Multi-Objective Optimization for the Compiler of Real-Time Systems based on Flower Pollination Algorithm

Shashank Jadhav, and Heiko Falk
Institute of Embedded Systems
Hamburg University of Technology
Germany
shashank.jadhav|heiko.falk@tuhh.de

ABSTRACT

Real-time systems usually face stringent constraints such as execution time, energy consumption, code-size, etc. Performing multi-objective optimization at compile time is one way to find approximations over the possible solutions which fulfill these constraints. Flower pollination algorithm (FPA) is a relatively recently proposed metaheuristic algorithm which makes use of the evolutionary characteristics of flower pollination process to find solutions to an optimization problem. In this paper, we propose a theoretical framework for an extension for the WCET-Aware C Compiler (WCC) framework [2] for performing multi-objective optimizations based on the FPA during compile time.

CCS Concepts

• Theory of computation → Discrete optimization;
• Software and its engineering → Compilers;
• Computer systems organization → Embedded systems;

Keywords

Compiler, Multi-Objective, Optimization, Flower Pollination Algorithm

1. INTRODUCTION

A modern embedded hard real-time system as the name suggests has to perform specific tasks within certain time constraints. Furthermore, based on the application and specifications of this embedded system, constraints like energy consumption, code size, code-security, etc also apply. In a real-world scenario, we will always face a problem where we have to design an embedded system which has to fulfill multiple objectives and these objectives often conflict with each other. One of the important properties of these systems is Worst-Case Execution time (WCET). The WCET is defined as the worst possible execution time of a program, independently from its input data. Optimizing for the WCET might lead to an increase in the Average-Case Execution Time (ACET) of that program, which in turn lead to overall increased energy consumption. We have to take into account these various factors while developing for these systems, which can be a strenuous job. If a hard real-time system does not complete a task within given constraints it can lead to disasters.

Currently, WCC - the WCET-Aware C compiler is equipped with different optimizations to deal with issues like WCET, energy, schedulability, code-size, etc. WCC uses Integer-Linear Programming (ILP) based approaches to deal with these optimizations. Within WCC, an ILP model is used to deterministically provide an optimal solution which fulfills given constraints.

WCC is also equipped to deal with multi-core systems [8] and multi-tasking systems [4]. There are ongoing efforts in the direction of an Evolutionary algorithm based approach to deal with multi-objective optimizations within WCC for high-level C-like intermediate representation. The ILP-based approach is quick for the ILP models with a certain level of complexity. But, for an optimization problem with a lot of constraints and multiple contradicting objectives, there might be solutions, but they are difficult to find with such a complex ILP. So, it can lead to computational infeasibility due to the exponential complexity of ILP. On the other hand, the genetic algorithm based approach will be able to approximate optimal solutions in such scenarios but the time required is much higher comparatively.

Deviating from the ILP-based and evolutionary algorithm based approaches, in this paper, we propose the Flower Pollination Algorithm [12] based approach within WCC and extend the WCC framework to deal with multi-objective optimizations. The overall goal is to develop a flexible base for multi-objective optimizations within WCC based on FPA. FPA is a metaheuristic algorithm inspired by the natural process of flower pollination and uses its evolutionary characteristics for solving complex optimization problems. We have coupled the multi-objective and binary extensions of FPA together in this paper and try to explore a binary solution space for minimizing multiple objectives using FPA.

In this paper, we have formulated the algorithm while keeping in mind the so-called static SPM-Allocation, which is a common compiler optimization technique. A Scratchpad Memory (SPM) is small but very fast compared to flash memory. The objectives that we have taken into consideration while formulating our multi-objective optimization problem, are WCET and Energy consumption. The optimization is not limited by the choice of objectives taken into consideration. Furthermore, we describe our current ongoing efforts on how FPA based approach could be ex-
tended to other compiler optimization techniques too, based on the objectives and the goal which needs to be achieved from a certain code optimization. In the original paper from Yang [12], FPA is shown to be more efficient than genetic algorithms and has a good convergence rate. Therefore, by using this FPA based approach we hope to improve the optimization time compared to the evolutionary algorithm based approach.

This paper is organized as follows: Section 2 provides an overview of the related work. Section 3 briefly discusses the WCC within which the FPA framework is integrated. In Section 4 we define a multi-objective optimization problem for SPM-Allocation. Section 5 discusses the FPA and explains FPA based multi-objective framework. A summary and a brief discussion of future prospects close the paper.

2. RELATED WORK

Performing a multi-objective optimization for contradicting objectives is a challenging task let alone within a compiler framework for real-time systems on an arbitrary source code. Efforts have been made by considering multiple objectives using evolutionary algorithms within a compiler for real-time systems. Lokuciejewski et al. [3] presents a stochastic evolutionary approach to identify Pareto optimal compiler optimization sequences for a different pair of objectives. [6] presents a multi-criteria optimization framework based on multi-objective binary probability optimization algorithm (MBPOA) and generalized differential evolution (GDE3).

