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-

SCOPES '19, May 27–28, 2019, Sankt Goar, Germany

 ISBN 978-1-4503-6762-2/19/05…$15.00
DOI: https://doi.org/10.1145/3323439.3323977

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* [1].

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,

$$\begin{aligned} \min_{x} \quad & f(x) = (f_1(x), f_2(x), \ldots, f_s(x)) \\ \text{subject to} \quad & g_{k_1}(x) \leq 0, k_1 = 1, 2, \ldots, r, \\ & h_{k_2}(x) = 0, k_2 = 1, 2, \ldots, l. \end{aligned} \tag{1}$$

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 fuctions, s is the total number of objectives, $g_{k_1}(x)$ and $h_{k_2}(x)$ 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,

$$g(x) = \sum_j B_j x_j - S_{SPM} \leq 0 \tag{2}$$

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 POLLINATION 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 base 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 >= 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_*^t in the initial population.
11 Define the switch probability p ∈ [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 + γL(λ)(g_*^t - x_i^t)
18         else
19             Draw ε_1 from a uniform
                 distribution in [0, 1].
20             Find x_{N_1}^t and x_{N_2}^t.
21             Local pollination using the
                 update equation;
               S(x_i^{t+1}) = x_i^t + ε_1(x_{N_1}^t - x_{N_2}^t)
22         end
23         for j = 1 : d do
24             Calculate P(S(x_ij^t))
25             Assign binary value to x_ij^t
26         end
27         Check if it satisfies SPM constraints.
28         If not, discard this solution without
             performing next evaluation step.
29         Perform Jump correction.
30         Evaluate the updated flower by doing
             WCET and Energy analysis.
31         Calculate the composite single
             objectives.
32         If the updated flower has better
             objectives update them in the new
             generation.
33     end
34     Find the best solution g_*^t in the updated
         population.
35 end
36 Output the pareto optimal solutions from the final
     population.
```

neighborhood of a flower by the pollinators and pollination can occur at much longer distances. In FPA, the pollinators obey Lévy flight behavior, i.e., with jumping and flying over long distant steps obeying a Lévy distribution [9], for global pollination mirroring the biotic cross-pollination process. Within a certain neighborhood of the flowers, we can observe a certain similarity between the flowers which is reflected in the local pollination. The local pollination and the global pollination in FPA is controlled by a switch-probability $p \in [0, 1]$. As the probability of local pollination happening in nature is higher than the global pollination, the switch-probability in FPA is also slightly biased towards local pollination.

Algorithm 1 presents the proposed binary multi-objective flower pollination algorithm for SPM-Allocation within the WCC. Let $x_i^t \in \{0, 1\}^d$ where $i = 1, 2, \ldots, n$ be a d-dimensional flower, i.e., a binary solution vector at iteration $t = 1, 2, \ldots, m$, where d represents the total number of basic blocks, n is total number of flowers in t^{th} generation and m represents the maximum number of generations. Lines 1-8 initialize the objectives and the initial population which satisfies the SPM size constraints. If the SPM size constraint is not satisfied, we repair the population by removing basic blocks from SPM until the constraint is satisfies.

Line 9 is the evaluation of the initial population by conducting WCET and Energy analysis within WCC. As from section 4, we can see that our optimization problem is a multi-objective problem where minimizing WCET and Energy being those objectives, finding the best solution $\mathbf{g}_*^t$ from a population iteration t becomes a difficult task. One of the simplest approach used is to take weighed sum of multiple objectives and combine them into a single composite objective, i.e.,

$$f_c = \sum_{o=1}^{s} w_o f_o, \text{ where } \sum_{o=1}^{s} w_o = 1, \text{ and } w_o > 0, \quad (3)$$

