Search-based Approach for Software Cost Estimation

Violeta Todorova Bozhikova and Mariana Tsvetanova Stoeva

Abstract – **Search-Based Software Engineering (SBSE) is a new Software Engineering (SE) branch that uses Search-based approach (SBA) to a big number activities of the software development process. The possibility of formulating a software engineering activity as an optimization problem is the reason for using Search-based approach to resolve it, which consists in applying meta-heuristic algorithms in order to find a rational (near optimal or sub-optimal) instead of optimal solution of the problem. The paper presents the author's work in the field of SBSE: arguments why software cost estimation is "Search|Optimization problem" are given and a search-based procedure for software cost estimation is presented.**

Keywords – **Search-Based Software Engineering, Software Cost Estimation, Tabu search method, Search-based approach**

I. INTRODUCTION

The term Search-Based Software Engineering ("SBSE") was first introduced by Harman and Jones in 2001, although the optimization has long been used to solve problems in Software Engineering. The reason is that almost all activities in software production can be formulated as optimization problems. This is an argument to use Search-based approach to solve them which consists in applying meta-heuristics in order to find a rational (near optimal or sub-optimal) instead of optimal solution of the problem [1]…[8].

Meta-heuristic search algorithms such as hill climbing, Tabu search, simulated annealing and probabilistic search algorithms (for example, GA) are local search-based techniques, widely used to solve various optimization problems in the presence of many local extremes, with many parameters and conflicting constraints. They effectively find acceptable approximations of many considered as "NP-complete" and "NP-complex" problems, where is impossible or unreasonable to use accurate analytical algorithms that produce the optimal solution, but is possible to determine which of two candidate solutions is better. These algorithms are easily designed and implemented. The only downside is that the final solution may be far from optimal. But, in many practical cases, it is preferable to the alternative - an endless and hopeless search for the optimal solution.

V. Bozhikova is with the Department of Computer Science and Technologies, Faculty of Computer Technique and Automation, Technical University - Varna, 1 Studentska str., 9010 Varna, Bulgaria, e-mail: e-mail: vbojikova2000@yahoo.com

M. Stoeva is with the Department of Computer Science and Technologies, Faculty of Computer Technique and Automation, Technical University - Varna, 1 Studentska str., 9010 Varna, Bulgaria, e-mail: mariana_stoeva@abv.bg

"Hill climbing" is considered to be the most used meta-heuristic search technique. There are many variations of this approach. The search process begins typically with a "current solution" (current state), usually an accidental solution. Many "neighboring" solutions are then appreciated and a "neighboring solution" that improves the goal function is selected which becomes the "current solution" and then the process is repeated. It is believed that this is the best technique for solving optimization problems and it is reasonable to start searching solution with it. Some authors even consider [4] that if this technique gives a worse result than another meta-heuristic, either the problem is not well understood, either the problem formalization is inadequate.

The purpose of this paper is both to show that Software Cost Estimation is "Search|Optimization" problem and to comment the authors work in the field of SBSE: a meta-heuristic procedure for software cost estimation is presented in this context.

Arguments why software cost estimation is "Search|Optimization problem" are given in the next section.

II. SOFTWARE COST ESTIMATION AND SBSE

SBSE seeks to reformulate Software Engineering problems as "search problems" [1]... [8]. Possibility of formulating a problem as a "Search|Optimization problem" is a reason for applying meta-heuristic approach to solve it. Meta-heuristics are solution methods that organize an interaction between local improvement procedures and higher level heuristics to escape from local optima and to perform a robust search of solution space.

Why Software Cost Estimation could be seen as a Search|Optimization problem? This is because Software Cost Estimation is an activity that has a large area of solutions and does not have an effective accurate solution. This is because an appropriate objective function for evaluating the solutions could be designed and an easy (not expensive) generation of candidate solutions could be invented.

Software Cost Estimation is associated with the prediction of the resources needed for the software production (usually human effort, project duration, staff needed and cost in dollars and so on), so helping the project managers to evaluate the project progress and to ensure that the spending will not exceed the budget provided. It becomes more and more difficult task due of the enhanced development of increasingly large and complex software projects and the specific nature of the software as product: to estimate something that cannot be seen and touched is a complex task that requires great knowledge and experience.

The research efforts in the field of Software Cost Estimation are directed at developing reliable and effective methods and tools. After more than 20 years of research in this field a big number of software cost estimation methods are available but no one method is considered to be the best for all type projects. The advantages and disadvantages of the existing methods are often complementary each other which is a reason to use a combination of methods, most of which algorithmic in order to find the best estimate of the costs. The algorithmic methods such as CoCoMo, Function Point Analysis, Putnam model, etc. are considered to be more reliable (i.e. more objective and accurate) than the non-algorithmic methods.

The next section discusses our approach for Software Cost Estimation and focuses on the elements of a Tabu search procedure for software effort estimation.

