Metrics for Early Web Cost Estimation, Journal of Systems and Software, pp. 157-172, 77 (2), (2005).

[92] Mendes, E., The Use of Bayesian Networks for Web Effort Estimation: Further Investigation. Procs. International Conference on Web Engineering, pp. 203 – 216, 2008.

[93] Mendes, E., Pollino C., Mosley N., Building an Expert-based Web Effort Estimation Model using Bayesian Networks Procs EASE Conference, pp. 1-10, 2009.

[94] Mendes, E., Mosley, N., Counsell, S., Investigating Web Size Metrics for Early Web Cost Estimation. Journal of Systems and Software, 77 (2): 157-172, (2005a).

[95] Mendes, E., Di Martino, S., Ferrucci, F., Gravino, C., Cross-company vs. single-company web effort models using the Tukutuku database: An extended study. Journal of System & Software 81 (5): 673-690, (2008).

[96] Mendes, E., Mosley, N., Counsell S., Investigating early web size measures for web cost estimation Procs. Evaluation and Assessment in Software Engineering, pp. 1–22, (2003a).

[97] Mendes, E., Kitchenham, B.A., Further Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. Procs. IEEE International Software Metrics Symposium, pp. 348-357, (2004).

[98] Mendes, E., Counsell, S., Mosley, N., Triggs, C., Watson, I., A Comparative Study of Cost Estimation Models for. Web Hypermedia Applications. Empirical Software Engineering 8 (23): 163-196, (2003b).

[99] Mendes, E., Mosley, N., Counsell, S., The Need for Web Engineering: An Introduction, Web Engineering. Springer-Verlag, Mendes, E. and Mosley, N. (Eds.), 1-28, (2005b).

[100] Menzies T., Chen Z., Hihn J., Lum K., Selecting best practices for effort estimation, IEEE Transactions on Software Engineering 32 (11) (2006), 883-894.

[101] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E., Equation of state calculations by fast computing machines, Journal of Chemical Physics 2, pp. 1087–1092, (1953).

[102] Miyazaki Y, Terakado M, Ozaki K, Nozaki H (1994) Robust regression for developing software estimation models, Journal of Systems and Software 27 (1): 3-16.

[103] Mitchell, B.S., Mancoridis, S., Using heuristic search techniques to extract design abstractions from source code, in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1375–1382, Morgan Kaufmann Publishers.

[104] Miller, J., Applying meta-analytical procedures to software engineering experiments, Journal of Systems and Software 54 (2000), pp. 29–39.

[105] Montgomery, D. , Peck, E., Vining, G., Introduction to Linear Regression Analysis, John Wiley and Sons, Inc., 1986.

[106] Moser, R., Pedrycz, W., Succi, G., Incremental Effort Prediction Models in Agile Development using Radial Basis Functions, Procs. International Conference on Software Engineering and Knowledge Engineering, pp. 519-522, 2007.

[107] Oliveira, A.L.I., Estimation of software project effort with support vector regression, Neurocomputing, 69 (13-15), 2006, pp. 1749–1753.

[108] Parlos, A.G., Fernandez, B., Atyla, A., Muthusami, J., Tsai, W., An accelerated algorithm for multilayer perceptron networks, IEEE Transactions on Neural Networks 5(3) (1994), pp. 493-497.

[109] PROMISE Repository of empirical software engineering data, http://promisedata.org/repository.

[110] Royston, P. An extension of Shapiro and Wilk's W test for normality to large samples. Applied Statistics 31(2):115-124, (1982).

[111] Rumelhart, D.E., Hinton, G.E., Williams, R.J., Learning internal representations by backpropagation, Parallel Distributed Processing: Explorations in the Microstructers of Cognition 1 pp. 318–362, (1986),.

[112] Saliu, M.O., Ruhe, G., Bi-objective release planning for evolving software systems, in Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE), pp. 105–114, 2007.

[113] Sarro, F., Search-based approaches for software development effort estimation. In: Proceedings of the PROFES Doctoral Symposium, 2011, pp. 38-43.

[114] Sarro, F., Ferrucci, F., Gravino, C. Single and multi objective genetic programming for software development effort estimation. In: Proceedings of the ACM Symposium on Applied Computing, pp. 1221-1226, 2012.

[115] Sarro, F., Di Martino, S., Ferrucci, F., Gravino, C. A further analysis on the use of genetic algorithm to configure support vector machines for inter-release fault prediction, in Proceedings of the 27th Symposium On Applied Computing - SE track (ACM SAC), pp.1215-1220, 2012.

