Using Tabu Search to Estimate Software Development Effort

Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, and Federica Sarro

Dipartimento di Matematica e Informatica, University of Salerno
Via Ponte Don Melillo, I-84084 Fisciano (SA), Italy
{fferrucci, gravino, roliveto, fsarro}@unisa.it

Abstract. The use of optimization techniques has been recently proposed to build models for software development effort estimation. In particular, some studies have been carried out using search-based techniques, such as genetic programming, and the results reported seem to be promising. At the best of our knowledge nobody has analyzed the effectiveness of Tabu search for development effort estimation. Tabu search is a meta-heuristic approach successful used to address several optimization problems. In this paper we report on an empirical analysis carried out exploiting Tabu Search on a publicity available dataset, i.e., Desharnais dataset. The achieved results show that Tabu Search provides estimates comparable with those achieved with some widely used estimation techniques.

Keywords: Effort Estimation, Empirical analysis, Search-based approaches, Tabu Search.

1 Introduction

Several methods have been proposed in the literature to estimate software development effort. Many of them, named Model-Based, exploit data from past projects in order to estimate the effort for a new project under development [3,4,41]. These data consist of some relevant factors of the software projects, named cost drivers, and the actual effort spent to develop the projects. In this class, we can find some widely used techniques, such as Linear and Stepwise Regression (LR and SWR), Classification and Regression Tree (CART), and Case-Based Reasoning (CBR) [5].

In the last years, some researchers have analyzed the use of genetic algorithms [18] to address the effort estimation problem, reporting results which encourage further investigations (e. g., [7,15,32]).

Genetic algorithms are search-based approaches that exploit techniques inspired by evolutionary biology to address optimization problems [22]. Indeed, effort estimation can be seen as an optimization problem, where we have to search for the most accurate estimate, i.e. the one that minimizes the difference with the actual effort. There exist other search-based techniques that have been found to be very effective and robust in solving numerous optimization problems. In particular, Tabu search is

© Springer-Verlag Berlin Heidelberg 2009

an approach that has been applied to a wide range of application domains ranging from telecommunication and transport, to network design and parallel computing [17]. As for software engineering Tabu Search has been successfully applied for software testing [1,12,13,14], for object replication in distributed web server [34] and for Software-Hardware Partitioning [31]. To the best of our knowledge nobody has analyzed the effectiveness of Tabu search for development effort estimation. Thus, in this paper we report on an empirical analysis carried out by applying Tabu Search on a publicity available dataset, i.e., Desharnais dataset [11]. In particular, the specific contributions of this work are:

- the definition of a Tabu Search algorithm for effort estimation;
- the analysis of the estimation accuracy of the proposed approach;
- the comparison of the effectiveness of the proposed approach with widely used estimation methods, i.e. SWR and CBR;

The remainder of the paper is organized as follows: Section 2 provides a brief description of the Tabu Search approach and presents the Tabu Search algorithm we used to estimate development effort. The design of the case study we performed is summarized in Section 3, while the results of the performed empirical analysis are presented in Section 4. Related works are described in Section 5 while some final remarks and an analysis of future work conclude the paper.

2 Using Tabu Search for Software Development Effort Estimation

In the following we first provide a brief description of the Tabu Search and then we give some details of the Tabu Search algorithm we designed for effort estimation.

2.1 Tabu Search

The Tabu Search (TS) is an optimization method proposed originally by Glover aiming at overcome some limitations of Local Search (LS) heuristics [17].

As in classical LS, a general step of the TS optimization process consists in constructing from a current solution *i* a next solution *j* and in checking whether one should stop there or perform another step. Indeed, TS is a neighbourhood search method in which a neighbourhood *N(i)* is defined for each feasible solution *i*, and the next solution *j* is searched among the solutions in *N(i)*.

In contrast to traditional LS heuristics, Tabu Search is based on the premise that problem solving, in order to qualify as intelligent, must incorporate adaptive memory and responsive exploration [17]. The adaptive memory feature of TS allows the implementation of procedures that are capable of searching the solution space economically and effectively. Since local choices are guided by information collected during the search, TS contrasts with memory-less designs that heavily rely on semi-random processes that implement a form of sampling. Examples of memory less methods include semi-greedy heuristics and the prominent genetic and simulated annealing algorithms. The emphasis on responsive exploration in TS derives from the supposition that a bad strategic choice can yield more information than a good random choice. Responsive exploration integrates the basic principles of intelligent search, i.e., exploiting good solution features while exploring new promising regions.