III. Applying Tabu Search Heuristic for Software Cost Estimation

A. Our Hybrid approach for Software Cost Estimation

The Tabu search procedure, presented in this paper is based on our approach for Software Cost estimation, proposed for first time in [7] which is a combination between Basic COCOMO, Intermediate COCOMO, COCOMO II and Function Point Analysis. In fact, the cost function PMadjasted, commented in [7] and [8] is a function of the effort adjustment factor EAF.

$$PM_{adjasted} = PM_{nom}(EAF)[person-months] \quad (1)$$

where

$$PM_{nom} = EF(KSLOC)^{ee}[person-months] \quad (2)$$

and

$$EAF = \prod_{1}^{CDN} EM_i \quad (3)$$

where

$$CDN = \begin{cases} 18 & if \quad COCOMO \ II \ is \ used; \\ 15 & if \quad COCOMO \ Intermedia te \ COCOMO \ is \ used. \end{cases}$$

EAF is calculated as a product of the effort ratings of the fifteen Intermediate CoCoMo Cost Drivers or the seventeen COCOMO II Cost Drivers, plus one extra. Total of eighteen Cost Drivers in the latter case are grouped into 4 major categories “Personnel attributes”, “Project attributes”, “Platform attributes” and “Product attributes”. An additional user defined cost driver, named USER was added to the classic COCOMO II Cost Drivers:

Personnel factors

1. Analyst Capability,
2. Applications Experience,
3. Programmer Capability,
4. Language and Tool Experience,
5. Personnel Continuity
6. Platform Experience

Product factors

7. Required Software Reliability
8. Database Size
9. Software Product Complexity
10. Required Reusability
11. Documentation Match to Life-Cicle needs

Platform factors:

12. Execution Time Constraint,
13. Main Storage Constraint,
14. Platform Volatility,

Project factors

15. Use of Software Tools,
16. Multi-site Development,
17. Required Development Schedule,
18. USER.

The fifteen Intermediate CoCoMo Cost Drivers are structured into the following four categories:

Product attributes

1. Required software reliability
2. Size of application database
3. Complexity of the product

Hardware attributes

4. Run-time performance constraints
5. Memory constraints
6. Volatility of the virtual machine environment
7. Required turnabout time

Personnel attributes

8. Analyst capability
9. Software engineering capability
10. Applications experience
11. Virtual machine experience
12. Programming language experience

Project attributes

13. Use of software tools
14. Application of software engineering methods
15. Required development schedule

The possible values for these attributes depend on the impact they could have on a project. We could see for example in [7] that the rating of “Required Software Reliability” varies from 0.75 to 1.4 where the value of “Programmer Capability” changes from 1.14 to 0.88 depending on the project.

PM_{nom} is a product of the size of the software product in KSLOC (in thousands of source lines of code) and EF - a coefficient based on Intermediate COCOMO model:

Software project	EF	ee
Organic	3.2	1.05
Semi-detached	3.0	1.12
Embedded	2.8	1.20

PM_{nom} could be considered as an early design cost estimate of the software project. According our approach, the estimation of the product size begins with the calculation of ΦT - Unadjusted Function Points. In order to calculate ΦT, the estimator has to classify the product functions into 5 groups: Inputs, Outputs, Files, Interfaces, and Queries. Within each group, the functions are classified (according to their complexity) in simple, medium and complex. So, the weight of each function depends on both: on its type and its complexity. Finally, Unadjusted Function Points (ΦT) is the sum of the weights of all functions, which is expressed by the next formula:

$$\Phi T = \sum_{i=1}^{5} \sum_{j=1}^{3} Nij \quad Wij \quad (4)$$

where Nij and Wij are respectively the number and weight of functions type i with complexity j.

Next, the unadjusted function points are converted into equivalent SLOC depending of a LangFactor of the language used. For example, the LangFactor [3] for Assembly language is 320SLOC/UFP, for C++ - 29SLOC/UFP, for Fortran 77 – 105SLOC/UFP, for Lisp – 64SLOC/UFP, for Pascal – 91 SLOC/UFP and so on.

Next, the value of DI ("degree of influence" fourteen application characteristics) is accounted. DI calculation is based on the rating (rating scale of 0 to 5 for each characteristic) of fourteen application characteristics (such as performance, reusability, etc. - the figure below). The ratings of the 14 characteristics are added together (6), then the result DI is multiplied to 0.01 (equation 7) and added to a base level of 0.65 to produce a general characteristics adjustment factor that ranges from 0.65 to 1.35.

$$DI = \sum_{1}^{14} rating_i \qquad (5)$$

$$SLOC = \Phi T \times (0.65 + (0.01 \times DI)) \times LangFactor \qquad (6)$$

B. Applying Tabu Search Heuristic for Software Cost Estimation

Tabu search is often used for solving optimization problems. Glover and Laguna give in 1997 a comprehensive description of this technique. As in classical local search process, the general step of Tabu search process consists in constructing new solution from the neighborhood (the search space) of the current solution "cs" and in checking whether the search process should stop there or perform another step. Specific to Tabu search method is that the local search procedure tries to avoid falling into local optima by creating a special list of forbidden solutions (forbidden moves), called "Tabu" list for each current solution "cs". Below, the general Tabu search algorithm is summarized:

1. Identify the starting solution (usually random generated), let's accept it as current solution.
2. Loop

Define the neighborhood set of solutions for the current solution (define the search space);
Identify the Tabu set of solutions for cs (define a forbidden moves);
Identify the aspiring set of solutions for the current solution (define aspiring moves);
Choose the best neighbor solution for the current (find the best move) and take it to be the current solution;
Exit when the goal function is satisfied or stopping condition is reached.
End Loop

3. The final (sub-optimal) solution is found

C.Tabu search procedure for software effort estimation

The key issues that have to be addressed in order to use Tabu search for Software Cost Estimation are:

- How to evaluate the neighbourhood (e.g. what is the objective function)?

Our procedure, presented below, tries to minimize (7) the value of the effort function PMadjusted (1) simultaneously trying to satisfy a restrictive condition (8) in terms of a predetermined threshold PM_{max}.

$$PM_{adjasted} = \min \qquad (7)$$

$$PM_{adjasted} \le PM_{\max} \qquad (8)$$

In order to satisfy the goal function (9), the space of all possible combinations of cost drivers is searched in a sequence of moves from one possible combination to the best available alternative which minimizes the effort adjustment factor EAF, taking in consideration the forbidden combinations saved in a "Tabu" list:

$$EAF = \min \qquad (9)$$

The goal function is a measure of the quality of the current combination of cost drivers and shows how well the cost driver's combination is a solution to the problem.

- How to define the Tabu list, the aspiration criteria and the termination criteria?

In general, the criteria for classifying aspiring and forbidden moves are specific to the application. To minimize the chance of cycling in the same solution, any combination of cost drivers which has been already selected is put into a "Tabu" list so that it becomes 'taboo' (forbidden).

The six steps of our Tabu search procedure are described below:

1. Identify an initial solution that satisfies the restrictive condition.

Loop
  Generate a set of Cost Driver values.
    If the set is not in the Tabu List
    Calculate EAF, then $PM_{adjusted}$;
    Put the set in the Tabu List.
    Endif
  Exit when the restrictive condition is satisfied (an initial solution is found) or the specific number of iteration is reached.
EndLoop

2. Optimize the solution found (if is found in step 1): Identify a sub-optimal solution that minimizes the goal function, generating new set of Cost Driver values.

Loop
  Generate a new set of Cost Driver values.
  If the set is not in the Tabu List
    Put the set in the Tabu List.
    Calculate $EAF_{current}$.
    If ($EAF_{current}$ < EAF)
      EAF= $EAF_{current}$
      Calculate $PM_{adjusted}$;
    EndIf
  EndIf
Exit when the specific number of iteration is reached.
EndLoop

The final (sub-optimal) solution is found.

IV. CONCLUSION

The paper presents the author's work in the field of SBSE: arguments why software cost estimation is a "Search|Optimization problem" are given, our hybrid approach for software cost estimation is commented and a Tabu-search procedure for software cost estimation is presented.

Although to date we have not yet conducted extensive experiments, the results of the study of a small number of student projects are encouraging: the Tabu search algorithm finds for a reasonable time a solution that is close to optimal.

REFERENCES

[1] Goran Mauša, *Search Based Software Engineering and Software Defect Prediction*, http://www.fer.unizg.hr/_download/repository/Kvalifikacijski_-_Goran_Mausa.pdf

[2] Mark Harman and al., *Search Based Software Engineering: Techniques, Taxonomy, Tutorial*, http://www0.cs.ucl.ac.uk/staff/mharman/laser.pdf

[3] Mark Harman and Bryan F. Jones, *Search based software engineering*. Information and Software Technology, 43(14):833{839, December 2001.

[4] John Clarke, Jose Javier Dolado, *Reformulating Software Engineering as a Search Problem*, http://www.discbrunel.org.uk/seminal

[5] Filomena Ferrucci, Carmine Gravino, Rocco Oliveto, and Federica Sarro, *Using Tabu Search to Estimate Software Development Effort*, http://www0.cs.ucl.ac.uk/staff/fsarro/resource/papers/C20.pdf

[6] *COCOMO II Model Definition Manual*, ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/cocomo97docs/modelman.pdf.

[7] Bozhikova V., M. Stoeva, *An Approach for Software Cost Estimation*, Proc. of the International conference on computer systems and technologies (CompSysTech'2010), International Conference Proceedings series - V.471, София, Bulgaria, 2010, pp.119 -124, ACM ISBN 978-1-4503-0243-2.

[8] Bozhikova V., *Estimation of the software costs using heuristic search algorithms*, Proceedings of third Int'l Scientific Congress 50 anniversary TU-Varna, ТУ-Варна, Volume 1, pp.162-165