[116] Scholkopf, B., Smola A., Learning with Kernels. MIT Press, 2002.

[117] Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., Vapnik, V., Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers, IEEE Transactions on Signal Processing 45 (11): 2758-2765, (1997).

[118] Shan, Y., Mckay, R. I., Lokan, C. J., Essam, D. L., Software project effort estimation using genetic programming, in Procs of International Conference on Communications Circuits and Systems, IEEE press, pp. 1108–1112, (2002).

[119] Shepperd, M.J., Kadoda, G., Using Simulation to Evaluate Prediction Techniques, Procs. IEEE International Software Metrics Symposium, 2001, pp. 349-358.

[120] Shepperd, M., Schofield, C., Estimating software project effort using analogies. IEEE Transactions on Software Engineering 23 (11) :736-743, (1997).

[121] Shepperd, M., Schofield, C., Kitchenham, B.A., Effort estimation using analogy. Procs. International Conference on Software Engineering, pp. 170-178, (1996).

[122] Shepperd, M., Schofield, C., Estimating software project effort using analogies, IEEE TSE, pp. 736-743, 23(11), (2000).

[123] Shepperd, M.J., MacDonell, S.J., Evaluating prediction systems in software project estimation. Information & Software Technology 54(8): 820-827, (2012)

[124] Shin, M., Goel, A. L., Empirical data modeling in software engineering using radical basis functions, IEEE Transactions Software Engineering, 26(6): 567–576, (2000).

[125] Shukla, K. K., Neuro-genetic prediction of software development effort, Information and Software Technology, pp. 701–713, 42 (10), (2000).

[126] Smola, A.J., Schölkopf, B., A tutorial on support vector regression. Statistics and Computing, 14 (3): 199-222, (2004).

[127] Srinivasan, K., Fisher, D., Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering 21 (2) (1995), pp. 126–137.

[128] Stensrud, E., Myrtveit, I., Human performance estimating with analogy and regression models: an empirical validation. In: Proceedings of International Software Metrics Symposium, pp. 205-213. IEEE press, 1996.

[129] Uysal, M., Estimation of the Effort Component of the Software Projects Using Simulated Annealing Algorithm, in Proceedings of World Academy of Science, Engineering and Technology 31 (2008), ISSN 1307-6884, pp. 258-261.

[130] Vapnik, V., Chervonenkis, A., A note on one class of perceptrons, Automatics and Remote Control 25, (1964).

[131] Vapnik, V., Chervonenkis, A., Theory of Pattern Recognition (in Russian), Nauka, Moscow, (1974).

[132] Vapnik, V., The nature of Statistical Learning Theory, Springer-Verlag, (1995).

[133] Wieczorek I., Ruhe M., How Valuable is Company-Specific Data Compared to Multi-Company Data for Software Cost Estimation? Procs. International Software Metrics Symposium, pp. 237-246, (2002).

[134] Yoo, S., Harman, M., Ur, S., Highly scalable multi objective test suite minimisation using graphics cards. In: Proceedings of the Third international conference on Search-based software engineering, pp. 219-236. Springer-Verlag, Berlin, Heidelberg, 2001.

[135] Wohlin, C., Runeson, P., Host, M., Regnell, B., Wesslen, A., Experimentation in Software Engineering-An Introduction, Kluwer Academic Publishers Norwell, MA, USA, (2000).

Appendix

A. Datasets descriptions

In this appendix we provided further information on the employed datasets from the PROMISE repository and the Tukutuku database. In particular, summary statistics for the employed variables are shown Tables 6, 7, and 8, and each dataset is detailed in the following.

Table 30. Summary statistics for the variables of the datasets extracted from the PROMISE repository

