35

Dual Criteria Optimization Problems for Imprecise Computation Tasks

Kevin I-J. Ho
Chung-Shan Medical University

35.1 Introduction

In real-time scheduling problems, one of the paramount criteria is meeting the deadlines of all the tasks. Unfortunately, it is impossible to satisfy this criterion all the time, especially in heavily loaded real-time systems. To cope with this situation, one can completely discard some less critical tasks in favor of meeting the deadlines of more important ones. Or, one can schedule tasks partially, i.e., executing tasks not completely, so as to make more tasks receive minimum required execution time and meet their deadlines. The latter approach is applicable to situations such as the numerical analysis and the curing process of composite products. To model this kind of task system, Lin et al. [1–3] proposed the *Imprecise Computation Model*, where the accuracy of the results can be traded for meeting the deadlines of the tasks. For details of the model, the readers are referred to the previous chapter and [4]. In this model, each task can be logically decomposed into two subtasks, mandatory and optional. For each task, the mandatory subtask is required to be completely executed by its deadline to obtain acceptable result, while the optional subtask is to enhance the result generated by the mandatory subtask and hence can be left incomplete.

1-58488-397-9/$0.00+$1.50

If the optional subtask of a task is left incomplete, an error is incurred, which is defined as the execution time of the unfinished portion. Task systems in the traditional scheduling model can be treated as a special case where the execution time of each task's optional subtask is zero.

In the Imprecise Computation Model, each task T_i in the task system TS is symbolized by the quadruple (r_i, d_i, m_i, o_i), where r_i, d_i, m_i, and o_i denote its release time, deadline, mandatory subtask's execution time and optional subtask's execution time, respectively. Let e_i denote the total execution time of task T_i, i.e., $e_i = m_i + o_i$. In certain circumstances, the importance, regarding the criteria, of tasks may be different. To represent the difference among tasks, each task may be assigned one or more weights.

In this model, a *legitimate* schedule of a task system satisfies the following requirements: (1) no task is assigned to more than one processor and (2) at most one task is assigned to a processor at any moment of time. A legitimate schedule is said to be *feasible* if each task starts no earlier than its release time and finishes no later than its deadline, and the mandatory subtask is completely executed. A task system is said to be feasible if there exists at least one feasible schedule for it. The feasibility of a task system can be determined in $O(n^2 \log^2 n)$ time for parallel and identical processors [5] and $O(n \log n)$ time for single processor [6]. Without loss of generality, all the task systems discussed in this chapter are assumed to be feasible, since the time complexity of all the algorithms introduced in this chapter is higher than that of the feasibility test. Additionally, we assume that all the task parameters are rational, all the tasks are independent, and all the processors are identical. Furthermore, in this chapter, only preemptive schedules are considered.

Before we proceed, we define general terminology and notations to facilitate the reading. Let S be a feasible schedule for a task system TS with n tasks in the Imprecise Computation Model. For each task T_i in TS, let $\alpha(T_i, S)$ denote the amount of processor time assigned to T_i in S. The error of T_i in S, denoted by $\varepsilon(T_i, S)$, is defined as $e_i - \alpha(T_i, S)$. The total error of S, denoted by $\varepsilon(S)$, is defined as $\sum_{i=1}^{n} \varepsilon(T_i, S)$. The minimum total error of TS, denoted by $\varepsilon(TS)$, is defined as $\min\{\varepsilon(S) : S \text{ is a feasible schedule for } TS\}$. If the importance of the tasks with respect to a given criterion are not the same, then a positive value w_i is assigned to each task T_i. And the product of $\varepsilon(T_i, S)$ and w_i is called the *w-weighted* error of T_i in S, denoted by $\varepsilon_w(T_i, S)$. The total *w-weighted* error of S and minimum total *w-weighted* error of TS are defined analogously, and are denoted by $\varepsilon_w(S)$ and $\varepsilon_w(TS)$, respectively.

In the past, several researchers have studied the total unweighted and total weighted error problems in the Imprecise Computation Model; see details in Section 35.2. But, most of the studies are only concerned with the total unweighted or weighted error, without considering the distribution of errors among tasks. Therefore, the obtained schedules might result in very uneven error distribution among the tasks. Even worse, some important tasks have large errors, while less important tasks have small errors. This kind of schedule might not be suitable for some task systems.

In this chapter we will discuss, based on the research results done by Ho et at. [7,8], the problems related to the error distribution. We will focus on three dual-criteria optimization problems of preemptively scheduling a set of imprecise computation tasks on $m \geq 1$ identical processors. The criterion of the first optimization problem is the maximum weighted error of a given task system. The second problem is to minimize the total weighted error subject to the constraint that the maximum weighted error is minimized. The third one is to minimize the maximum weighted error subject to the constraint that the total weighted error is minimized. In these problems, each task may have two different weights, one for computing the total weighted error and the other for the maximum weighted error. In Section 35.2, we will briefly review some works related to the problems of minimizing total (weighted) error and the problems of error distribution. Then, the description of the problems on which this chapter focuses is given. In Section 35.3, we will study the problem of minimizing the maximum weighted error. Section 35.4 will focus on the problem of minimizing the total weighted error subject to the constraint that the maximum weighted error is minimized. The problem of minimizing the maximum weighted error under the constraint that the total weighted error is minimized will be discussed in Section 35.5. In the last section, we will make some concluding remarks for this chapter and discuss the relationship among the problems studied in Sections 35.4 and 35.5.

