2012 24th Euromicro Conference on Real-Time Systems

Partitioned Packing and Scheduling for Sporadic Real-Time Tasks in Identical Multiprocessor Systems

Jian-Jia Chen[1] and Samarjit Chakraborty[2]
[1]Department of Informatics, Karlsruhe Institute of Technology (KIT), Germany
[2]Institute for Real-Time Computer Systems, Technical University of Munich (TUM), Germany
E-mail: jian-jia.chen@kit.edu, samarjit@tum.de

Abstract

Multiprocessor platforms have been widely adopted to accommodate the increasing computation requirement of modern applications. Partitioned scheduling (or packing) has been widely exploited by partitioning real-time tasks onto processors to meet the timing constraints, which has been shown to be $\mathcal{NP}$-complete in the strong sense. This paper studies the approximation of partitioned scheduling by exploiting resource augmentation with (1) speeding up or (2) allocating more processors. When adopting speeding up to meet timing constraints, we provide a polynomial-time approximation scheme (PTAS) to derive near-optimal solutions only with the assumption that the ratio of the maximum relative deadline to the minimum relative deadline is a constant. The previously known PTAS for this problem imposes additional restrictions on the periods and the execution times of tasks. By removing these additional constraints, our scheme can be adopted for wider task sets. When considering the resource augmentation by allocating more processors, we show that there does not exist any asymptotic polynomial-time approximation scheme (APTAS) unless $\mathcal{P} = \mathcal{NP}$.

Keywords: Sporadic real-time tasks, resource augmentation, approximation, schedulability analysis.

1 Introduction

The sporadic task model is the most adopted model for infinitely recurring executions in the real-time systems domain [18]. Namely, a sporadic real-time task τ_i is characterized by its (worst-case) execution time C_i, its relative deadline D_i, and its minimum inter-arrival time T_i. A sporadic real-time task has an infinite sequence of task instances, in which a task instance is also usually called a *job*. The minimum inter-arrival time T_i for task τ_i specifies the minimum temporal separation for any two consecutive jobs of task τ_i. For historical reasons, according to the Liu and Layland model [17], the minimum inter-arrival time is also referred to as the period of the task. When a job of task τ_i arrives at time t, the job should finish no later than its *absolute deadline* $t + D_i$.

By considering the relations between the relative deadlines and the minimum inter-arrival times of tasks, we can classify the input task set with (1) *implicit deadlines*, in which the relative deadline of a task is equal to its minimum inter-arrival time for each sporadic task, (2) *constrained deadlines*, in which the minimum inter-arrival time is no less than the relative deadline for every sporadic task, and (3) *arbitrary deadlines*, otherwise. For uniprocessor systems, the earliest-deadline-first (EDF) policy has been shown to be optimal [17] to feasibly schedule the sporadic tasks in a preemptive setting. In EDF policy, the system gives the highest priority for execution to the job with the earliest absolute deadline. For the simplest case for tasks with implicit deadlines, the schedulability test of EDF involves testing whether the total utilization $\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i}$ is at most 100% [17], in which the utilization of task τ_i is defined as $\frac{C_i}{T_i}$. However, verifying the schedulability of EDF for task sets with constrained or arbitrary deadlines is not an easy problem, and has been shown to be co$\mathcal{N}P$-hard [12].

Moreover, as embedded systems have started towards multicore and multiprocessor platforms, the computing system has to effectively and efficiently schedule and execute programs/tasks on such platforms. To schedule tasks on multiprocessor platforms, there are three alternative paradigms adopted in the literature: *partitioned*, *global*, and *semi-partitioned* scheduling. The partitioned scheduling approach statically partitions tasks onto processors, in which a task is statically assigned on a specific processor. That is, all the jobs of a task are executed on a specific processor. The global scheduling approach decides the scheduling of the jobs by allowing a job to be migrated from one processor to another. The semi-partitioned scheduling approach assigns most tasks statically onto processors by only dividing a few tasks into sequential subtasks to be executed on different processors. The partitioned scheduling approach is more restricted, but requires no run-time management. The global scheduling approach requires more run-time overhead, while the semi-partitioned scheduling approach has moderate run-time overhead [6]. Davis and Burns [10] have provided a comprehensive survey of multiprocessor scheduling in real-time systems.