Dataset	Variable	Min	Max	Mean	St Dev
Albrecht	Input	7	193	40.25	36.91
	Output	12	150	47.25	35.17
	Inquiry	0	75	16.88	19.34
	File	3	60	17.38	15.41
	Effort	0.50	105.20	21.88	28.42
China	Input	0	9404	167.1	486.34
	Output	0	2455	113.6	221.27
	Inquiry	0	952	61.6	105.42
	File	0	2955	91.23	210.27
	Interface	0	1572	24.23	85.04
	Effort	26	54620	3921	6481
Desharnais	TeamExp	0	4	2.3	1.33
	ManagerExp	0	4	2.65	1.52
	Entities	7	386	121.54	86.11
	Transactions	9	661	162.94	146.08
	AdjustedFPs	73	1127	284.48	182.26
	Envergure	5	52	27.24	8.6
	Effort	546	2349	4903.95	4188.19
Desharnais1	TeamExp	0	4	2.43	1.39
	ManagerExp	0	7	2.30	1.59
	Entities	7	332	118.30	77.43
	Transactions	33	886	169.52	143.43
	AdjustedFPs	83	1116	277.91	179.73
	Envergure	6	51	29.75	277.91
	Effort	805	23940	5413	4366
Desharnais2	TeamExp	1	4	2.17	1.11
	ManagerExp	1	7	3.09	1.38
	Entities	31	387	137.96	109.95
	Transactions	9	482	166.30	135.46
	AdjustedFPs	62	688	279.91	194.24
	Envergure	5	52	23.30	11.27
	Effort	1155	14973	5095.391	4123.559
Desharnais3	TeamExp	0	4	2	1.56
	ManagerExp	1	4	3.20	1.14
	Entities	38	176	90.40	51.08
	Transactions	97	661	256.10	177.60
	AdjustedFPs	99	698	325.70	216.57
	Envergure	6	43	26.90	13.73
	Effort	546	5880	1685	1631
Finnish	HW	1	3	1.26	0.64
	AR	1	5	2.24	1.5
	FP	65	1814	763.58	510.83
	CO	2	10	6.26	2.73
	Effort	460	25670	7678.29	7135.28
Kemerer	AdiFP	99.3	2306.8	999.14	589.59
	Effort	23.2	1107.31	219.25	263.06
Maxwell	Nlan	1	4	2.55	1.02
	T01	1	5	3.05	1

	T02	1	5	3.05	0.71
	T03	2	5	3.03	0.89
	T04	2	5	3.19	0.70
	T05	1	5	3.05	0.71
	T06	1	4	2.90	0.69
	T07	1	5	3.24	0.90
	T08	2	5	3.81	0.96
	T09	2	5	4.06	0.74
	T10	2	5	3.61	0.89
	T11	2	5	3.42	0.98
	T12	2	5	3.82	0.69
	T13	1	5	3.06	0.96
	T14	1	5	3.26	1.01
	T15	1	5	3.34	0.75
	SizeFP	48	3643	673.31	784.08
	Effort	583	63694	8223.21	10499.90
MaxwellA2	Nlan	1	4	2.41	1.12
	T01	2	5	3.34	0.90
	T02	1	4	3.03	0.68
	T03	2	5	3.10	0.86
	T04	2	5	3.28	0.75
	T05	1	5	3.10	0.82
	T06	1	4	2.86	0.64
	T07	2	5	3.41	0.98
	T08	2	5	3.69	0.97
	T09	3	5	4.17	0.66
	T10	2	5	3.83	0.97
	T11	2	5	3.17	0.89
	T12	2	5	3.79	0.82
	T13	1	5	3.07	0.92
	T14	1	5	3.07	1.03
	T15	2	5	3.45	0.78
	SizeFP	59	3368	687.86	769.84
	Effort	845	63694	9628.86	12946.97
MaxwellA3	Nlan	1	4	2.67	0.97
	T01	2	5	2.89	0.96
	T02	2	5	3.11	0.83
	T03	2	5	3.17	0.92
	T04	2	4	3.17	0.71
	T05	2	4	2.89	0.58
	T06	1	4	2.72	0.75
	T07	1	4	3.17	0.86
	T08	2	5	3.83	0.99
	T09	3	5	4.22	0.55
	T10	2	5	3.50	0.71
	T11	2	5	4.00	0.97
	T12	3	5	3.89	0.58
	T13	1	4	3.00	1.03
	T14	2	5	3.28	1.02
	T15	1	4	3.28	0.83
	SizeFP	48	3643	874.17	1006.22
	Effort	583	39479	9824.44	9555.48
MaxwellS2	Nlan	1	4	2.54	1.00
	T01	1	5	2.89	0.96
	T02	1	5	3.11	0.69
	T03	2	5	2.96	0.89
	T04	2	4	3.22	0.66
	T05	2	4	2.98	0.49
	T06	1	4	2.93	0.67
	T07	1	5	3.15	0.83
	T08	2	5	3.83	0.97
	T09	2	5	4.04	0.73
	T10	2	5	3.61	0.86
	T11	2	5	3.50	0.99
	T12	3	5	3.83	0.50
	T13	1	5	3.11	0.96
	T14	1	5	3.22	1.00
	T15	1	5	3.28	0.63
	SizeFP	48	3643	636.96	821.61
	Effort	583	63694	8347.222	11211.18
MaxwellT1	Nlan	1	4	2.30	0.95
	T01	1	5	3.09	1.00
	T02	2	5	3.13	0.71
	T03	2	5	3.06	0.92

