# The Explicit Preemption Placement Problem for Real-Time Conditional Code

Marko Bertogna[1] and Nathan Fisher[2]

[1]Scuola Superiore Sant'Anna, Pisa, Italy, marko.bertogna@sssup.it
[2]Department of Computer Science, Wayne State University, Detroit, Michigan, USA, fishern@cs.wayne.edu

## I. Introduction

In real-time worst-case execution time (WCET) analysis, an upper bound is calculated, for each job in the system, on the total aggregate amount of execution required to successfully complete the job. Real-time schedulability analysis has traditionally used the estimates determined from WCET analysis to determine whether every job in a system can be completed by its deadline. Thus, the effectiveness of the resulting schedulability analysis hinges upon the precision of WCET estimates. Unfortunately, many of scheduler properties that simplify schedulability analysis often introduce pessimism into WCET analysis. For example, the oft-assumed property that jobs are arbitrarily preemptible leads to significant increase in the WCET estimates, as the analysis must assume that a preemption occurs often and the overhead of such preemption (due to context switch time and cache effects) must be added to the estimate. Furthermore, most real-time scheduling algorithms and associated schedulability analysis do not take the heterogeneous "cost" of preemption overhead into account when making scheduling decisions. For instance, preemption of a job may cause cache lines to be invalidated that are needed in subsequent instructions; the memory access pattern of nearby instructions will greatly influence the cost of the preemption due to such invalidations. Thus, a better strategy may be to delay the preemption of a job that is executing instructions with a degree of spatial or temporal locality (in terms of memory access) until it reaches instructions with a lower level of memory locality.

Very recently, Bertogna et al. [1] (in ECRTS 2011) proposed such an approach that explicitly and efficiently determines (prior to runtime) the optimal choice of *explicit preemption points* (EPPs) in a job's code that minimize the preemption overhead while ensuring that system schedulability is not affected due to increased non-preemptivity. However, their proposed approach only deals with linear (non-branching) code and cannot handle jobs with control flow such as conditional statements (e.g., if-then-else statements) and loops. In this abstract, we propose specific open problems towards extending the EPP approach of Bertogna et al. [1] for handling general conditional code. Furthermore, our objective is to obtain a solution that retains the efficient running time of the non-branching version of the problem. We believe that such extensions are absolutely necessary for the EPP approach to be widely applicable and useful to a real-time system designer.

## II. Model

We refer to the problem of determining the optimal choice of EPPs for a program (i.e., job) as the *explicit preemption placement* problem. To model the explicit preemption placement problem, we assume that a program $\mathcal{P}$ has been divided into a set of non-preemptive basic blocks (BBs). (Currently, [1] assumes that any conditional code is entirely contained within a single BB). A directed graph $G_{\mathcal{P}} = (V, E)$ describes the control flow of $\mathcal{P}$. Each vertex $v \in V$ represents a BB in $\mathcal{P}$. An edge $(u, v) \in E \subseteq V \times V$ means that the execution of BB $u$ immediately precedes the execution of BB $v$ in some execution path of $\mathcal{P}$, and that a preemption is permitted between the two BBs, i.e., $E$ is the set of possible EPPs. We assume that there is a special vertex $s$ that indicates the initial BB of the $\mathcal{P}$. Similarly, there is a special vertex $z$ that is the terminating BB of $\mathcal{P}$, i.e., that has no successor BB. A path $p$ is an ordered set of consecutive vertices, such that each vertex in $p$ has an edge from its predecessor. Let $\mathsf{paths}$ be the set of possible execution paths from the initial BB $s$ to the terminating BB $z$.

For the purposes of quantifying the preemption overhead of selecting an EPP, we assume that a function $\xi : E \mapsto \mathbb{R}_{\geq 0}$ is given. Similarly, the WCET of a BB is given by a function $C : V \mapsto \mathbb{R}_{\geq 0}$. Finally, since the selection of EPPs will create non-preemptible regions in $\mathcal{P}$, the schedulability of the system is affected by a choice of EPPs. Thus, we will assume that a constant $Q$ is determined which quantifies the maximum duration of any non-preemptive region in $\mathcal{P}$.

Note that since every structured program can be expressed as a combination of sequential instructions, conditional branches and loops, the adopted model is general enough to express every real-time task implemented with a structured programming language.

## III. Open Problems

Given the above model, our goal is to find a selection of EPPs that minimize the WCET of $\mathcal{P}$. More formally, our main open problem is:

Given $G_{\mathcal{P}}$ and associated functions $\xi$ and $C$, find $S \subseteq E$ that minimizes

$$\max_{p \in \mathsf{paths}} \left\{ \sum_{u \in p} C(u) + \sum_{\substack{u,v \in p \\ (u,v) \in S}} \xi(u,v) \right\} \quad (1)$$

subject to the constraint that, for each path in paths and for any two $u, v \in p$, if the total execution of BBs from $u$ to $v$ in path $p$ is greater than $Q$ then some edge of $p$ must be in $S$.

To solve the above general problem, we must answer the following subproblems:

**SP1** *Given conditional, non-looping code, can a solution be determined in a time polynomial in the number of EPPs?* Systems without loops, or with loops entirely contained inside a BB, can be modeled using Directed Acyclic Graphs (DAGs), simplifying the problem. However, no optimal method is known even for this simplified problem, due to the presence of conditional branches. In fact, the selection of EPPs for one path of the DAG might be conflicting with the selection for other paths that share some vertex.

**SP2** *How do loops affect the analysis?* As it is common in the timing analysis domain, we are only interested in loops that have a deterministic number of iterations.

## REFERENCES

[1] M. BERTOGNA, O. XHANI, M. MARINONI, F. ESPOSITO, AND G. BUTTAZZO. Optimal Selection of Preemption Points to Minimize Preemption Overhead. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Porto, Portugal, July 2011. IEEE Computer Society Press.