This paper focuses on partitioned scheduling on identical multiprocessor platforms. The decision version of the multiprocessor partitioned scheduling problem has been shown to be $\mathcal{NP}$-complete [18]. Due to the $\mathcal{NP}$-completeness of the studied problem, to provide efficient algorithms, some approximations are needed, unless $\mathcal{P} = \mathcal{NP}$. This paper studies the approximation of partitioned scheduling by exploiting *resource augmentation* with (1) speeding up or (2) allocating more processors. Therefore, two specific problems are explored in this paper: (1) *partitioned scheduling*: how to partition sporadic real-time tasks in a multiprocessor system with M identical processors to meet the timing constraints of tasks, and (2) *partitioned packing*: how to minimize the required number of processors and partition sporadic real-time tasks to meet their timing constraints. We explore the maximum required resource augmentation, in which the *resource augmentation factor* is defined as the worst-case speed-up factor for the multiprocessor partitioned scheduling problem and the worst-case factor of additionally allocated processors for the multiprocessor partitioned packing problem.

For task sets with implicit deadlines, in which the schedulability can be easily verified by testing the total utilization, the partitioned scheduling problem is equivalent to the Makespan problem [15], where the objective is to minimize the maximum utilization among the processors after task partitioning. Similarly, the partitioned packing problem is equivalent to the bin packing problem [14], for task sets with implicit deadlines. Both problems have been widely studied in the literature, in which the Makespan problem admits polynomial-time approximation schemes, due to Hochbaum and Shmoys [16], and the bin packing problem admits asymptotic polynomial-time approximation schemes, due to de la Vega and Lueker [11].

For partitioned scheduling on identical multiprocessor systems for arbitrary- or constrained-deadlined task sets, the first non-trivial algorithm for partitioned scheduling was proposed by Baruah and Fisher in [3]. They show that deadline-monotonic partitioning based on the approximate demand bound functions for task partitioning has a constant resource augmentation factor for speeding up [3, 4]. Chen and Chakraborty [9] provide a tighter analysis for the deadline-monotonic partitioning algorithm, in which the resource augmentation factor for speeding up is $2.6322 - \frac{1}{M}$ for constrained-deadline task sets and is $3-\frac{1}{M}$ for arbitrary-deadline task sets. Moreover, the difficulty with partitioned scheduling of sporadic real-time tasks has also been addressed by Fisher in [13].

Our Contributions Closely related to this paper, Baruah [2] proposes a polynomial-time approximation scheme (PTAS) for the partitioned scheduling problem, in which the resource augmentation factor is $(1 + \epsilon)$ and the time complexity is polynomial time by assuming $\frac{1}{\epsilon}$ is a constant, for $\epsilon > 0$. However, the approach in [2] only works if each of the three parameters $\frac{C_{\max}}{C_{\min}}$, $\frac{D_{\max}}{D_{\min}}$, $\frac{T_{\max}}{T_{\min}}$ is bounded by a constant, in which $C_{\max}$ ($C_{\min}$, resp.) is the maximum (minimum, resp.) worst-case execution time, $D_{\max}$ ($D_{\min}$, resp.) is the maximum (minimum, resp.) relative deadline, and $T_{\max}$ ($T_{\min}$, resp.) is the maximum (minimum, resp.) minimum inter-arrival time. Therefore, for a task with negligible workload, e.g., very small C_i or very large T_i, the resulting $C_{\min}$ is very small or $T_{\max}$ is very large, and, hence, the approach in [2] cannot lead to polynomial-time approximation schemes any more.

This paper explores the multiprocessor partitioned scheduling problem by adopting the results in [8] for vector scheduling. By a proper reduction from the partitioned scheduling problem to the vector scheduling problem, we provide a scheme to allow a $(1 + \epsilon)$ resource augmentation factor for speeding up, for $\epsilon > 0$. The scheme has polynomial-time complexity by assuming $\frac{1}{\epsilon}$ is a constant when (1) $\frac{D_{\max}}{D_{\min}}$ is a constant or (2) there is a constant number of clusters with bounded relative deadlines. Here, a cluster is defined such that the maximum relative deadline of the tasks divided by the minimum relative deadline of the tasks in the cluster is a constant. This removes the required constraints on the execution time and minimum inter-arrival time of tasks in the approach in [2]. Therefore, it can be applicable for more general task sets. Moreover, when considering the resource augmentation by allocating more processors, we show that there does not exist any asymptotic polynomial-time approximation scheme (APTAS), unless $\mathcal{P} = \mathcal{NP}$.