	T04	2	5	3.15	0.75
	T05	1	5	2.98	0.77
	T06	1	4	2.74	0.64
	T07	1	5	3.23	0.91
	T08	2	5	3.87	0.92
	T09	2	5	4.04	0.75
	T10	2	5	3.62	0.92
	T11	2	5	3.21	0.88
	T12	2	5	3.77	0.70
	T13	1	5	3.13	0.95
	T14	1	5	3.34	0.96
	T15	1	5	3.28	0.80
	SizeFP	48	3643	606.77	791.48
	Effort	583	63694	7806.72	10781.81
Miyazaki	SCRN	0	281	33.69	47.24
	FORM	0	91	22.38	20.55
	FILE	2	370	20.55	53.56
	Effort	896	253760	13996	36601.56
Telecom	Changes	3	377	138.06	119.95
	Files	3	284	110.33	91.33
	Effort	23.54	1115.54	284.34	264.71

Table 31. Summary statistics for the variables of the Tukutuku database

Variable	**Min**	**Max**	**Mean**	**Std. Dev**
Nlang	1	8	3.9	1.4
DevTeam	1	23	2.6	2.4
TeamExp	1	10	3.8	2.0
TotWP	1	2,000	69.5	185.7
NewWP	0	1,980	49.5	179.1
TotImg	0	1,820	98.6	218.4
NewImg	0	1,000	38.3	125.5
Fots	0	63	3.2	6.2
HFotsA	0	611	12.0	59.9
Hnew	0	27	2.1	4.7
totHigh	611	611	1	0.0
FotsA	0	38	2.2	4.5
New	0	99	4.2	9.7
totNHigh	0	137	6.5	13.2
TotEff	1.1	5,000	468.1	938.5

Table 32. Summary statistics for variables of the Tukutuku split

Dataset	**Variable**	**Min**	**Max**	**Mean**	**St Dev**
DocProNo	Nlang	1	8	4.17	1.21
	DevTeam	1	6	1.63	0.97
	TeamExp	1	10	5.02	1.77
	TotWP	3	1390	49.07	147.96
	NewWP	0	1333	28.03	140.10
	TotImg	0	780	59.97	107.38
	NewImg	0	583	22.01	66.49
	Fots	0	63	3.58	7.53
	HFotsA	0	611	25.67	86.35
	Hnew	0	8	0.72	1.84
	totHigh	0	611	26.39	86.16
	FotsA	0	38	3.06	6.04
	New	0	99	6.36	13.34
	totNHigh	0	137	9.41	18.60
	TotEff	4	5000	350.90	851.41
DocProYes	Nlang	1	8	3.65	1.59
	DevTeam	1	23	3.39	2.88
	TeamExp	1	10	2.80	1.65
	TotWP	1	2000	86.97	211.94
	NewWP	0	1980	67.99	205.72
	TotImg	0	1820	131.69	276.92
	NewImg	0	1000	52.21	158.61
	Fots	0	21	2.86	4.90
	HFotsA	0	4	0.21	0.57