More particularly, the Tabu search begins by marching to a local minima and records recent moves in one or more Tabu lists, marking these moves as taboo. Such information is useful to avoid retracing the steps previously used. It is worth noting that the aim of the tabu list is not to prevent a previous move from being repeated, but rather to insure it is not reversed. Since taboo sometimes may prohibit attractive moves or they may lead to an overall stagnation of the searching process [16], aspiration criteria are used to revoke the tabu status of a move. The searching process will terminate when a stopping condition is met.

Thus, for the application of TS several key issues have to be addressed [17]:

- defining a representation of possible solutions;
- defining the neighbourhood;
- choosing a means to evaluate the neighbourhood (e.g. an objective function);
- defining the Tabu list, the aspiration criteria, the termination criteria.

The following section gives some details on the design of the proposed TS for effort estimation.

2.2 Design of the Proposed Tabu Search Algorithm

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

$$\mathit{Effort} = c_1\ op_1\ f_1\ op_2\ \ldots\ op_{2n-2}\ c_n\ op_{2n-1}\ f_n\ op_{2n}\ C \tag{1}$$

where f_i represents the value of the ith factor and c_i is its coefficient, C represents a constant, while $op_i \in \{+,-,\cdot\}$ represents the ith operators of the model. Obviously, to be feasible for our problem, the *Effort* value has to be positive.

So, the search space of TS is simply the space of all feasible equations that can be generated changing the values of c_i, C, and op_i during each move. To avoid a restriction of the search space, an initial feasible solution is generated selecting randomly the values for coefficients and constants and the arithmetic operators. Starting from this solution, TS at each iteration applies local transformations to the current solution, i.e. moves, defining a set of neighboring solutions in the search space. For our problem we defined a neighbourhood to a given solution S as any other solution that is obtained by a random variation of the equation, $expr_S$, representing the current solution. In particular, a *move* consists in three steps:

1. change each coefficient c_i of $expr_S$ with probability ½; the new coefficient is calculated by applying an arithmetic operator, chosen randomly in the range {+, *, - , /}, to c_i and a number r, chosen randomly in the range]0,1];
2. change the constant factor C of $expr_S$ with probability ½ in the same way coefficients are changed;
3. change each arithmetic operators op_i of $expr_S$ with probability ½ by selecting another operator in $\{+,-,\cdot\}$.

It is worth noting that in the definition of the move we do not consider the values of the factors, as they are constant and do not change during the searching.

Once the neighbourhood of a solution is obtained we have to compare the current best solution *expr* with the qualities of each $expr_i$, in order to decide whether or not a move to a neighbouring solution has to be performed. Since in the effort estimation context we would like to select the prediction model which minimize the error made in estimating the effort, to evaluate the quality of a solution over a set of data projects we employed an objective function which minimize the value of the Mean of Magnitude of Relative Error (MMRE) [10]. MMRE is one of the widely used summary measures proposed in the literature to evaluate the accuracy of an estimation model (the definition is reported in Section 3.2).

Thus, when performing a move, the algorithm has to consider the improvement of the current solution in terms of MMRE. If the MMRE value achieved by an $expr_i$ is less then the one achieved by the current best solution *expr*, this is replaced by $expr_i$ which will be used in the next iteration to explore a new neighbourhood; otherwise, the search continues by generating other moves starting from *expr*.

To avoid loops and to guide the search far from already visited portions of the search space, the recently visited solutions are marked as taboo and stored in a Tabu list. Since only a fixed and fairly limited quantity of information is usually recorded in the Tabu list [16], we prohibit the use of a taboo move for ten iterations. Thus, at each iteration the Tabu list contains at most ten taboo equations. In order to allow one to revoke taboo we employed the most commonly used *aspiration criterion*, namely we permit a taboo move if it results in a solution with an objective function value (i.e. the MMRE value) better than the one of the current best-known solution.

The search is stopped after a fixed number of iterations or after some number of iterations that do not provide an improvement in the objective function value.

To implement the proposed TS algorithm a Java application based on the OpenTS framework [38] has been realized. For our analysis the application was executed exploiting a 1.4Ghz Pentium machine with 1Gb Ram.