35.2 Related Works

In this section we will review some research results related to the topics discussed in this chapter. Blazewicz [9] was the pioneer of studying the problem of minimizing the total weighted error for the special case where each task has an optional subtask only, i.e., $m_i = 0$ for each $1 \leq i \leq n$. He reduced the problems of scheduling tasks on parallel and identical processors as well as uniform processors to minimum-cost-maximum-flow problems, which can then be transformed to linear programming ones. Then, by using the minimum-cost-maximum-flow approach again, Blazewicz and Finke [10] gave faster algorithms for both problems. Potts and van Wassenhove [11] proposed an $O(n \log n)$-time algorithm for the same problem on single processor, with the added constraint that all tasks have identical release times and weights. Moreover, they proved that the problem becomes NP-hard for the nonpreemptive case, and present a pseudo-polynomial time algorithm for the nonpreemptive case. In [12], Potts and van Wassenhove gave a polynomial approximation scheme and two fully polynomial approximation schemes based on the pseudo-polynomial time algorithm.

Now, consider the general case in which each task has mandatory, optional, or both subtasks. For parallel and identical processors, Shih et al. [5] proposed an $O(n^2 \log^2 n)$-time algorithm to minimize the total error. And they also showed that the weighted version problem can be transformed to a minimum-cost-maximum-flow problem. Using Orlin's $O(|A| \log |V| (|A| + |V| \log |V|))$-time algorithm for the minimum-cost-maximum-flow problem [13], where A and V denote the arc set and vertex set, respectively, the problem of minimizing the total weighted error on parallel and identical processors can be solved in $O(n^2 log^3 n)$ time, since in the network shown in [5], $|A|$ is equal to $O(n \log n)$ and $|V|$ is equal to $O(n)$. Regarding the single processor case, Shih et al [14] gave an algorithm that runs in $O(n \log\ n)$ time for the unweighted case and $O(n^2 \log\ n)$ time for the weighted case. Later, Leung et al. [15] gave a faster algorithm that runs in $O(n \log\ n + kn)$ time, where k denotes the number of distinct weights.

In 1991, Shih et al. [14] proposed a new constraint, the so-called *0/1-constraint*, to be added on the Imprecise Computation Model, where each optional subtask is either fully executed or completely discarded. The 0/1-constraint is motivated by some real world problems. For example, an algorithm for solving a task in the real world might have two different versions. One runs slower than the other, but produces a result with better quality. To meet the deadline constraints of tasks, some tasks might be forced to execute the fast version of the algorithm. By treating the time difference between the fast and slow versions as the execution time of the optional subtask, one can easily transform this type of problem into the problem of scheduling with 0/1-constraint [16]. For a single processor, Shih et al. [14] showed that the problem of minimizing the total error is NP-hard. Later, Ho et al. [17] showed that the problem can be reduced to the problem of minimizing the weighted number of late tasks in classical scheduling theory (denoted as the $1 \mid pmtn, r_j \mid \sum w_j U_j$ problem in the classification scheme of [18]). Lawler [19] proposed an $O(n^3 W^2)$-time algorithm for the $1 \mid pmtn, r_j \mid \sum w_j U_j$ problem, where W is the total weight of all the n tasks. Therefore, the problem of minimizing the total error on a single processor with 0/1-constraint can be solved in pseudo-polynomial time $O(n^5 \sigma^2)$, where σ denotes the total execution time of all the optional subtasks. Motivated by the computational complexity of the problem, Ho et al. also developed an $O(n^2)$-time approximation algorithm for it, with a worst-case performance bound of 3, which is tight.

All the research results stated above were only concerned with minimizing the total (weighted) error, without any regard to the distribution of errors among tasks. Thus, the error distribution of tasks might be biased in the schedules generated by all the algorithms mentioned above. In other words, some tasks have very large (weighted) errors while others have very small (weighted) errors, which might not be acceptable under certain circumstances. Motivated by this consideration, Shih and Liu [20] studied the problem of preemptively scheduling a task system TS on single processor to minimize the maximum *normalized* error, defined as $\max\{\varepsilon(T_i, S)/o_i : T_i \in TS\}$, under the constraint that the total error is minimum. An $O(n^3 \log n)$-time algorithm was given to solve this problem.

To generalize the problem proposed by Shih and Liu [20], Ho et al. [7] defined a *doubly weighted* task system, in which each task T_i is weighted with two weights, w_i^T and w_i^M, for two different criteria,

total w^T-*weighted* error and maximum w^M-*weighted* error, respectively. (Note that the normalized error defined in [20] can be interpreted as each task T_i having two weights, $w_i^T = 1$ and $w_i^M = 1/o_i$. And then, the problem studied in [20] is to minimize the maximum w^M-*weighted* error subject to the constraint that the total w^T-*weighted* error is minimized.) In [7], they considered the problems of preemptively scheduling a doubly weighted task system with two objectives: (1) minimizing the maximum w^M-*weighted* error and (2) minimizing the total w^T-*weighted* error under the constraint that the maximum w^M-*weighted* error is minimized. They also presented two algorithms for both objectives that run in $O(n^3 \log^2 n)$ for parallel and identical processors and $O(n^2)$ for single processor.