	Hnew	0	27	3.24	5.95
	totHigh	0	27	3.45	5.93
	FotsA	0	16	1.54	2.47
	New	0	19	2.43	3.78
	totNHigh	0	19	3.97	4.04
	TotEff	1.1	3712	568.58	1000.30
EnhancementProjects	Nlang	1	6	3.15	1.17
	DevTeam	1	15	2.46	1.94
	TeamExp	1	8	2.87	1.60
	TotWP	1	2000	97.51	299.33
	NewWP	0	1980	65.03	289.61
	TotImg	0	1238	100.73	219.81
	NewImg	0	1000	48.46	150.17
	Fots	0	19	1.84	4.17
	HFotsA	0	4	0.37	0.85
	Hnew	0	10	1.19	2.43
	totHigh	0	12	1.57	2.67
	FotsA	0	16	2.72	3.10
	New	0	19	1.58	3.39
	totNHigh	0	19	4.30	4.07
	TotEff	1.1	5000	203.65	634.19
NewProjects	Nlang	1	8	4.27	1.43
	DevTeam	1	23	2.64	2.58
	TeamExp	1	10	4.33	2.05
	TotWP	1	440	54.80	74.02
	NewWP	0	440	41.45	72.40
	TotImg	0	1820	97.46	218.46
	NewImg	0	800	32.94	110.66
	Fots	0	63	3.90	7.00
	HFotsA	0	611	18.02	73.24
	Hnew	0	27	2.54	5.48
	totHigh	0	611	20.56	72.82
	FotsA	0	38	1.99	5.12
	New	0	99	5.63	11.43
	totNHigh	0	137	7.63	15.96
	TotEff	4	3712	606.54	1039.35
MetricsYes	Nlang	1	7	3.18	1.32
	DevTeam	1	23	3.12	3.27
	TeamExp	1	10	2.84	1.79
	TotWP	1	600	55.08	99.97
	NewWP	0	440	31.12	71.85
	TotImg	0	1064	84.14	160.88
	NewImg	0	500	34.69	91.44
	Fots	0	15	1.11	2.79
	HFotsA	0	4	0.22	0.62
	Hnew	0	12	1.23	2.69
	totHigh	0	12	1.45	2.72
	FotsA	0	16	1.89	2.79
	New	0	13	1.66	2.95
	totNHigh	0	16	3.55	3.50
	TotEff	1.1	2768	197.41	461.20
MetricsNo	Nlang	1	8	4.24	1.39
	DevTeam	1	7	2.31	1.72
	TeamExp	1	10	4.32	1.97
	TotWP	3	2000	76.68	216.20
	NewWP	0	1980	58.76	213.17
	TotImg	0	1820	105.81	242.31
	NewImg	0	1000	40.06	139.70
	Fots	0	63	4.23	7.18
	HFotsA	0	611	17.83	72.69
	Hnew	0	27	2.50	5.39
	totHigh	0	611	20.33	72.28
	FotsA	0	38	2.42	5.19
	New	0	99	5.53	11.43
	totNHigh	0	137	7.95	15.82
	TotEff	4	5000	603.46	1078.75
ProImprYes	Nlang	1	7	3.45	1.17
	DevTeam	1	23	2.79	2.93
	TeamExp	1	10	3.23	1.75
	TotWP	1	600	55.89	95.09
	NewWP	0	440	36.52	76.21
	TotImg	0	1238	102.38	199.66
	NewImg	0	800	37.48	111.45
	Fots	0	63	2.43	7.56

	HFotsA	0	4	0.19	0.61
	Hnew	0	12	1.10	2.37
	totHigh	0	12	1.29	2.40
	FotsA	0	38	2.97	5.69
	New	0	99	5.60	13.31
	totNHigh	0	137	8.57	18.23
	TotEff	1.1	2768	192.36	399.99
ProImprNo	Nlang	1	8	4.27	1.57
	DevTeam	1	7	2.39	1.74
	TeamExp	1	10	4.35	2.12
	TotWP	3	2000	81.37	238.20
	NewWP	0	1980	60.95	234.70
	TotImg	0	1820	95.26	234.43
	NewImg	0	1000	38.96	137.10
	Fots	0	21	3.86	4.74
	HFotsA	0	611	22.26	80.73
	Hnew	0	27	2.93	5.93
	totHigh	0	611	25.19	80.14
	FotsA	0	20	1.61	3.09
	New	0	15	3.05	4.17
	totNHigh	0	35	4.65	5.64
	TotEff	4	5000	709.39	1180.34

Albrecht

The Albrecht dataset contains data on 24 applications developed by the IBM DP Services organization with different programming language (i.e., COBOL, PL/I or DMS). We employed as independent variables the four types of external input/output elements (i.e., Input, Output, Inquiry, File) used to compute Function Points [3] and as dependent variable the Effort quantified in person-hours and representing the time employed to design, develop, and test each application. We excluded from the analysis the number of SLOC.

China

The China dataset contains data on 499 projects developed in China by various software companies in multiple business domains. We employed as independent variables the external input/output elements used to calculate Function Points (i.e., Input, Output, Inquiry, File, Interface) and Effort as dependent variable [109].

