Multi-Criteria Compiler-Based Optimization of Hard Real-Time Systems

Kateryna Muts, and Arno Luppold, and Heiko Falk
Institute of Embedded Systems
Hamburg University of Technology
Germany
k.muts|arno.luppold|heiko.falk@tuhh.de

ABSTRACT
Real-Time Systems often come with additional requirements apart from being functionally correct and adhering to their timing constraints. Common additional optimization goals are meeting code size requirements or the reduction of energy consumption. We show how to extend modern compiler frameworks to allow for optimizations towards multiple design criteria.

CCS Concepts

• Theory of computation → Evolutionary algorithms;
• Computer systems organization → Real-time systems; • Software and its engineering → Compilers;

Keywords

Compiler, Multi-Criteria, Optimization, Real-Time Systems

1. INTRODUCTION

Embedded systems are highly specialized integrated systems tailored towards solving a very specific task. Many times, the system must react within a given deadline. If missing this deadline might lead to catastrophic system failure, the system is called a hard real-time system.

One of the key properties of such a system is the so-called Worst-Case Execution Time (WCET). The WCET is defined as the worst possible execution time of a program, independently from its input data. If the WCET is too high, the program may violate its timing constraints. In a hard real-time system, such a program is considered to be functionally broken, i.e., returning the result too late is considered as bad as returning a wrong result.

To ensure that the timing constraints are met, optimizing compilers such as the WCET-Aware C Compiler Framework (WCC) [5] can be used. Traditionally, WCC focuses on minimizing the WCET of the compiled program, in order to guarantee that all timing constraints will always be met. Lately, WCC was extended to optimize multi-core systems [13] and multi-tasking systems [12]. This allows WCC to optimize systems with several tasks running on multiple cores with regard to their schedulability.

However, in modern embedded systems, time is just one resource for which hard design restrictions apply. Other criteria include, but are obviously not limited to, life-time, program size and energy consumption.

An optimized program which provably meets all timing constraints is not very useful if its code footprint is too large to fit into the available flash memory or if the high processing load leads to massive battery drainage.

As a result, optimizing compilers for embedded systems must be extended towards coping with multiple design goals at the same time. Some of these goals might be re-formulated as a constraint which must be held (e.g., “WCET must be below n cycles”), some of them are somewhat best-effort (“The energy consumption should be as low as possible, while the WCET is met”).

To tackle these design issues, we strive to extend the WCET-Aware C Compiler Framework towards flexible multi-criteria optimizations. The overall goal is to build a flexible framework where multiple design goals can arbitrarily be combined as needed by the system designer.

In the following sections of this paper, we will show our current approach to realize this task as well as our preliminary current results. Due to the quite early state of the project, the evaluation will focus on bicriteria optimization of a single tasking system towards WCET and code size. However, the framework is designed such that additional design goals such as energy consumption or minimal chip aging can be added as needed in the future.

This paper is organized as follows: Section 2 will first give a brief overview of related work, both with regard to existing multi-criteria optimizations and promising compiler optimization techniques. Section 3 briefly introduces the WCC compiler framework used as a basis for the proposed multi-criteria optimization framework. Section 4 explains the optimization framework. In Section 5, we present preliminary evaluation results. This paper closes with a conclusion.

2. RELATED WORK

Almost all work presented so far has concentrated on modeling and optimizing just one objective function such that WCET, energy consumption or code size, being completely monocriterial.

M. Lukasiewycz et al. [11] describe a method for the multi-criteria optimization of systems whose design space
can only be represented by means of binary decision variables.

Chakraborty et al. [2] extend an evolutionary algorithm by a local branch-and-bound search to modify individuals of the evolutionary algorithm that violate design constraints. However, this requires to efficiently find any valid solution in the design space.

P. Łokciewski et al. [10] exploit the stochastic evolutionary multi-objective algorithm identifying Pareto optimal compiler optimization sequences for the pairs of objectives WCET and Average Case Execution Time (ACET) or code size. In the paper, a string where each character denotes a specific optimization is considered as a decision variable.

3. WCC FRAMEWORK

We use the WCET-aware C compiler framework WCC [5] as basis for our optimization. The main advantage of it is that the estimation of the program’s WCET performs efficiently, since WCC is tightly coupled to a static WCET analyzer, the tool atT [1].