The rest of this paper is organized as follows: Section 2 specifies the task and platform model. Section 3 presents the results for the multiprocessor partitioned scheduling problem. The hardness for the multiprocessor partitioned packing problem is presented in Section 4. Section 5 concludes the paper with an outline of possible future research.

2 System Models and Preliminary Results

This section provides the task and platform model used in this paper, defines the studied problems, and gives the definition of resource augmentation.

2.1 Task and Platform Model

We consider a set $\mathbf{T} = \{\tau_1, \tau_2, \ldots, \tau_N\}$ of N independent sporadic real-time tasks. A task τ_i is characterized by its relative deadline D_i, its minimum inter-arrival time T_i, and its (worst-case) execution time C_i. For notational simplicity, the *utilization* of task τ_i is denoted by $u_i = \frac{C_i}{T_i}$. The execution platform has several identical processors, and a task has the same execution and timing property on all of these processors.

Note that, for the rest of this paper, we index the tasks from the shortest relative deadline to the longest, i.e., $D_i \leq D_j$ if $i < j$. Without loss of generality, we only consider the cases with $u_i \leq 100\%$ and $\frac{C_i}{D_i} \leq 100\%$ for any task τ_i. Otherwise, by definition, the task cannot meet the timing constraint. For notational brevity, let $\mathcal{N}$ be the set of all non-negative integer numbers.

2.2 Problem Definition

Given task set $\mathbf{T}$, a *feasible task partition* on M identical processors is to partition task set $\mathbf{T}$ to M subsets, says, $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$, such that

- $\mathbf{T}_i \cap \mathbf{T}_j$ is an empty set,
- $\cup_{i=1}^{M} \mathbf{T}_i$ is equal to the input task set $\mathbf{T}$, and
- $\mathbf{T}_i$ can be feasibly scheduled on processor i by applying the EDF scheduling algorithm.

The multiprocessor partitioned scheduling problem: The objective is to find a feasible task partition on the given M identical processors.

The multiprocessor partitioned packing problem: The objective is to minimize M such that there exists a feasible task partition on M identical processors.

2.3 Demand Bound Function

To provide an exact schedulability test, Baruah et al. [5] show that the most critical instance is to release the first job of tasks synchronously (says, at time 0), and the subsequent job arrivals should be as rapidly as legally possible. Based on that observation, the *demand bound function* $dbf(\tau_i, t)$, as in [5], of a task τ_i within any time interval with length equal to t (or from time 0 to time t, under the above synchronous release) can be defined as the maximum demand of task τ_i to be released and finished in the interval. That is,

$$dbf(\tau_i, t) = \max\left\{0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right\} \times C_i. \qquad (1)$$

However, as the demand bound function is not a continuous function, the schedulability test by verifying the summation of the demand bound functions of all the tasks for all $t \geq 0$ requires pseudo-polynomial time in the worst case.

In fact, for a given $n \in \mathcal{N}$, as $dbf(\tau_i, t)$ remains the same for any $D_i + (n-1)T_i < t < D_i + nT_i$, only some time points are important. The following lemma specifies the time points to test without sacrificing the correctness for schedulability.

Lemma 1 *Let $t_{i,n}$ be the time instant t such that $\frac{t_{i,n} - D_i}{T_i}$ is exactly equal to n. EDF schedule is feasible for a set $\mathbf{T}_m$ of tasks on a processor if and only if*

$$\forall \tau_i \in \mathbf{T}_m, n \in \mathcal{N}, \sum_{\tau_j \in \mathbf{T}_m} dbf(\tau_j, t_{i,n}) \leq t_{i,n}.$$

Proof. This lemma comes directly from the demand bound function analysis in [5]. □

For notational brevity, the maximum density $\gamma_{\max}$ of a task partition $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$ is defined as

$$\gamma_{\max} \stackrel{\text{def}}{=} \max_{m=1,2,\ldots,M, t>0} \frac{\sum_{\tau_i \in \mathbf{T}_m} dbf(\tau_i, t)}{t}. \qquad (2)$$

Therefore, a feasible task partition requires $\gamma_{\max} \leq 1$.

2.4 Resource Augmentation and Approximation

The multiprocessor partitioned scheduling problem and the multiprocessor partitioned packing problem are both $\mathcal{NP}$-hard problem in the strong sense. Therefore, unless $\mathcal{P} = \mathcal{NP}$, there does not exist any polynomial-time algorithm to decide whether there exists a feasible solution for the multiprocessor partitioned scheduling problem. Moreover, unless $\mathcal{P} = \mathcal{NP}$, there does not exist any polynomial-time algorithm to derive optimal solutions for the multiprocessor partitioned packing problem.