Desharnais

Desharnais [33] has been widely used to evaluate estimation methods, e.g.,[16][42][120][121]. It contains data about 81, but we excluded four projects that have some missing values, as done in other studies (e.g., [120][121]).

As independent variables we employed: TeamExp (i.e., the team experience measured in years), ManagerExp (i.e., the manager experience measured in years) Entities (i.e., the number of the

entities in the system data model), Transactions (i.e., the number of basic logical transactions in the system), AdjustedFPs (i.e., the adjusted Function Points), and Envergure (i.e., a complex measure derived from other factors defining the environment). We considered as dependent variable the total effort while we excluded the length of the code. The categorical variable YearEnd was also excluded from the analysis as done in other works (e.g., [79][119]) since this not an information that could influence the effort prediction of new applications. The other categorical variable, namely Languages, was used (as done in [79][120]) to split the original dataset into three different datasets Desharnais1 (having 44 observations), Desharnais2 (having 23 observations), and Desharnais3 (having 10 observations) corresponding to Languages 1, 2, and 3, respectively.

Finnish

Finnish contains data on 38 projects from different Finnish companies [121]. In particular, the dataset consists of a dependent variable, the Effort expressed in person-hours, and five independent variables. We decided to do not consider the PROD variable because it represents the productivity expressed in terms of Effort and size (FP).

Kemerer

This dataset contains 15 large business applications, 12 of which were written entirely in Cobol. In particular, for each application the number of both adjusted and raw function points is reported (only AdjFP has been exploited in our study). The Effort is the total number of actual hours expended by staff members (i.e., not including secretarial labor) on the project through implementation, divided by 152.We excluded from our analysis the KSLOC variable which counts the thousands of delivered source instructions, the variable Duration, which represents the project durations in calendar months, and two categorical variables, Software and Hardware, that indicate the software (i.e., Bliss, Cobol, Natural) and the hardware (e.g., IBM 308X, IBM 43XX, DEC Vax) employed in each project, respectively. Note that differently from Desharnais dataset these categorical variables could not be used to create subsets since the resulting sets were too small.

Maxwell

The Maxwell dataset [87] contains data of 62 projects in terms of 17 features: Function Points and 16 ordinal variables, i.e., number of different development languages used (Nlan), customer

participation (T01), development environment adequacy (T02), staff availability (T03), standards used (T04), methods used (T05), tools used (T06), software's logical complexity (T07), requirements volatility (T08), quality requirements (T09), efficiency requirements (T10), installation requirements (T11), staff analysis skills (T12), staff application knowledge (T13), staff tool skills (T14), staff team skills (T15). As done for the Desharnais dataset, we used the categorical variables to split the original dataset. In particular, using the three variables, App, Source, and TelonUse (the former indicates the application type, the second indicates in-house or outsourcing development, and the last indicates whether the Telon CASE tool was employed) we obtained 9 datasets, however only those datasets having a number of observations greater than the feature number were used in our experimentation. In particular, we employed the set of 29 observations having App equals to 2, the set of 18 observations having App equals to 3, the set of 54 observation having Source equals to 2, and the set of 47 observations having TelonUse equals to 1. In the following we refer to these datasets as MaxwellA2, MaxwellA3, MaxwellS2, and MaxwellT1, respectively.

Miyazaki

The Miyazaki dataset is composed by projects data collected from 48 systems in 20 Japanese companies by Fujitsu Large Systems Users Group [102]. We considered the independent variables SCRN (i.e., the number of different input or output screen formats), and FORM (i.e., the number of different form) as done in [102]. The dependent variable is the Effort defined as the number of person-hours needed from system design to system test, including indirect effort such as project management.

Telecom

It includes information on two independent variables, i.e., Changes and Files, and the dependent variable Effort [120]. Changes represents the number of changes made as recorded by the configuration management system and Files is the number of files changed by the particular enhancement project.

Tukutuku

It contains Web hypermedia systems and Web applications. The former are characterized by the authoring of information using nodes (chunks of information), links (relations between nodes),

anchors, access structures (for navigation) and its delivery over the Web. Conversely, the latter represent software applications that depend on the Web or use the Web's infrastructure for execution and are characterized by functionality affecting the state of the underlying business logic. Web applications usually include tools suited to handle persistent data, such as local file system, (remote) databases, or Web Services.

