CHAPTER 1

INTRODUCTION

Compiler optimizations aim at improving the quality of the code, generated for an input program. The optimizing compiler makes critical decisions regarding the type of optimization to be performed, and the program segments to target the optimization. Typically, optimizations can be performed either statically or dynamically. The traditional compilers generate statically optimized codes irrespective of their execution behavior, thus limiting the performance of the statically generated code. Virtual machine environments like the Java Virtual Machine and Microsoft’s Common Language Runtime, however, convert the program in a high level language into machine independent statically optimized bytecodes that are executed by an execution engine. The Just-In-Time (JIT) execution engine, in turn, converts the bytecodes to machine dependent codes before execution. These dynamically generated machine codes are then optimized on the fly. Since dynamic optimizations are expensive, only selected regions of the code are optimized.

The selective dynamic optimizations are more effective than static optimizations, because they are tuned to the runtime program behavior. To tap the runtime program behavior, hardware, software and hybrid profiling techniques are used. Even though the hardware counters are the fastest solutions for profile collection they are dependent on the machine architecture. Software solutions are slower because of the additional code that should be executed in order to perform profile collection. Moreover, the
overhead increases with the collection of a variety of profile information from the program. The hybrid solution is a combination of both hardware and software profiling techniques, with the benefits and limitations of both. Since the virtual machines use online profile information for selective optimization during runtime, the accuracy of the profile information is crucial to the effectiveness of the optimizations. Unfortunately, the overhead from accurate profiling impedes the program speed and performance. Hence, the primary aim of refining the profiling techniques is to minimize the associated runtime profile collection overhead. Selective optimizers using online profiles, delay their optimization decisions until sufficient profiles are collected, and this affects the start up performance of the program. Selective optimizers that use offline profiles do not depict the current behavior of the program. So, any contribution for improving the dynamic optimization system is relevant and important.

With the advent of new computer architectures and environments, finding a suitable optimization technique for a program, and collecting additional profiles at runtime for optimization is difficult. Therefore, the identification of the best region in a program for optimization without profile collection is a big challenge. Program segments called hot methods that are frequently executed and long running, are some of the preferred sequences in selective optimization, and predicting these regions of the code will relieve the virtual machine from profiling. This new approach is bound to transform the way compiler writers develop optimization heuristics. So, this work proposes a new selective optimization technique to predict hot methods by using the Machine Learning (ML) algorithm, the Support Vector Machine (SVM).
1.1 MOTIVATION

The novel idea of machine learning induced hot method prediction is generated from the concepts and arguments found in the literature on profiling and machine learning. Although there are several machine learning based approaches for the construction of predictive heuristics in optimization research, the works of Stephenson and Amarasinghe (2005) and Monsifrot et al (2002) are primarily responsible for triggering this work. Monsifrot et al (2002) have used machine learning algorithms to devise a heuristic for loop unrolling while Stephenson and Amarasinghe (2005) have applied machine learning in building a predictive model for the loop unroll factor.

Besides these works, the publication by Arnold et al (2005), reviewing the challenges confronting dynamic optimizations in production virtual machines, has also served as a source of motivation. They have discussed the profiling and adaptive optimization techniques under different categories, such as, selective optimization, profiling for feedback directed optimization (FDO), feedback directed code generation using profile information to improve quality of the generated code, and other FDOs using profile information. They have also laid emphasis on FDO techniques, like inlining, code layout, instruction scheduling, etc., to improve the quality of the dynamic code generated by the optimizing compiler. While concluding, they have called for research developments in adaptive optimization areas ranging from optimization control policies to challenges from new microprocessor architecture and design.

The issues discussed in these works and the introductory sections of this thesis have provided the motivation to work on machine learning based hot method prediction for optimization.
1.2 OBJECTIVES OF THE THESIS

Based on the issues mentioned above, the objectives of this work are formulated as follows:

(i) To try out an approach other than profiling to perform selective optimization. The idea is to eliminate runtime profile collection overhead, by developing machine learning-based predictive models to predict the hot methods of a program offline, prior to optimization.

(ii) To build two independent models, one for predicting the frequently called hot methods and another for the long running hot methods.

(iii) To use the machine learning algorithm, SVM, in building the predictive model.

(iv) To apply the standard search tool of the genetic algorithm in feature subset selection, aimed at maximizing the hot method prediction accuracy.

(v) To implement a new ‘knock-out’ algorithm coupled with the classical sequential backward elimination process, in the construction of an effective feature set in order to cut the time spent in feature selection.

(vi) To investigate the impact of feature categories and their combinations on building the predictive models.

(vii) To evaluate the benefits of the proposed technique by applying inter-procedural and intra-procedural optimizations on the predicted hot methods.
(viii) To compare the execution time of a program, whose predicted hot methods are optimized against the traditional profile-based optimization system.

(ix) To develop a relearning virtual machine.

### 1.3 ORGANIZATION OF THE THESIS

Based on the objectives stated above, the work has been carried out and presented in the thesis as given below.

Chapter two gives an overview of the concepts, tools and standard techniques employed in this work, followed by the literature review. This chapter features the selective optimization ideas in the opening section. The subsequent sections discuss machine learning, the basic SVM concepts, the Low Level Virtual Machine (LLVM), selective optimization techniques, the genetic algorithm, the feature selection algorithms and the benchmarks used. The latter part of the chapter reviews papers with respect to profiling based hot method prediction, the application of ML in compilers and the most important optimization component of inlining. Also, the works related to the standard feature selection techniques in ML are introduced, and a brief review of the literature on relearning concludes the chapter.

Chapter three explores the strategies for the machine language based technique and the construction of two independent models, one for predicting the frequently called hot methods, and another for the long running hot methods, trained with an exhaustive full set of ninety features. The picking of static features, construction of training and testing data sets, training the SVM-based predictive models and the evaluation of the prediction of different set of benchmark programs are discussed. The section on evaluation presents a detailed account of the new metrics used in the prediction technique.
Chapter four describes the application of the genetic algorithm (GA) to the problem of feature selection. The first section gives an account of the application methodology. The hot method prediction ability of the frequently called and long running hot method predictive models is discussed in the next section. Chapter five describes feature reduction by implementing a new ‘knock-out’ algorithm, which is a combination of the conventional sequential backward elimination process, and a novel ‘knock-out’ scheme, developed for a faster construction of an effective feature subset.

Chapter six is the feature category-wise analysis. An account of the sequential forward selection procedure is followed by the results of the impact of individual feature categories and the category combinations on hot method prediction. The chapter concludes with the result of the evaluation of the hot method prediction models. The impact of optimizing the predicted hot methods on program performance is presented in Chapter seven. The application of inlining and intra-procedural optimizations in the form of constant propagation and loop unrolling, and the resulting performance improvement over the profiling based system are described in detail.

Chapter eight introduces relearning in the virtual machine. An overview of the relearning virtual machine’s system architecture is followed by the performance evaluation of relearning by the frequently called hot methods (FCHM) and long running hot methods (LRHM). Chapter nine concludes with a summary of the work done and the highlights of the research. It also suggests a few topics for future work that could be carried out in continuation of this research.