Approximations are, hence, needed to allow efficient algorithms. For the multiprocessor partitioned packing problem, the approximation is to sacrifice the optimality. For example, we can use more processors. If an algorithm $\mathcal{A}$ for the multiprocessor partitioned packing problem has a ρ approximation factor, it guarantees that *the task partition derived from the algorithm $\mathcal{A}$ is always feasible on each processor by using at most ρM^* processors, where M^* is the minimum (optimal) number of processors to schedule the task set* $\mathbf{T}$. As the approximation is also a resource augmentation, we also say such an algorithm $\mathcal{A}$ has a ρ resource augmentation factor for the multiprocessor partitioned packing problem, where $\rho > 1$.

For the multiprocessor partitioned scheduling problem, the approximation is based on resource augmentation on the speeds of the M processors [3, 9]. If an algorithm $\mathcal{A}$ for the multiprocessor partitioned scheduling problem has a ρ resource augmentation factor, it guarantees that *the task* *partition derived from the algorithm $\mathcal{A}$ is always feasible on each processor by speeding up to ρ times as fast as in the original platform speed, if task set $\mathbf{T}$ is feasible on the original M identical processors.*

According to the above definition for the existence of a feasible task partition on M processors, we know that the maximum density $\gamma_{\max}$ for the feasible task partition is no more than 1. Suppose that $\gamma^*_{\max}$ is the minimum $\gamma_{\max}$ among the feasible task partitions. Therefore, the multiprocessor partitioned scheduling problem is equivalent to the minimization of $\gamma_{\max}$. *Therefore, for the rest of this paper, when considering the partitioned scheduling problem, we will implicitly target at the approximation schemes for the minimization of $\gamma_{\max}$.*

For $\mathcal{NP}$-hard problems in the strong sense, a polynomial-time approximation scheme (PTAS) is the best we can achieve for approximation in polynomial time [14], unless $\mathcal{P} = \mathcal{NP}$. That is, for a minimization problem, a PTAS is an algorithm with a user-input $\epsilon > 0$, in which its approximation factor is $1 + \epsilon$ and the complexity is polynomial in the input size by taking $\frac{1}{\epsilon}$ as a constant. According to the above definitions, we are targeting for this definition of PTAS with respect to resource augmentation.

This paper focuses on the case where the arrival times of the sporadic tasks are not specified. Therefore, the approximation is for the worst cases. For tasks with specified arrival times on a uniprocessor, Bonifaci et al. [7] have shown that there is no polynomial-time algorithm to test the feasibility of a task set within a constant resource augmentation factor for speeding up, unless $\mathcal{P} = \mathcal{NP}$.

3 Partitioned Scheduling Problem

This section presents the proposed polynomial-time approximation (resource augmentation) scheme for the partitioned scheduling problem when the maximum relative deadline divided by the minimum relative deadline, i.e., $\frac{D_N}{D_1}$, is a constant or the task set has a constant number of clusters of tasks with bounded relative deadlines.

The algorithm proposed here is more complicated than the deadline-monotonic partitioning algorithm in [3]. Therefore, from an implementation perspective, as also suggested in [2], it is recommended to use the algorithm in [3] first, and our proposed algorithm should be adopted when the algorithm in [3] fails to derive feasible task partitions.

Vector Scheduling *The vector scheduling problem:* Given an integer M and a set $\mathbf{V}$ of vectors $< v_1, v_2, \ldots, v_N >$ with d dimensions, in which $q_{i,j}$ is the value for vector v_i in the j-th dimension, the problem is to partition $\mathbf{V}$ into M subsets $\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_M$ such that

$$Q^* = \max_{1 \le j \le d, 1 \le m \le M} \sum_{v_i \in \mathbf{V}_m} q_{i,j}$$

is minimized. Without loss of generality, the values $q_{i,j}$ are all non-negative numbers and less than or equal to 1.

The vector scheduling problem can be approximated with polynomial-time approximation schemes if d is a constant [8]. When the number of dimensions is not a constant, the vector scheduling problem can be approximated with a $O(\log d)$ approximation factor [8].

We will adopt the polynomial-time approximation scheme proposed by Chekuri and Khanna [8] for the vector scheduling problem with a constant dimension d.