Flower Pollination Algorithm is a metaheuristic algorithm first proposed by Yang [12] inspired by the pollination process seen in flowers. Yang showed that FPA is very efficient and can outperform evolutionary algorithms. A multi-objective extension of FPA was proposed by Yang et al. [13] displaying the simplicity and flexibility of FPA towards multiple objectives. Rodrigues et al. [10] presents a binary extension of FPA with its application to feature selection.

3. WCC: THE WCET-AWARE C COMPILER FRAMEWORK

The WCET-Aware C Compiler (WCC) is a C compiler consisting of sophisticated WCET-oriented analysis and optimizations, which forms the basis of the optimization carried out in this paper. WCC currently supports ARM7TDMI architecture and Infineon TriCore TC1796 and TC1797 processors as well. WCC has two intermediate representations one high-level C-like intermediate representation (ICD-C) and another assembler-like low-level intermediate representation (ICD-LLIR). Different optimizations can be carried out at both levels of representations. WCC offers optimizations for worst-case execution time during the compilation process due to its tight integration with a static WCET analyzer tool called aiT

WCC is also coupled with an energy analysis module that attaches energy consumption information at the ICD-LLIR [11]. WCC is also integrated with Synopsys’s cycle-true instruction set simulator, i.e., COMET. COMET comprises of virtual system prototypes for various target platforms with which we can simulate complex embedded systems within WCC and emit average-case execution time and average-case execution count. This data is annotated to the assembly level within the LLIR. Once the average-case execution data is available at the LLIR level we use the energy data to obtain the total energy consumption. We use aiT and the energy model within WCC to carry out the WCET and Energy analysis, respectively during our optimization.

4. PROBLEM STATEMENT

In this section, we define a multi-objective optimization with constraints based on SPM-Allocation. A Scratchpad Memory (SPM) is small but very fast compared to flash memory and can provide us with the opportunity to optimize the WCET and Energy consumption of a program by moving parts of a program called basic blocks from slow flash memory to SPM. While performing SPM-Allocation in the quest of optimizing WCET we might end up increasing energy consumption and vice versa. Therefore, in our case, we consider WCET and Energy consumption as our objective functions, which we are minimizing. SPM-Allocation is just one example of base optimization which can be used to demonstrate the FPA based optimization framework. The choice of objectives for minimization and the choice of base optimization depends on the goal which needs to be achieved by performing the FPA based multi-objective optimization.

A multi-objective optimization problem is formulated as,

\[
\min_x f(x) = (f_1(x), f_2(x), \ldots, f_s(x))
\]

subject to \( g_k(x) \leq 0, k = 1, 2, \ldots, r \),

\( h_k(x) = 0, k = 1, 2, \ldots, l \).

where \( x = (x_1, x_2, \ldots, x_d)^T \) is a decision variables vector, \( f_o(x), o = 1, 2, \ldots, s \) are objective functions, \( s \) is the total number of objectives, \( g_k(x) \) and \( h_k(x) \) are constraint conditions.

In context of our problem for SPM-Allocation, the decision variables vector \( x \) represents the decision of whether a basic block is placed in SPM or in flash memory, i.e., \( x = (x_1, x_2, \ldots, x_d)^T \) where \( x_j \in \{0,1\}, j = 1, 2, \ldots, d \) and \( d \) represents the total number of basic blocks.

With the help of SPM-Allocation we opt to minimize the WCET and the energy consumption of a given task set and the constraints that are implemented on our problem is the size of the SPM, i.e., \( \sum_j B_j x_j \leq S_{SPM}, j = 1, 2, \ldots, d \), where \( B_j \) is the size of the code in a basic block \( j \), and \( S_{SPM} \) represents the size of the SPM. Therefore,

\[
g(x) = \sum_j B_j x_j - S_{SPM} \leq 0
\]

which acts as the constraint on our optimization problem. While moving a basic block from flash memory to SPM or vice versa, we have to take into consideration the actual correction of jump instruction inside the assembly code to ensure the valid control flow [7]. SPM-Allocation alters the memory addresses of the succeeding basic block and rendering moot the previously valid jumps between the basic blocks. Within the WCC, we perform jump correction after moving the basic blocks during our optimization so that the control flow is not broken.