The Tukutuku database has data on 195 projects, where:

- projects came mostly from 10 different countries, mainly New Zealand (47%), Italy (17%), Spain (16%), Brazil (10%), United States (4%), England (2%), and Canada (2%);
- project types are new developments (65.6%) or enhancement projects (34.4%);
- about dynamic technologies, PHP is used in 42.6% of the projects, ASP (VBScript or .Net) in 13.8%, Perl in 11.8%, J2EE in 9.2%, while 9.2% of the projects used other solutions;
- the remaining projects used only HTML and/or Javascript,
- each Web project in the database is characterized by process and product variables [94].

The features characterizing the web projects have the following meaning:

- nlang: Number of programming languages adopted in the project.
- DevTeam: Number of Developers involved in the project.
- TeamExp: Mean number of years of experience for the team members.
- TotWP: Total number of Web pages (new and reused).
- NewWP: Total number of new Web pages.
- TotImg: Total number of images (new and reused).
- NewImg: Total number of new images.
- Fots: Number of features/functions reused without any adaptation.
- HFotsA: Number of reused high-effort features/ functions adapted.
- Hnew: Number of new high-effort features/ functions.
- totHigh: Total number of high-effort features/ functions.
- FotsA: Number of reused low-effort features adapted.
- New: Number of new low-effort features/functions.
- totNHigh: Total number of low-effort features/ functions.
- TotEff: Effort in person-hours (dependent variable).

The Tukutuku database contains also the following categorical variables:

- TypeProj: Type of project (new or enhancement).

- DocProc: If project followed defined and documented process.
- ProImpr: If project team was involved in a process improvement programme.
- Metrics: If project team was part of a software metrics programme.

Table 33. The 10 fold for China dataset

Fold	Project Id
1	10, 42, 61, 65, 82, 87, 91, 99, 102, 105, 118, 120, 128, 144, 155, 156, 162, 168, 187, 201, 216, 223, 266, 275, 278, 286, 288, 289, 290, 291, 306, 312, 325, 343, 350, 358, 363, 391, 397, 399, 404, 420, 437, 441, 446, 449, 450, 455, 471, 476
2	4, 8, 17, 20, 53, 63, 68, 70, 84, 111, 115, 117, 121, 122, 135, 146, 147, 202, 212, 219, 224, 229, 252, 257, 267, 271, 295, 304, 340, 367, 372, 378, 386, 389, 400, 413, 415, 419, 428, 430, 433, 434, 435, 440, 448, 467, 470, 472, 484, 486
3	9, 15, 18, 34, 51, 52, 62, 89, 101, 109, 110, 114, 154, 157, 160, 161, 165, 178, 198, 206, 210, 214, 236, 243, 245, 249, 253, 260, 264, 272, 285, 294, 299, 309, 315, 320, 321, 328, 329, 362, 371, 379, 384, 406, 418, 421, 438, 452, 464, 489
4	5, 7, 11, 45, 47, 58, 64, 66, 69, 74, 75, 78, 79, 80, 138, 170, 179, 186, 196, 204, 207, 217, 220, 221, 239, 248, 274, 276, 305, 323, 333, 336, 338, 339, 342, 348, 357, 366, 374, 376, 394, 402, 407, 426, 436, 447, 454, 460, 465, 495
5	2, 21, 22, 27, 30, 33, 35, 43, 50, 59, 71, 90, 95, 97, 100, 116, 127, 129, 137, 148, 151, 172, 173, 190, 191, 195, 197, 203, 209, 213, 250, 262, 263, 280, 281, 282, 283, 284, 296, 331, 337, 377, 398, 411, 443, 456, 482, 487, 494, 498
6	3, 37, 39, 44, 56, 67, 81, 83, 93, 106, 126, 131, 139, 142, 143, 180, 181, 188, 208, 226, 230, 237, 240, 244, 256, 259, 261, 265, 270, 297, 307, 308, 319, 334, 335, 359, 370, 380, 390, 401, 403, 414, 422, 427, 431, 442, 475, 477, 481, 488
7	6, 26, 32, 41, 48, 49, 54, 60, 72, 104, 130, 132, 140, 141, 150, 152, 159, 163, 166, 169, 174, 184, 192, 194, 200, 211, 222, 231, 242, 279, 300, 310, 313, 314, 324, 330, 344, 347, 351, 354, 381, 393, 405, 417, 457, 462, 474, 491, 492, 496
8	1, 12, 13, 14, 16, 19, 25, 28, 31, 77, 124, 125, 134, 145, 149, 167, 171, 177, 183, 185, 199, 205, 215, 233, 235, 251, 255, 258, 269, 277, 292, 298, 303, 311, 341, 364, 368, 382, 385, 409, 410, 444, 451, 463, 466, 469, 473, 479, 497, 499
9	23, 24, 36, 38, 46, 57, 73, 76, 96, 107, 108, 136, 153, 158, 176, 182, 193, 227, 228, 232, 234, 238, 241, 246, 247, 254, 273, 293, 301, 317, 318, 326, 332, 346, 352, 365, 369, 375, 392, 412, 416, 458, 461, 468, 478, 480, 483, 485, 490, 493
10	29, 40, 55, 85, 86, 88, 92, 94, 98, 103, 112, 113, 119, 123, 133, 164, 175, 189, 218, 225, 268, 287, 302, 316, 322, 327, 345, 349, 353, 355, 356, 360, 361, 373, 383, 387, 388, 395, 396, 408, 423, 424, 425, 429, 432, 439, 445, 453, 459