Lemma 2 *[Chekuri and Khanna [8]] Given any $\epsilon > 0$, there is a $(1 + \epsilon)$-approximation algorithm for the vector scheduling problem that runs in $O((\frac{Nd}{\epsilon})^{O(s)})$, where $s = O((\frac{\ln(\frac{d}{\epsilon})}{\epsilon})^d)$.*

After introducing the vector scheduling problem and its algorithm, we will present our algorithm for multiprocessor partitioned scheduling in Section 3.1. Moreover, Section 3.2 will give some remarks about efficiency by reducing unnecessary dimensions.

3.1 Algorithm for Partitioned Scheduling

Our proposed algorithm for the multiprocessor partitioned scheduling problem is based on the polynomial-time approximation scheme in Lemma 2. The basic idea is to transform the input instance of the multiprocessor partitioned scheduling problem to an input instance of the vector scheduling problem so that the error is tolerable. Even though the general form for the schedulability test requires us to test the demand bound functions in an infinite number of points in time, as in Lemma 1, we may be able to reduce the number of tests without incurring significant errors.

Therefore, if we can reduce the required number of tests of the demand bound functions to a constant, the multiprocessor partitioned scheduling problem for such input instances can allow polynomial-time approximation schemes.

Lemma 3 *For a given task set $\mathbf{T}$ of the multiprocessor partitioned scheduling problem and constant time points $t_1, t_2, \ldots, t_d$ (where d is a constant), suppose that any subset $\mathbf{T}'$ of $\mathbf{T}$ is schedulable by EDF if and only if*

$$\forall 1 \le j \le d, \qquad \sum_{\tau_i \in \mathbf{T}'} dbf(\tau_i, t_j) \le t_j.$$

There is a polynomial-time approximation scheme for such a task set $\mathbf{T}$ for the multiprocessor partitioned scheduling problem.

Proof. According to the assumption, for such task sets, we can construct a vector v_i for task τ_i in which $q_{i,j} = \frac{dbf(\tau_i, t_j)}{t_j}$ for $j = 1, 2, \ldots, d$. Then, we can adopt the polynomial-time approximation scheme in Lemma 2 to decide the scheduling of the constructed vectors on M processors. For such cases, if v_i is assigned to $\mathbf{V}m$, we can assign task τ_i on processor m. The time complexity is dominated by the algorithm in Lemma 2 and the resource augmentation factor is $1 + \epsilon$. □

The condition in Lemma 3 is not easy to satisfy by selecting only a constant number of time points for exact schedulability tests. One specific example is to have only one common relative deadline for all the tasks.[1]

Therefore, we will move to more general cases by introducing some tolerable errors when selecting the number of time points for the demand bound function testing. Instead of finding a constant number of time points for exact schedulability tests, we find a constant number of time points with tolerable resource augmentation factors.

Suppose that each task τ_i in $\mathbf{T}$ has a corresponding vector v_i. We define a *d-dimensional representative vector set* $\mathbf{V}$ for the given task set $\mathbf{T}$ under a user-defined tolerable error $0 < \eta < 1$ if for any subset $\mathbf{T}'$ of $\mathbf{T}$ and the corresponding vector set $\mathbf{V}'$ of $\mathbf{V}$

$$\gamma(\mathbf{T}') \geq ||\mathbf{V}'||_\infty \geq (\frac{1}{1+\eta})\gamma(\mathbf{T}'), \tag{3}$$

where $\gamma(\mathbf{T}')$ is defined as $\max_{t>0} \frac{\sum_{\tau_i \in \mathbf{T}'} dbf(\tau_i, t)}{t}$. That is, the d-dimensional representative vector set for $\mathbf{T}$ guarantees a bounded error.

Theorem 1 *For the multiprocessor partitioned scheduling problem, if there exists a d-dimensional representative vector set $\mathbf{V}$ for the given task set $\mathbf{T}$ under a user-defined tolerable error $\eta > 0$, in which d is a constant, then there is a polynomial-time approximation scheme.*

Proof. By applying the algorithm in Lemma 2 for the vector scheduling problem for vector set $\mathbf{V}$ on M processors, let $\mathbf{V}_1^\dagger, \mathbf{V}_2^\dagger, \ldots, \mathbf{V}_M^\dagger$ be the resulting vector partition. We can simply return the solution $\mathbf{T}_1^\dagger, \mathbf{T}_2^\dagger, \ldots, \mathbf{T}_M^\dagger$ by assigning τ_i to $\mathbf{T}_m$ if v_i is in $\mathbf{V}_m^\dagger$.

