Prediction of Software Performance Using Genetic Programming

I. Thamarai¹,* and S. Murugavalli²

¹Department of CSE, Research Scholar, Sathyabama University, Chennai, India.
²Department of CSE, Research Supervisor, Sathyabama University, Chennai, India.
e-mail: ¹ilango.thamarai@gmail.com

Abstract. Performance is a non-functional requirement for a software product. It is related to reliability, security and other non-functional requirements. Various approaches are available for software performance prediction. In this paper we present a novel method of using Genetic Programming in reverse engineering concept. Reverse Engineering is the process of analyzing software product with the aim of recovering its design. Genetic Programming is applied for components to predict its performance. Reusing of existing components is common in Component Based Software Engineering. The response time of an entire component is predicted from the duration of execution of individual component in it. In this paper, the time taken for compression of real time data is predicted and actual time is measured. This approach can be best utilized to reverse engineer parameterized behavior models consisting of black box components.

Keywords: Performance prediction, Reverse engineering, Genetic search, Genetic programming, File compression.

1. Introduction

The Prediction of performance of a software product is very important as it is related to other non-functional requirements of a software product such as reliability and maintainability. As today’s software products are largely component based, the prediction is also done on the basis of individual components of the software [2]. The performance of the full system can be predicted by creating models to calculate the performance of single component. The performance of a component based software product depends on four factors namely assembly context, runtime context, usage context, deployment context and the implementation of the individual component of the software system. The assembly context refers to the software system and the connection between the components. The usage context refers to the software system and the connection between the components. The deployment context refers to the execution platform consisting of software and hardware.

The methods that exist today are monolithic and they do not separate these four factors [3]. But for performance prediction, we have to consider these factors as explicit parameters [1]. In this paper, we present a new approach that is used to reverse engineer the models of already existing software for which the source code is unavailable. When a new software system is developed, it uses existing components whose functionalities are proven [13]. The concept of genetic programming is used to create models for such black box components. Genetic Programming is an extension of Genetic Algorithm. It offers the flexibility to perform the operations in a hierarchical manner. The execution time is predicted by byte code benchmarking. This approach is validated using SPECJVM2008 compression for text and image files. In section 2, we describe related work. In section 3 concepts are explained and its implementation is given. In section 4, conclusion is given.

2. Related Work

Search based Software Engineering can be used in the prediction of software effort. In [4] Mark Harman and Afshin Mansouri suggested that Genetic Algorithm can be successfully used in software engineering methods like software testing, software design and software effort prediction. Search based techniques can also be used in problems like

*Corresponding author

© Elsevier Publications 2013.
optimization of project planning, testing and quality assessment [5]. Zheng et al concentrated on run time monitoring and online prediction of performance of software [6]. Their estimation was based on the metrics like resource utilization and response time. In [8] performance prediction method was proposed based on byte code benchmarking. But their proposal was based on the interpreted execution of byte code. In [11] the authors presented a method that uses a semantic approach to retrieve components from a heterogeneous software repository. For this, Genetic Algorithm based approach is used. It is suggested that Genetic Algorithm offers significant advantages over other approaches due to its ability to naturally represent the data. A group of components that compose a system can have many overlaps and gaps. Hence using Genetic Algorithm in the software component retrieval is proposed.

In [7], Ekram Kocaguneli et al. proved that AI techniques provide significant improvement in estimation accuracy. Arturo Chavoya et al. created a LISP implementation of the Genetic Programming algorithm to create a model for predicting software development effort. Their results proved that prediction accuracy of a genetic programming model is equal to the feed forward neural network for short scale projects [9]. Filomena Ferrucci et al. in [12] emphasize the choice of fitness function as it would significantly affect the accuracy of prediction. It is also proved that Genetic Programming provides better estimate than CBR [Case Based Reasoning] and MSWR [Manual Stepwise Regression] for a dataset. Genetic Programming is a specialization of Genetic Algorithm. Similar to Genetic Algorithm, Genetic Programming also focuses on the evolution of genotypes. The fundamental difference is in the representation followed.

Genetic Algorithm uses string or vector representation while Genetic Programming uses a tree representation. The concept of Genetic Programming is explained in detail in Section 3.

3. Genetic Programming and Performance Prediction

The Non continuous functions are very common in software engineering applications due to branching statements. Genetic Programming can be effectively used in such situations. It is robust against distributive data. Genetic Programming was mainly developed to evolve executable computer programs [10]. Using a tree based representation in Genetic Programming requires adaptive individuals and domain specific grammar. A grammar forms an important part in chromosome representation. This requires defining a terminal set, a function set and semantic rules. All the variables and constants should be specified in terminal sets. The function set consists of all the functions applied to the elements in the terminal set. Semantic rules ensure the proper construction of the trees. The initial population is randomly generated. The individuals are initialized as simple as possible. This is because during the evolutionary process, these individuals grow and as a result complexity increases. The fitness function for a genetic programming depends on the problem.

In this paper, Genetic programming utilizes the available genes to build a tree structure. Mutation is applied to the genes that are constant. Crossover is applied on sub trees of individuals. Then appropriate input values are selected and other genes that are not relevant to the function are rejected. The fitness of the genes is judged based on two criteria. One is the prediction error and the other is the complexity of the expression. The process of evaluation, selection and modification of individuals to produce a new generation continues until some termination condition is satisfied. The termination condition may be reaching pre-defined number of generations or a desired individual has been found.

During the design and development of a software system, both functional and non-functional requirements are to be considered. Performance is a non-functional requirement which is related with other quality attributes of a software product such as reliability, security and maintainability. Many Effort Estimation models suffer due to incorrect performance prediction [14]. Performance prediction of component based software system is necessary for the evaluation of design decisions. In component based software engineering, the existing components are reused in the new application. The response time of an individual is predicted from the time taken for executing the individual component. In [15], the authors says that systematic exploration of relationship between prediction system accuracy and characteristics of data set will lead to a better understanding of when to use a particular technique. The overview of our approach is shown in figure 1.

For the component considered, execution is done. The executed byte code instructions and method calls are counted. As the next step, inputs and outputs of the component are monitored. Genetic Programming is used to estimate the dependencies between the input data and number of executions for each byte code instruction. Based on the result, a behavioural model is constructed. This model is used to predict the performance of the application.

The approach is tested in a java implementation of a file sharing application. The time for compression of image files and text files that are chosen randomly are predicted using byte code benchmark. Then actual time is also measured and the values are shown in table 1.
The Prediction accuracy ratio $[R]$ is calculated as below:

$$R = \frac{\text{Prediction time based on byte code}}{\text{Actually measured time for compression}}$$

It is found that the average prediction error is less than 30 percent. That is, the according to the results accuracy in prediction is more than 70 percent.
4. Conclusion

In this paper, we have presented a novel approach using Genetic Search to predict the performance of an application. We have successfully evaluated the concept of reverse engineering and performance prediction using Java implementation of a file sharing application. Genetic Programming is used for estimating resource consumptions of black box components for which only byte code is known. The results show that this approach gives almost accurate prediction results. The reliability and efficiency improves by adopting this method. Also our approach is suitable for components whose source code is not known. In future, we plan to capture multiple files concurrently with respect to byte code execution.

References