where f_c represents the composite single objective, s is the total number of objectives and w_o are non-negative weights. After calculating a composite single objective for each flower we can find the best solution $\mathbf{g}_*^t$ out of the population at each iteration. Line 10 calculates a composite single objective f_c for each individual in the population as we are dealing with multiple objectives, based on which we choose the best solution $\mathbf{g}_*^t$ from the initial population. We also define the switch-probability $p \in [0, 1]$ (Line 11) to switch between the local pollination and the global pollination and define stopping criteria (Line 12), here we are using the maximum number of iterations/generations as the stopping criteria.

Lines 13-35 is the main loop of the proposed optimization process. For each generation/iteration of the population and for each flower within the population we choose to do global or local pollination (Lines 15-22) based on the switch-probability p. The update equation for global pollination can be represented mathematically as

$$S(x_i^{t+1}) = x_i^t + \gamma L(\lambda)(\mathbf{g}_*^t - x_i^t), \quad (4)$$

where $S(x_i)^t \in \mathbb{R}^d$, and γ is a scaling factor to control the step size during the update to the next generation. $L(\lambda)$ corresponds to the Lévy flights' based step size, that corressponds to the strength of the pollination, which mimic the characteristics of the pollinators moving over a long distance with various long distance steps. $L > 0$ is drawn from

a Lévy distribution [9],

$$L \sim \frac{\gamma\Gamma(\lambda)\sin(\pi\lambda/2)}{\pi}\frac{1}{s^{1+\lambda}}, \quad (s \gg s_0 > 0),$$

where $\Gamma(\lambda)$ is a standard gamma function, λ 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

$$S(x_i^{t+1}) = x_i^t + \epsilon_1(x_{N_1}^t - x_{N_2}^t), \tag{5}$$

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 ϵ_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

$$\mathcal{P}(S(x_{ij}^t)) = \frac{1}{1 + \exp^{-S(x_{ij}^t)}} \tag{6}$$

$$x_{ij}^t = \begin{cases} 1, & \text{if } \mathcal{P}(S(x_{ij}^t)) > \sigma, \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

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

[1] AbsInt Angewandte Informatik, GmbH. aiT Worst-Case Execution Time Analyzers, 2018.

[2] H. Falk and P. Lokuciejewski. A Compiler Framework for the Reduction of Worst-Case Execution Times. *Real-Time Systems*, 46(2):251–298, 2010.

[3] P. Lokuciejewski, S. Plazar, H. Falk, P. Marwedel, and L. Thiele. Approximating Pareto optimal compiler optimization sequences - a trade-off between WCET, ACET and code size. *Softw., Pract. Exper.*, 41:1437–1458, 2011.

[4] A. Luppold and H. Falk. Schedulability-Aware SPM Allocation for Preemptive Hard Real-Time Systems with Arbitrary Activation Patterns. In *Proceedings of Design, Automation and Test in Europe (DATE)*, pages 1074–1079, Lausanne / Switzerland, March 2017.

[5] R. N. Mantegna. Fast, accurate algorithm for numerical simulation of levy stable stochastic processes. *Physical Review E*, 49(5):4677, 1994.

[6] K. Muts, A. Luppold, and H. Falk. Multi-criteria compiler-based optimizations of hard real-time systems. In *In Proc. of SCOPES*. ACM, May 2018.

[7] D. Oehlert, A. Luppold, and H. Falk. Practical challenges of ilp-based spm allocation optimizations. In *Proceedings of the 19th International Workshop on Software and Compilers for Embedded Systems*, pages 86–89. ACM, 2016.

[8] D. Oehlert, A. Luppold, and H. Falk. Bus-aware Static Instruction SPM Allocation for Multicore Hard Real-Time Systems. In *Proceedings of the 29th Euromicro Conference on Real-Time Systems (ECRTS)*, Dubrovnik / Croatia, June 2017.

[9] I. Pavlyukevich. Lévy flights, non-local search and simulated annealing. *Journal of Computational Physics*, 226(2):1830–1844, 2007.

[10] D. Rodrigues, X.-S. Yang, A. N. De Souza, and J. P. Papa. Binary flower pollination algorithm and its application to feature selection. In *Recent advances in swarm intelligence and evolutionary computation*, pages 85–100. Springer, 2015.

[11] M. Roth, A. Luppold, and H. Falk. Measuring and modeling energy consumption of embedded systems for optimizing compilers. In *Proceedings of the 21st International Workshop on Software and Compilers for Embedded Systems*, pages 86–89. ACM, 2018.

[12] X.-S. Yang. Flower pollination algorithm for global optimization. In *International conference on unconventional computing and natural computation*, pages 240–249. Springer, 2012.

[13] X.-S. Yang, M. Karamanoglu, and X. He. Flower pollination algorithm: a novel approach for multiobjective optimization. *Engineering Optimization*, 46(9):1222–1237, 2014.

[14] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.