B. Manual Stepwise Regression

We applied MSWR using the technique proposed by Kitchenham [76]. Basically the idea is to use this technique to select the important independent variables according to the R^2 values and the significance of the model obtained employing those variable, and then to use linear regression to obtain the final model.

In our study we employed the variables shown in Tables 6, 7, and 8 during cross validation and we selected the variables for the training set of each split by using the MSWR procedure. In particular, at the first step we identified the numerical variable that had a statistically significant effect on the variable denoting the effort and gave the highest R^2. This was obtained by applying simple regression analysis using each numerical variable in turn. Then, we constructed the single variable regression equation with effort as the dependent variable using the most highly (and significantly) correlated input variable and calculated the residuals. In the subsequent step we correlated the residuals with all the other input variables. We continued in this way until there were no more input variables available for inclusion in the model or none of the remaining variables were significantly correlated with the current residuals [76]. At the end of the procedure, the obtained variables were used to build the estimation model for the considered training set, which was then used to obtain the estimates for the observations in the corresponding validation set.

It is worth mentioning that whenever variables were highly skewed they were transformed before being used in the MSWR procedure. This was done to comply with the assumptions underlying stepwise regression [87] (i.e. residuals should be independent and normally distributed; relationship between dependent and independent variables should be linear). The transformation employed was to take the natural log(Ln), which makes larger values smaller and brings the data values closer to each other [71]. A new variable containing the transformed values was created for each original variable that needed to be transformed. In addition, whenever a variable needed to be transformed but had zero values, the Ln transformation was applied to the variable's value after adding 1.

To verify the stability of each effort estimation model built using MSWR, the following steps were employed [71][75]:

- Use of a residual plot showing residuals vs. fitted values to investigate if the residuals are randomly and normally distributed.
- Calculate Cook's distance values [23] for all projects to identify influential data points. Any projects with distances higher than $3 \times (4/n)$, where n represents the total number of projects, are immediately removed from the data analysis [75]. Those with distances higher than 4/n but smaller than $3 \times (4/n)$ are removed to test the model stability by observing the effect of their removal on the model. If the model coefficients remain stable and the adjusted R^2 (goodness of fit) improves, the highly influential projects are retained in the data analysis.

C. Case-Based Reasoning

To apply CBR we have to choose the similarity function, the number of analogies to pick the similar projects to consider for estimation, and the analogy adaptation strategy for generating the estimation. Moreover, also relevant project features could be selected.

In our case study, we applied CBR by employing the tool ANGEL [120] that implements the Euclidean distance which is the measure used in the literature with the best results [98]. As for the number of analogies, we used 1, 2, and 3 analogies, as suggested in other similar works [15][97]. Moreover, to select similar projects for the estimation, we employed as adaptation strategies the mean of *k* analogies. Regarding the feature selections, we considered the independent variables that are statistically correlated to the effort (at level 0.05), obtained by carrying out a Pearson correlation test [92] on the training set of each split. We did not use feature subset selection of ANGEL since it might be inefficient, as reported in [14][120]. In addition, all the project attributes considered by the similarity function had equal influence upon the selection of the most similar project(s). We also decided to apply CBR employing all the variables of Table 1 as set of features, as done for the application of SVR+TS, considering all relevant factors for designers and developers. In the paper we distinguish between the two different applications of CBR, using CBRfss for denoting the use of the method with feature selection.