5. FLOWER POLLENATION ALGORITHM BASED OPTIMIZATION FRAMEWORK

The flower pollination algorithm [12] is based on the flow pollination process seen in the flowering plants. Within nature the flower pollens can be carried far away from the
Algorithm 1: Pseudo code of the proposed Flower Pollination Algorithm based Optimization Framework within WCC

1. Objectives \( \min f(x) = (f_1(x), f_2(x)) \) where \( f_1(x) = WCET(x) \) and \( f_2(x) = Energy(x) \).

2. for \( i = 1 : n \) do
   3. Initialize a flower by placing basic blocks in SPM or Flash.
   4. while \( \sum_{j=1}^{d} B_j x_{ij} - S_{SPM} \geq 0 \) do
      5. Repair solution by removing a basic block from SPM.
   6. end
   7. Perform jump correction for the initial population.
   8. end
   9. Evaluate the initial population by doing WCET and Energy analysis.
10. Calculate the composite single objectives. Find the best solution \( g_0^* \) in the initial population.
11. Define the switch probability \( p \in [0, 1] \).
12. Define the decision maker or the stopping criteria such as \( m \) - maximum number of generations/iterations.
13. while \( t < m \) do
   14. for \( i = 1 : n \) do
      15. if \( rand < p \) then
         16. Draw \( L \) obeying a Lévy distribution.
         17. Global pollination using the update equation;
            \[ S(x_i^{t+1}) = x_i^t + \gamma L(\lambda)(g_i^* - x_i^t) \]
      else
         18. Draw \( \epsilon_1 \) from a uniform distribution in \([0, 1]\).
         19. Find \( x_{N_i}^t \) and \( x_{N_2}^t \).
         20. Local pollination using the update equation;
            \[ S(x_i^{t+1}) = x_i^t + \epsilon_1(x_{N_i}^t - x_{N_2}^t) \]
      end
     21. end
   22. for \( j = 1 : d \) do
      23. Calculate \( P(S(x_{ij}^t)) \)
      24. Assign binary value to \( x_{ij}^t \)
   25. end
26. Check if it satisfies SPM constraints.
27. If not, discard this solution without performing next evaluation step.
28. Perform jump correction.
29. Evaluate the updated flower by doing WCET and Energy analysis.
30. Calculate the composite single objectives.
31. If the updated flower has better objectives update them in the new generation.
32. Find the best solution \( g_i^* \) in the updated population.
33. end
34. Output the pareto optimal solutions from the final population.
a Lévy distribution [9],

\[ L \sim \frac{\gamma(\lambda) \sin(\pi \lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}, \quad (s \gg a_0 > 0), \]

where \( \Gamma(\lambda) \) is a standard gamma function, \( \lambda \) is a positive integer, and the above-mentioned distribution is valid for large steps \( s > 0 \). One of the ways for drawing such random numbers is so-called Mantegna algorithm [5] for drawing step size \( s \) by using two Gaussian distributions.

The update equation which represents the local pollination can be represented mathematically as

\[ S(x_i^{t+1}) = x_i^{t} + \epsilon_1(x_{N_1}^t - x_{N_2}^t), \quad (5) \]

where \( S(x_i^t) \in \mathbb{R}^d \), \( x_{N_1}^t \) and \( x_{N_2}^t \) are any two flowers from the local neighbourhood of \( x_i^t \) and \( \epsilon_1 \) is drawn from a uniform distribution in \([0, 1]\). This update equation mimics the behaviour of a local random walk.

As we deal with the binary decision vector we convert the updated solution that lies in the continuous-valued search space \( S(x_i^{t+1}) \in \mathbb{R}^d \) to a binary decision vector \( x_i^t \in \{0, 1\}^d \) (Lines 23-26). In our optimization problem, we want a binary solution vector where we make the decision of putting the basic block of code in SPM or in the Flash memory. Therefore, the new solution will be restricted to binary values after updating the flower population. The search space for our optimization is restricted to the d-dimensional boolean lattice. Let

\[ P(S(x_i^t)) = \frac{1}{1 + \exp^{-S(x_i^t)}} \quad (6) \]

\[ x_{ij}^t = \begin{cases} 1, & \text{if } P(S(x_i^t)) > \sigma, \\ 0, & \text{otherwise} \end{cases} \quad (7) \]

where \( \sigma \sim U(0, 1) \) is the sample drawn from a uniform distribution between the interval \([0, 1]\) and \( j = 1, 2, \ldots, d \).

We again check if it satisfies SPM constraint, if not, we discard the solution or perform jump correction on the newly found solution. (Line 27-29). Lines 30-31 is the evaluation of the newly updated individual by doing WCET and Energy analysis and calculation of composite single objectives. Further, we update the new individual within the population if it is better than the previous individual. Finally, we again find the best solution from the updated population to use it for the next generation.

6. SUMMARY AND FUTURE WORK

In this paper, we proposed a framework for multi-objective optimization within the WCET-aware C Compiler WCC based on Flower pollination algorithm. We have shown how FPA can be exploited to perform WCET and Energy-aware basic compiler optimization such as SPM-Allocation.

Since, Yang et al. [13] claims that FPA is very efficient and outperforms evolutionary algorithms, we plan to carry out extensive evaluations to test the efficiency of FPA based framework against an evolutionary algorithm [14] within WCC framework. Furthermore, we plan to exploit other compiler optimizations and objectives functions to test the flexibility of the FPA based framework.

Acknowledgments

This work is part of a project that has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 779882.

7. REFERENCES