3 Experimental Method

This section presents the design of the case study we carried out to assess the effectiveness of the proposed TS in estimating software development effort. The *goals* of the empirical investigation were:

- analyzing the effectiveness of TS in estimating software development effort;
- comparing the estimates achieved by applying TS with the estimates obtained with widely and successfully employed estimation methods.

Regarding the former research goal to evaluate the accuracy of the obtained estimates we employed widely used summary measures, namely *MMRE*, *MdMRE*, and *Pred*(25) [10] whose definitions are reported in Section 3.2.

As for the second research goal, we compared the TS estimations with those obtained by using two widely used methods, i.e., Stepwise regression [26,28] and Case-Based Reasoning [23].

3.1 Dataset and Feature Selection

In our case study we exploited an existing dataset comprising 81 software projects. This dataset was derived from a Canadian software house in the late 1980s by Jean-Marc Desharnais [11]. Despite of this dataset is about 30 years old, it is one of the larger, publicly available datasets and it has been widely and recently used to evaluate and compare estimation methods, see e.g., [7,23,41,42].

Table 1 reports on the description of the eleven variables (nine independent and two dependent) included in the dataset. It is worth nothing that categorical (or nominal) variables (i.e., Language and YearEnd) were excluded from the analysis, as done in other works (e.g. [23]). We could handle each categorical variable as in classical regression analysis by transforming it into a set of *n - 1* dummy variables, where *n* is the number of distinct categories in the nominal scale. However, the use of categorical variables to partition the dataset is not particularly attractive, especially when a categorical variable can assume a lot of values or there are a large number of these variables [23]. So, we preferred eliminated them from the analysis. Moreover, we also excluded the LOC variable, since this information is not available at prediction time but it is only known once the software is completed [41]. Moreover, we excluded from the analysis four projects that had missing values. It is worth noting that the same choice has been done in most other studies, see e.g., [23,41]. Table 2 reports on the descriptive statistics of each selected factor.

Table 1. Project features of the Desharnais dataset

Variable	Description	Type
TeamExp	The team experience measured in years	Discrete
ManagerExp	The manager experience measured in years	Discrete
Entities	The number of the entities in the system data model	Discrete
Transactions	The number of basic logical transaction in the system	Discrete
AdjustedFPs	The adjusted Function Points	Continuous
RawFPs	The raw Functions Points	Continuous
Envergue	A complex measure derived from other factors defining the environment	Discrete
Language	The language used to develop the system	Categorical
YearEnd	The project year finisched	Discrete
Effort	The actual effort measured in person hours (dependent variable)	Discrete
Length	The length of the code (dependent variable)	Discrete

Table 2. Descriptive statistics of the factors selected from the dataset

Variable	Min	Max	Mean	Std. Dev.
TeamExp	0.00	4.00	2.30	1.33
ManagerExp	0.00	7.00	2.65	1.52
Entities	7.00	387.00	120.55	86.11
Transactions	9.00	886.00	177.47	146.08
AdjustedFPs	73.00	1127.00	298.01	182.26
RawFPs	62.00	1116.00	282.39	186.36
Envergue	5.00	52.00	27.45	10.53
Effort	546.00	23490.00	4903.95	4188.19

3.2 Validation Method and Evaluation Criteria

In order to verify whether or not the selected method gives useful estimations of the actual development efforts a validation process is required. For this reason, we performed a "hold-out validation" approach, thus a validation based on the use of a hold-out sample of applications [29]. In particular, we randomly split the original Desharnais dataset obtaining two datasets (i.e., training and test sets) composed of 59 (about 3/4 of the original dataset) and 18 (about 1/4 of the original dataset) observations, respectively.

Concerning the evaluation of the estimation methods, we performed a preliminary analysis by using some summary measures, namely MMRE, MdMRE, and Pred(25) [10]. 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* [10] 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.

The *Prediction at level l* [10] 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.* [10], 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.

