A Hybrid Heuristic Approach for Scheduling Bug Fix Tasks to Software Developers

Fernando Netto, Márcio Barros, Adriana C. F. Alvim
Postgraduate Information Systems Program – UNIRIO
Federal University of Rio de Janeiro State
Av. Pasteur 458, Urca – Rio de Janeiro, RJ – Brazil
{fernando.netto, marcio.barros, adriana}@uniriotec.br

Abstract

Large software projects usually maintain bug repositories where both developers and end users can report and track the resolution of software defects. These defects must be fixed and new versions of the software incorporating the patches that solve them must be released. The project manager must schedule a set of error correction tasks with different priorities in order to minimize the time required to make the next release available and guarantee that the more important issues were fixed. In this paper, we propose a hybrid heuristic approach based on an algorithm to schedule error correction tasks which combines a genetic algorithm and a constructive heuristic. We believe that our method can propose schedules for bug fix tasks with lower cost when compared to an ad-hoc schedule for non-trivial projects.

1. Introduction

Bug reporting tools are commonly used to facilitate the communication among groups involved in software development and to control the defect resolution process [1]. Bug reporting tools may control a large number of bugs. These bugs are classified to facilitate search, retrieval, and updates. Bugs are usually associated to a component (the subset of the software where the bug was observed), a version (the release of the software where the bug was identified), and a priority (indicating how fast the development team must provide a patch that solves the bug).

Usually, when a bug is identified the project manager selects a developer to work on a patch to solve the problem described by the bug. When several bugs require attention from the development team, the selection of developers to solve each bug and the order in which those bugs are attended is determinant to the software project: the schedule of bug fix tasks determines which bugs will be corrected in each future release of the system. Therefore, project managers face an optimization problem: given a number of bugs reported for a software project, how to schedule bug fix tasks with different priorities to developers in order to minimize the cost of fixing these bugs?

By cost, we mean a measure of the quality of the schedule. This measure must be aligned to schedule's objectives, that are: (i) to minimize the time required to make the next release available; (ii) to guarantee that the more urgent bugs will be fixed in the next release; and (iii) to maximize the occupation rate of the test team. The lower the cost, the better is the schedule.

In this paper, we propose a hybrid heuristic approach to build a feasible schedule for a given set of bug fix task with different priorities to developers. The approach combines the genetic algorithm paradigm, based on random keys encoding, with a constructive heuristic. By using this method, a project manager can find the proper allocation of bug fix tasks to developers, in order to minimize the cost to accomplish the set of tasks.

2. Modeling the Schedule of Bug Fixes

Regarding the cost of scheduling a bug, duration is the key attribute to the optimization process. The duration (D_I) of a bug fix task is the time span from the date a developer starts working on a correction to the related bug to the date when the bug fix task is completed. The completion time of a bug fix task (CT_I) is the time span from the schedule start date to the date when the bug fix is completed. Given each bug's duration and a partial order of these bugs, we can calculate each bug's completion time.

It is desirable to anticipate the resolution of more urgent and the less complex bug fix tasks. This approach reduces the mean completion time required to fix the sets of bug since it is based on the shortest-processing-time heuristic [2]. Also, if a new release is required by the client before the conclusion of the planned schedule, it is likely that the more urgent bugs are already fixed. Finally, this strategy maximizes the occupation of the test team, since the more important bugs are fixed first and will be available for tests sooner than less relevant ones.

In order to achieve the objectives described in the previous paragraph, we define the cost of scheduling each bug as its completion time multiplied by a factor that increases with bug priority. So, bugs requiring less effort to be corrected and high priority bugs will be schedule before complex and low priority bugs. Our optimization process searches for schedules where the overall cost (the sum of each bug's cost) is minimal. This problem is similar to the NP-hard problem of scheduling a set of jobs with distinct priorities to a set of parallel service machines which was previously studied by Bruno et al [2].

Consider a set B of $b \geq 1$ bugs reported to a software system and another set D of $d \geq 1$ developers, available and capable of fixing those bugs. A task to correct a bug $j \in B$ is specified by its positive priority w_j and the positive effort t_j required to accomplish it. Consider a set S of all feasible schedules based on B and D. Given a schedule scheme $E \in S$, we denote by $f_j(E)$ the completion time of the task intended to correct the bug $j \in B$. Since we are interested in finding the schedule that minimizes the weighted sum of the time required to complete the bug fix tasks, we selected Equation (1) as a fitness function for a given schedule $E \in S$.

$$C(E) = \sum_{j=1}^{n} w_j f_j(E) \tag{1}$$

3. Hybrid heuristic approach

Genetic algorithms are used to solve a wide variety of optimization problems, including multiple machine scheduling [3]. The main idea consists in generating new individuals through the processes of crossover and mutation. The purpose of a genetic algorithm is to find good quality chromosome after several generations.

We have used the Random Keys Genetic Algorithm (RKGA) method introduced by Bean [3] to solve the bug fix scheduling problem. In our proposed algorithm, each chromosome represents a solution to the scheduling problem and is encoded as an array of N genes, where N is the number of bug fix tasks. Each gene is initialized with an integer random number between 1 and M, where M is the number of available developers.

Given a chromosome, the schedule it represents is built in two steps. First, we must assign a developer to accomplish each bug fix task (the genes). The algorithm reads each allele value (the random number) and sets a specific developer to fix the respective bug. For instance, if the random value assigned to a given gene was 9, then the decoding routine might assign developer number nine to fix the corresponding bug. Thus, we have the developer that will handle each particular bug, but not yet the sequence in which the bugs will be resolved by each developer.

In the second step, we sequence the tasks assigned to each developer using the LIST SCHEDULING constructive heuristic described in [4]. This heuristic establishes that tasks must be executed in the ascending order of their e_t/w_t ratio, where e_t is given by the effort/time required to conclude a given task and w_t is the task weight (or priority).

The proposed algorithm starts by creating an initial population consisting of a set of randomly generated chromosomes. Then, the schedule for each chromosome is generated using the list scheduling heuristic and the cost of these schedules are calculated using the fitness function defined in Equation 1. Next, the individuals are sorted according to the ascending order of their cost value. The lower the cost, the better is the solution. New populations are generated through the processes of crossover and mutation until a stop criterion is achieved.

4. Conclusions and Future Work

Given the size of the search space, we believe that the proposed method might produce schedules with lower maintenance cost than an ad-hoc schedule. The model presented is a simplification of the complex reality of bug fix task distribution. As an extension of this work, we intend to overcome some limitations by including developer's availability and skills information to build a more precise model. An experiment is required and it should properly address the stability of the GA parameters, such as population size and mutation percentile in order to find the more suitable percentiles.

References

[1] D'Ambros, M.; Lanza, M.; Pinzger, M., ""A Bug's Life" Visualizing a Bug Database", 4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007.

[2] Bruno, J.; Coffman Jr., E.G.; Sethi, R., "Scheduling Independent Tasks to Reduce Mean Finishing Time", Communications of the ACM, 17:382-387, 1974

[3] Bean, C., James. "Genetic algorithms and random keys for sequencing and optimization", ORSA Journal of Computing, vol. 6, pp. 154-160, 1994.

[4] Skutella, Martin, "Approximation and Randomizing in Scheduling", Berlin, 1998.