The considered C programs are annotated with flow facts within the ANSI-C source code. This data provides information about the code structure, such as the number of loop iterations or recursion depths, and is mandatory for a static WCET analysis. Obtaining these bounds is out of scope of this work, thus we consider them as given.

The formulation of the problem and the execution of the different Evolutionary Multi-Objective (EMO) algorithms are fully integrated within WCC. The process starts with defining the type of decision variables together with objective functions according to the modeling problem. Next, the model is passed to the evolutionary algorithm which creates an initial population based on the information provided by the model. In the next step, the EMO algorithm executes, generating the prescribed number of generations, to compute an approximated Pareto front.

4. OPTIMIZATION METHOD

Most real-world optimization problems are Multi-objective Optimization Problems (MOPs). In case of code generation for embedded real-time systems, a trade-off between WCET, code size or energy consumption has to be taken into account.

For instance, the well-known compiler optimization Function Inlining can be used to minimize WCET, but one of the drawbacks of the optimization is the increased code size. In this case, WCET and code size can be considered as two objective functions, which are to be minimized.

Compared with Single-Objective Optimization Problems, MOPs are more complex and difficult, they have a group of solutions, since an ideal multi-objective solution simultaneously optimizing each objective usually does not exist. Without loss of generality, MOPs can be formulated as follows:

\[
\begin{align*}
\min & \quad F(X) = (f_1(X), f_2(X), \ldots, f_s(X)) \\
\text{s.t.} & \quad g_k(x) \leq 0, k = 1, 2, \ldots, m, \\
& \quad h_j(x) = 0, j = 1, 2, \ldots, l,
\end{align*}
\]

where \( X = (x_1, x_2, \ldots, x_n)^T \) is the vector of decision variables, \( f_t(X), t = 1, 2, \ldots, s \) is the objective function, \( g_k(x) \) and \( h_j(x) \) are the constraint conditions.

If all objective functions are to be minimized, then we say that for two vectors of decision variables \( X \) and \( Y \), \( X \) dominates \( Y \), expressed by \( X \preceq Y \), if \( f_t(X) \leq f_t(Y) \), \( t \in \{1, 2, \ldots, s\} \) and \( \exists r \in \{1, 2, \ldots, s\} \) such that \( f_r(X) < f_r(Y) \). The dominance relation \( (X \preceq Y) \) is called Pareto dominance. Solutions that are not dominated by any other solution in the objective space are called Pareto optimal. The Pareto optimal set \( P^* \) and the Pareto optimal front \( PF^* \) are defined as follows:

\[
\begin{align*}
P^* &= \{X | X \text{ is Pareto optimal}\}, \\
PF^* &= \{F(X) | X \in P^*\}.
\end{align*}
\]

In practice, the number of Pareto optimal solutions is often too large and the determination of a single Pareto optimum might be even \( NP \) hard [8]. Additionally, a proof of optimality is computationally infeasible for many real-life problems. Therefore, the goal is to find a Pareto front approximation. The Pareto front approximation can be finally used to perform suitable optimization.

Randomized evolutionary multi-objective algorithms show good results for the approximation of Pareto fronts. Any evolutionary algorithm maintains a population of optimization sequences. After compiling the code, the objectives are determined and depending on their values, some sequences are selected for the next generation. Since in the case of the function inlining one is interested in a classification rule that decides if a particular function should be inlined or not for a particular call site, each sequence is encoded as a binary string where each bit denotes a specific call of a function.

Moreover, every evolutionary algorithm requires also information about the objective values when a particular optimization sequence is applied. Since we are interested in the worst-case behavior of real-time systems, the WCET has to be estimated for each generated machine code. This objective is provided by a static WCET analyzer which is tightly integrated into the WCC compiler. The analyzer does not run the program but performs static program analyses to estimate the WCET. This data is automatically made available to the compiler. The further objective, the code size, can be easily extracted from the binary executable. Furthermore, a map is utilized that holds the evaluated objective values for each considered optimization sequence to accelerate the evaluation of objectives. Whenever an objective of a sequence has to be determined that was already evaluated in the past, a costly re-evaluation is omitted and the objective value is obtained from the map.