To have a better visual insight on the effectiveness of the estimation models, we compared the prediction accuracies taking into account both the summary statistics both the boxplots of absolute residuals, where residuals are calculated as (EFreal – EFpred). Boxplots are widely employed in exploratory data analysis since they provide a quick visual representation to summarize the data, using five values: median, upper and lower quartiles, minimum and maximum values, and outliers [25]. The box of the plot is a rectangle with an end at each quartile and a line is drawn across the box at the sample median (m in Figure 1). The lower quartile (l in Figure 1) is determined considering the bottom half of the data, below the median, i.e., by finding the median of this bottom data. While, the upper quartile (u in Figure 1) is the median of the upper half of the data, above the median. The length of the box d=u-l is the inter-quartile range of the statistical sample. Lower tail is l–1.5*d while u+1.5*d is the upper tail. Points at a distance from the median greater than 1.5 times the inter-quartile range represent potential outliers and are plotted individually. In development effort estimation, boxplots are used to visually represent the amount of the error for a given prediction technique.

We use boxplot to graphically render the spread of the absolute residuals, calculated as |EFreal – EFpred|.

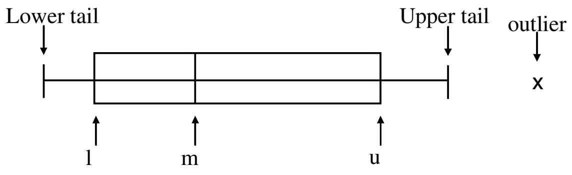

Fig. 1. A boxplot

In order to verify whether the estimates obtained with TS are characterized by significantly better accuracy than the considered benchmarks we statistically analyzed the absolute residuals, as suggested in [25]. Since (i) the absolute residuals for all the analyzed estimation methods were not normally distributed (as confirmed by the Shapiro test [39] for non-normality), and (ii) the data was naturally paired, we decided to use the Wilcoxon test [9]. The achieved results were intended as statistically significant at $\alpha = 0.05$.

3.3 Validity Evaluation

It is widely recognized that several factors can bias the validity of empirical studies. In this section we discuss on the validity of the empirical study based on three types of threats:

- *Construct validity*, related to the agreement between a theoretical concept and a specific measuring device or procedure;
- *Conclusion validity*, related to the ability to draw statistically correct conclusions;
- *External validity*, related to the ability to generalize the achieved results.

As highlighted by Kitchenham *et al.* [27], in order 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 [35]. Thus, the choice of the features and how to collect them represents the crucial aspects. We tried to mitigate such a threat by evaluating the proposed estimation methods on a reliable project data coming from the industrial world [11]. Moreover, since the dataset is publicly available it has been previously used in many other empirical studies carried out to evaluate effort estimation methods, e.g., [7,23,41].

Concerning the conclusion validity we carefully applied the statistical tests, verifying all the required assumptions. Moreover, we used a medium size dataset in order to mitigate the threats related to the number of observations composing the dataset. Nevertheless, the projects involved in this empirical analysis are representative samples of projects conducted by one software house. Thus, the projects are related to one context. Indeed, each context might be characterized by some specific project and human factors, such as development process, developer experience, application domain, tools, technologies used, time, and budget constraints [6]. This represents an important external validity threat that can be mitigated only replicating the study taking into account data from other companies. Indeed, this is the only way to get a generalization of the results.

4 Results and Discussion

In the following we report on the results achieved in the empirical study carried out in order to assess the use of Tabu Search in estimating development effort.

Since no case study has been conducted so far on the effectiveness of Tabu Search in building an effort prediction model we exploited a variety of parameter settings to find suitable value for moves and iterations numbers. Concerning the number of moves, we executed TS using three different values, i.e., 500, 1000, and 2000. The best results were achieved considering 1000 moves. We also executed the algorithm considering different number of iterations, and the best results were achieved using 3000 iterations. They are reported in Table 3 providing values for summary measures MMRE, MdMRE, and Pred(25). As we can see the thresholds provided in [10] are not satisfied, since Pred(25) values are less than 0.75 and MMRE (and MdMRE) values are greater than 0.25. In order to have an insight on these results and understand the actual effectiveness of TS for this dataset it is important to compare TS estimation accuracy with the ones of some widely used techniques, such as SWR and CBR, thus addressing our second research goal.