To continue the study of preemptively scheduling a doubly weighted task system, Ho et al. [8] considered the problem of minimizing the maximum w^M-*weighted* error under the constraint that the total w^T-*weighted* error is minimized. An algorithm was presented to solve this problem, with time complexity $O(kn^3 \log^2 n)$ for the parallel and identical processors case and $O(kn^2)$ for the single processor case, where k is the number of distinct w^T-*weights*. It is easy to verify that the problem studied by Shih and Liu [20] is a special case of the problem by letting $w_i^T = 1$ and $w_i^M = 1/o_i$ for each task T_i.

35.3 Minimizing Maximum Weighted Error

In this section we will focus on the problem of minimizing the maximum w^M-*weighted* error of a given doubly weighted task system. At the beginning we will formally define the *Doubly Weighted Task Systems*, and then the related terminology and notations used throughout this section. Let $TS = (\{T_i\}, \{r_i\}, \{d_i\}, \{m_i\}, \{o_i\}, \{w_i^T\}, \{w_i^M\})$, $1 \le i \le n$, be a doubly weighted task system consisting of n tasks, where w_i^T and w_i^M denote the two positive weights for computing the total weighted error and the maximum weighted error, respectively. Let S be an arbitrary feasible schedule for TS on $m \ge 1$ parallel and identical processors. The symbol $E_{w^M}(S)$ denotes the maximum w^M-*weighted* error of S (i.e., $E_{w^M}(S) = \max_{1 \le i \le n} \{\varepsilon_{w^M}(T_i, S)\}$) and the symbol $E_{w^M}(TS)$ denotes the minimum maximum w^M-*weighted* error of TS (i.e., $E_{w^M}(TS) = \min \{E_{w^M}(S) : S \text{ is a feasible schedule for } TS\}$). The symbol $\varepsilon_{w^T}^{w^M}(TS)$ denotes the minimum total w^T-*weighted* error under the constraint that the maximum w^M-*weighted* error is minimized, i.e., $\varepsilon_{w^T}^{w^M}(TS) = \min \{\varepsilon_{w^T}(S)$: S is a feasible schedule for TS and $E_{w^M}(S) = E_{w^M}(TS)\}$.

Let $\min_{1 \le i \le n}\{r_i\} = t_0 < t_1 < \cdots < t_p = \max_{1 \le i \le n}\{d_i\}$ be all the distinct release times and deadlines of all tasks in TS. These $p + 1$ distinct values divide the time frame into p intervals: $[t_0, t_1]$, $[t_1, t_2]$, ..., $[t_{p-1}, t_p]$, denoted by $I_1, I_2, \ldots, I_p$. The length of the interval I_j, denoted by L_j, is equal to $t_j - t_{j-1}$. By using McNaughton's rule [21], a schedule S for TS can be described by a $n \times p$ matrix SM_S such that $SM_S(i, j)$ represents the amount of processor time assigned to T_i in I_j. If S is a feasible schedule, SM_S must satisfy the following inequalities: (1) $SM_S(i, j) \le L_j$ for $1 \le i \le n$ and $1 \le j \le p$; (2) $\sum_{i=1}^{n} SM_S(i, j) \le m \times L_j$ for $1 \le j \le p$. In the remainder of this chapter, we will represent schedules using the matrix forms to facilitate the reading, if necessary.

An interval I_j is said to be *unsaturated* if $\sum_{i=1}^{n} SM_S(i, j) < m \times L_j$; otherwise, it is *saturated*. A task T_i is said to be *available* in time interval $I_j = [t_{j-1}, t_j]$ if $r_i \le t_{j-1} < t_j \le d_i$. A task T_i is said to be *fully scheduled* in I_j if $SM_S(i, j) = L_j$; otherwise, it is said to be *partially scheduled*. A task is said to be *precisely scheduled* in S if $\varepsilon(T_i, S) = 0$; otherwise, it is said to be *imprecisely scheduled*.

The remainder of this section is organized as follows. A polynomial-time preemptive scheduling algorithm, called Algorithm *MME*, for the problem will be given in Section 35.3.1. The algorithm runs in $O(n^3 \log^2 n)$ time for multiprocessor environment and in $O(n^2)$ time for a single processor environment. Then, we will analyze Algorithm *MME*, including the time complexity and the correctness, based on three properties of a specific type of tasks shown in Section 35.3.2. The correctness of these properties will be proved in the last subsection.

35.3.1 Algorithm for Minimizing Maximum Weighted Error

Before we show Algorithm *MME*, we need to define some terminology. For a given task system TS, a task T_r is said to be *removable* if $\varepsilon(TS) = \varepsilon(TS - \{T_r\})$; otherwise, *irremovable*. We will state two properties