In 2005, the third version of the Generalized Differential Evolution (GDE3) [9] was introduced by S. Kukkonen and J. Lampinen. It is currently one of the most suitable multi-objective evolutionary algorithms [3]. However, the standard GDE3 cannot be used for solving the binary-coded optimization problems directly, since the standard mutant operator generates real-coded vectors not bit strings. L. Wang et al. proposed the Modified Binary Differential Evolution Algorithm (MBDE) [16] for single-objective optimization problems to tackle this issue. The formulas of the standard DE [15], including the mutation, crossover and selection operators, are preserved in MBDE. However, in MBDE, the
probability estimation operator $P$ which is defined as

$$P(px_i) = (px_{i,1}, px_{i,2}, \ldots, px_{i,n})$$

is additionally used to construct the probability distribution model of new solutions as follows:

$$P(px_{i,j}^{G+1}) = \frac{1}{1 + e^{(2b \times (MO-0.5))/(1+2F)}},$$

$$MO = px_{r1,j}^G + F \cdot (px_{r2,j}^G - px_{r3,j}^G),$$

where $F$ is the scaling factor; $px_{r1,j}, px_{r2,j}$ and $px_{r3,j}$ are the $j$-th bits of three randomly chosen individuals; $b$, called bandwidth factor, is a positive real constant. The main advantage of this approach that the mutant operator of the standard DE represented as

$$C(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \preceq b\}|}{|B|},$$

$$\Delta_k = \frac{d_f + d_l + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_f + d_l + (N-1)d},$$

$$NR_k = \frac{|A_k \cap D|}{|D|},$$

where the dominance relation ($\preceq$) is called weak Pareto dominance and expressed by $a \preceq b$, if $f_i(a) \leq f_i(b)$, $\forall i \in \{1, 2, \ldots, s\}$; the parameters $d_f$ and $d_l$ are the Euclidean distance between the extreme solutions and the boundary solutions of the non-dominated set. To obtain $\bar{d}$, we calculate first the Euclidean distance $d_i$ between consecutive solutions in the obtained non-dominated set of solutions assuming that there are $N$ solutions on the best non-dominated front. Then, the average $\Delta$ of these distances is calculated. The set $A_k, k = 1, 2, \ldots, M$ is the approximate Pareto-optimal solution set of the $k$-th algorithm, $M$ is the total number of optimization algorithms used to solve the problem, and $D$ is the set of non-dominated solutions of the mixed Pareto-optimal solutions found by all algorithms.

The algorithms were run 50 times for every benchmark. The coverage, spacing and NR results are given in Tables 2-4. Table 2 shows that MBPOA has better performance on the coverage. However, the difference is not so significant. According to Table 3 GDE3 with the probability operator and MBPOA show the same results in spacing. From Table 4 we can see that both algorithms have the same nondominance ratio.

The runtime of both algorithms is almost the same, e.g. for one optimization run computing 10 generations with the population size that equals to 10, MBPOA takes 41.4s and 39.2m for minver and adpcm_decoder, respectively. For the same benchmarks the runtime of binary GDE3 is 45.6s and 37.1m, respectively.

At this point we cannot state that both algorithms show the same results, since more detailed analysis of the control parameters is necessary.
6. CONCLUSIONS

In this paper, a framework for the multi-criteria optimization in the WCET-Aware C Compiler Framework WCC has been presented.

We demonstrate how multi-objective evolutionary algorithms can be exploited for the construction of WCET- and code size aware function inlining optimization heuristic. For this purpose, function inlining was formulated as a binary bi-objective problem. In order to find the approximated Pareto front of the problem, the binary evolutionary algorithm MBPOA and the GDE3 algorithm with the probability operator mapping a real-coded sequence to a binary-coded string was used.

Since we expect that both algorithms are sensitive to the control parameters, we are going to perform parameter analysis to understand the influence of the control parameters to the solutions and adapt them depending on the problem. Moreover, we plan to extend our approach by considering other optimization techniques and adding additional design goals such as energy consumption or minimal chip aging.

We also plan to focus on studying which multi-objective algorithms are actually the most suitable ones for the compiler-based optimization of hard real-time systems.

Acknowledgments

This work received funding from Deutsche Forschungsgemeinschaft (DFG) under grant FA 1017/3-1.

7. REFERENCES