Table 3. The results in terms of MMRE, MdMRE, and Pred(25)

	MMRE	**PRED(25)**	**MdMRE**
TS	0.45	0.39	0.43
CBR	0.48	0.55	0.22
SWR	0.39	0.22	0.38

The results obtained by applying SWR and CBR on the same dataset are also reported in Table 3. First of all, we can observe that for all the employed estimation techniques the thresholds provided in [10] are not satisfied since Pred(25) values are less than 0.75 and MMRE (and MdMRE) values are grater than 0.25 (except for CBR having a MdMRE equals to 0.22). As for the comparison with SWR we can note that TS is characterized by a better Pred(25) values and slight worse MMRE and MdMRE values. Regarding the comparison with CBR, TS achieved a slightly better MMRE value but worse MdMRE and Pred(25). The analysis of these results does not provide a clear indication of what estimation methods, between TS, SWR, and CBR provide the best results. These considerations are confirmed by the boxplots in Figure 2, highlighting that TS has a median very close to the median of SWR and CBR. Furthermore, observe that SWR has three outliers but its box length and tails are less skewed than those of CBR and TS. The tails and box length of TS and CBR are very close even the boxplot of TS has one outlier.

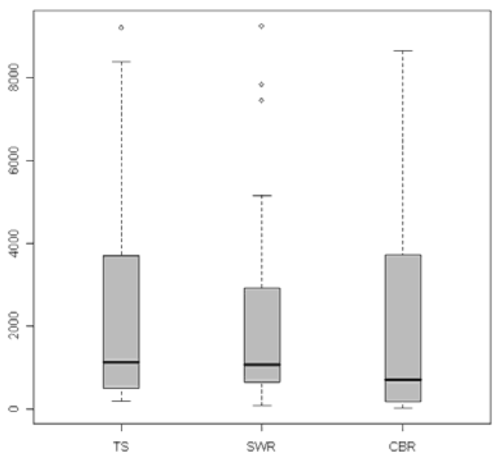

Fig. 2. The boxplots of absolute residuals

As designed we tested the statistical significance of the results obtained by comparing paired absolute residuals. The results of the Wilcoxon tests are reported in Table 4 where "<" means that "the estimation method indicated on the row provided significantly less absolute residuals than the estimation method on the column". As we can see, the Wilcoxon test revealed that there is no statistical significant difference between the absolute residuals obtained with TS, CBR, and SWR. Thus, for this dataset TS seems to have comparable performance with respect to two widely used estimation techniques.

Table 4. The results of Wilcoxon tests in terms of p-values

<	TS	SWR	CBR
TS	-	0.808	0.925
SWR	0.204	-	0.831
CBR	0.082	0.178	-

5 Related Work

To the best of our knowledge Tabu Search was never used for estimating software development effort while some empirical investigations were performed to assess the effectiveness of genetic algorithms in estimating software development effort. In particular, Dolado [15] employed 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 the Mean Squared Error[1] (MSE) [10], as fitness function. As dataset, 46 projects developed by academic students 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 [7] assessed the use of genetic algorithm for estimating software development effort in a case study that exploited the Desharnais dataset [11]. The settings they used for the employed genetic algorithm are: an initial population of 1000, 500 generations, 10 executions (i.e., run), and a fitness function designed to minimize MMRE. They obtained an MMRE = 0.45, which is close to the MMRE we obtained with Tabu search, and a Pred(25) = 0.23, which is worse than the one we obtained with Tabu Search. Even if GP did not outperform LR and ANN (Artificial Neural Networks) the results were promising and Burgess and Lefley suggested that a better set up of the genetic algorithm could improve the accuracy of the estimations.

Successively, Lefley and Shepperd [32] 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 [7], 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 obtained applying a Hold-out validation and used in the empirical analysis. Even if the results revealed that there was not a method that provides better estimations than the others, the evolutionary approach performed consistently well.

An evolutionary computation method, named Grammar Guided Genetic Programming (GGGP), was proposed in [40] to fit models, with the aim of improving the estimation of the software development effort. Data of software projects from ISBSG [22] database was used to built the estimation models using GGGP and LR. The fitness function was designed to minimize the Mean Squared Error (MSE), an initial population of 1000 was chosen, the maximum number of generations was 200, and the number of executions was 5. The results revealed that GPPP performed better than Linear Regression in terms of MMRE and Pred(25).

Finally, it is worth to mention that in the literature some proposals can be found that combine search-based approaches with other existing model-based techniques. Also in those cases, the investigations that have been carried out provide promising results, suggesting that the use of such approaches can improve estimation accuracy of traditional techniques [2,8,20,30,33,43].

---

[1] MSE is defined as $MSE=\frac{1}{n}\sum_{i=1}^{n}(EFreal-EFpred)^2$ 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.

6 Conclusions and Future Work

In this paper, we have provided a preliminary analysis of the use of Tabu Search for estimating development effort and compared the accuracy of the obtained estimates with those achieved by widely used estimation techniques, such as SWR and CBR. The empirical investigation was performed by exploiting a publicity available dataset, i.e., Desharnais dataset [11]. Although the configuration adopted for TS is not able to meet the Conte's thresholds, the results are promising since the summary measures MMRE, MdMRE, and Pred(25) related to TS estimates are comparable with those obtained with SWR and CBR. Moreover, SWR and CBR do not significantly outperform TS (and vice versa), as shown by the statistical significance tests obtained by comparing paired absolute residuals with Wilcoxon tests. It is worth to stressing that our analysis is preliminary but it has been useful to understand that further investigations deserve to be carried out (possibly with other datasets) to analyze the usefulness of TS in the context of effort estimation and more in general to investigate the effectiveness of search-based approaches. In particular, as highlighted also by [7,20], specific empirical studies should be performed to understand the role played by different configurations of search-based approaches to improve the obtained estimates. These configurations could be based on:

- other fitness/objective functions. Indeed, the choice of fitness function could influence the achieved results [19], particularly when the measure used by the algorithm to optimize the estimates is the same used to evaluate the accuracy of them [7];
- aggregated fitness or Pareto optimality to consider a multi-objective optimization [19];
- an interactive optimization approach where users can influence the evaluation of fitness [19].

It would be also interesting to perform studies comparing different search-based techniques (i.e. Genetic Algorithms, Simulated Annealing [36], Tabu Search) since they have many similarities, but also distinguishing features. At the same time other combinations of search-based approaches with existing estimation techniques could be explored. For example, Tabu Search (or other search-based techniques) could be exploited to optimize crucial steps in the application of the CBR, such as the feature subset selection.

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

Acknowledgments

We want to thank Alessio Niespolo for his contribution in the implementation of the Tabu Search algorithm.

The research has also been funded by University of Salerno's Finanziamento Medie e Grandi Attrezzature (2004) for the Software Engineering Research Laboratory.

References

1. Blesa, M.J., Xhafa, F.: A Skeleton for theTabu Search Metaheuristic with Applications to Problems in Software Engineering
2. 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: Proceedings of the ACM symposium on Applied computing, pp. 1788–1792 (2008)
3. Briand, L., El Emam, K., 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, pp. 313–322. IEEE Press, Los Alamitos (1999)
4. 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, pp. 377–386. IEEE Press, Los Alamitos (2000)
5. Briand, L.C., Wieczorek, I.: Software resource estimation. Encyclopedia of Software Engineering, 1160–1196 (2002)
6. Briand, L.C., Wust, J.: Modeling Development Effort in Object-Oriented Systems Using Design Properties. IEEE Transactions on Software Engineering 27(11), 963–986 (2001)
7. Burgess, C.J., Lefley, M.: Can genetic programming improve software effort estimation: a comparative evaluation. Information and Software Technology 43(14), 863–873 (2001)
8. Chiu, N.-H., Huang, S.: The adjusted analogy-based software effort estimation based on similarity distances. Journal of Systems and Software 80(4), 628–640 (2007)
9. Cohen, J.: Statistical power analysis for the behavioral science. Lawrence Erlbaum Hillsdale, New Jersey (1998)
10. Conte, D., Dunsmore, H., Shen, V.: Software engineering metrics and models. The Benjamin/Cummings Publishing Company, Inc. (1986)
11. 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)
12. Diaz, E., Bianco, R., Tuya, J.: Tabu Search for automated loop coverage in software testing. In: International Conference on Knowledge Engineering and Decision Support (ICKEDS), Porto, pp. 229–234 (2006)
13. Diaz, E., Tuya, J., Bianco, R.: Automated software testing using a metaheuristic technique based on Tabu search. In: Proceedings of International Conference on Automated Software Engineering (ASE 2003), pp. 3120–313 (2003)
14. Diaz, E., Tuya, J., Bianco, R., Dolado, J.J.: A tabu search algorithm for structural software testing. Computer and Operations Research 35(10), 3052–3072 (2008)
15. Dolado, J.J.: A validation of the component-based method for software size estimation. IEEE Transactions on Software Engineering 26(10), 1006–1021 (2000)
16. Gendreau, M.: An introduction to Tabu Search. In: Science Handbook of Metaheuristics. International Series in Operations Research & Management, vol. 57, pp. 37–54. Springer, Heidelberg (2002)
17. Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers, Boston (1997)

18. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
19. Harman, M.: The Current State and Future of Search Based Software Engineering. In: Workshop on the Future of Software Engineering (FOSE 2007), pp. 342–357 (2007)
20. 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 188(3), 898–909 (2008)
21. Huang, C.-L., Wang, C.-J.: A GA-based feature selection and parameters optimization for support vector machines. Expert Systems with Applications 31(2), 231–240 (2006)
22. ISBSG, `http://www.isbsg.org`
23. Kadoba, G., Shepperd, M.: Using simulation to evaluate predictions techniques. In: Proceedings of International Software Metrics Symposium, pp. 349–358. IEEE Press, Los Alamitos (2001)
24. Kampenes, V., Dyba, T., Hannay, J., Sjoberg, D.: A systematic review of effect size in software engineering experiments. Information & Software Technology 49(11-12), 1073–1086 (2007)
25. Kitchenham, B., Pickard, L.M., MacDonell, S.G., Shepperd, M.J.: What accuracy statistics really measure. IEEE Proceedings Software 148(3), 81–85 (2001)
26. Kitchenham, B.A.: A Procedure for Analyzing Unbalanced Datasets. IEEE TSE 24(4), 278–301 (1998)
27. Kitchenham, B.A., Pickard, L., Pfleeger, S.L.: Case studies for method and tool evaluation. IEEE Software 12(4), 52–62 (1995)
28. Kitchenham, B.A., Mendes, E.: A Comparison of Cross-company and Single-company Effort Estimation Models for Web Applications. In: Procs. EASE 2004, pp. 47–55 (2004)
29. Kitchenham, B., Mendes, E.: Travassos, Cross versus Within-Company Cost Estimation Studies: A systematic Review. IEEE Transactions on Software Engineering 33(5), 316–329 (2007)
30. Koch, S., Mitlöhner, J.: Software project effort estimation with voting rules. Decision Support Systems 46(4), 895–901 (2009)
31. Lanying, L., Shi, M.: Software-Hardware Partitioning Strategy Using Hybrid Genetic and Tabu Search. In: Proceedings of International Conference on Computer Science and Software Engineering, vol. 4, pp. 83–86 (2008)
32. Lefley, M., Shepperd, M.J.: Using genetic programming to improve software effort estimation based on general data sets. In: Proceedings of Genetic and Evolutionary Computation Conference, pp. 2477–2487 (2003)
33. 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 82(2), 241–252 (2009)
34. Mahmood, A., Homeed, T.S.K.: A Tabu Search Algorithm for Object Replication in Distributed Web Server System. Studies in Informatics and Control 14(2), 85–98 (2005)
35. 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 (2003)
36. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. Journal of Chemical Physics 21, 1087–1092 (1953)
37. Oliveira, A.L.I.: Estimation of software project effort with support vector regression. Neurocomputing 69(13-15), 1749–1753 (2006)

38. OpenTS, a Java Tabu Search Framework, http://www.coin-or.org/Ots/index.html
39. Royston, P.: An extension of Shapiro and Wilks Test for Normality to Large Samples. Applied Statistics 31(2), 115–124 (1982)
40. Shan, Y., Mckay, R.I., Lokan, C.J., Essam, D.L.: Software project effort estimation using genetic programming. In: Proceedings of International Conference on Communications Circuits and Systems, pp. 1108–1112. IEEE Press, Los Alamitos (2002)
41. Shepperd, M., Schofield, C.: Estimating software project effort using analogies. IEEE Transaction on Software Engineering 23(11), 736–743 (2000)
42. Shepperd, M., Schofield, C., Kitchenham, B.: Effort estimation using analogy. In: Proceedings of International Conference on Software Engineering, pp. 170–178. IEEE Press, Los Alamitos (1996)
43. Shukla, K.K.: Neuro-genetic prediction of software development effort. Information and Software Technology 42(10), 701–713 (2000)
44. 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, vol. 31, pp. 258–261 (2008) ISSN 1307-6884