Suppose that $\mathbf{T}_1^*, \mathbf{T}_2^*, \ldots, \mathbf{T}_M^*$ is the optimal task partition for the multiprocessor partitioned scheduling problem. By Lemma 2 and the assumption with the tolerable error in (3),

$$\max_{m=1,2,\ldots,M} \gamma(\mathbf{T}_m^\dagger) \leq (1+\eta) \max_{m=1,2,\ldots,M} ||\mathbf{V}_m^\dagger||_\infty \tag{4}$$

$$\leq_1 (1+\eta)(1+\epsilon) \max_{m=1,2,\ldots,M} ||\mathbf{V}_m^*||_\infty \tag{5}$$

$$\leq_2 (1+\eta)(1+\epsilon) \max_{m=1,2,\ldots,M} \gamma(\mathbf{T}_m^*) \tag{6}$$

where $\leq$ and $\leq_2$ come from the inequality assumed in (3) for the representative vector set, and $\leq_1$ is from Lemma 2.

Therefore, the derived solution has an approximation factor guarantee $(1+\epsilon)(1+\eta)$ for the multiprocessor partitioned scheduling problem. If we take ϵ equal to η, we know that the approximation factor guarantee is equal to $(1+\epsilon)^2$, which is upper-bounded by $1 + 3\epsilon$ when $0 < \epsilon < 1$.

The complexity for creating the corresponding vector set $\mathbf{V}$ is $O(dN)$. The complexity for recovering the task partition from the derived vector scheduling from Lemma 2 is $O(N)$. Therefore, it is clear that the procedure is dominated by the process of deriving $\mathbf{V}_1^\dagger, \mathbf{V}_2^\dagger, \ldots, \mathbf{V}_M^\dagger$ from Lemma 2. By the assumption that d is a constant, the complexity is polynomial in time by taking $\frac{1}{\eta}$ as a constant. Therefore, this proves the theorem. □

Therefore, the essential problem of our approach is to find a *d-dimensional representative vector set* for the given input task set, where *d must be a constant*. We will present the types of task sets that have polynomial-time approximation schemes for the multiprocessor partitioned scheduling problem. Two types will be presented: (1) task sets in which the maximum relative deadline divided by the minimum relative deadline, i.e., $\frac{D_N}{D_1}$, is a constant; (2) task sets with σ clusters of the relative deadlines, in which σ is a constant and the maximum relative deadline of the tasks divided by the minimum relative deadline of the tasks in a cluster is a constant.

3.1.1 $\frac{D_N}{D_1}$ is a constant

Suppose λ is defined as the constant $\frac{D_N}{D_1}$ in $\mathbf{T}$, i.e., $\lambda =^{def} \frac{D_N}{D_1}$. Let h be $\left\lceil \frac{\ln(1+\eta)}{\ln \frac{\lambda}{\eta}} \right\rceil$. We know that $D_1(1+\eta)^h \geq \frac{D_N}{\eta}$. Now, we can discretize the time interval $[D_1, D_1(1+\eta)^h]$, which covers $[D_1, \frac{D_N}{\eta}]$ as follows:

- let z_0 be D_1;
- z_i is $z_{i-1} \cdot (1+\eta)$.

As a result, we divide the time interval $[D_1, \frac{D_N}{\eta}]$ into h sub intervals: $[D_1, (1+\eta)D_1)$, $[(1+\eta)D_1, (1+\eta)^2 D_1)$, … $[(1+\eta)^{h-1}D_1, (1+\eta)^h D_1)$.

Then, we can create a vector v_i with $h+3$ dimensions for task τ_i by setting

- $q_{i,j} = \frac{dbf(\tau_i, z_{j-1})}{z_{j-1}}$ for $j = 1, 2, \ldots, h+1$,
- $q_{i,h+2} = \frac{dbf(\tau_i, D_N)}{D_N}$, and
- $q_{i,h+3} = \frac{dbf(\tau_i, \infty)}{\infty} = u_i$.

Figure 1 provides an example for the above process. The constructed vector set is of course an approximation as it only considers $h+3$ possible values for the demand bound

[1]This is proved in Lemma 9 in Section 4, in which only two time points $\{D_1, \infty\}$ are considered.

functions. However, the following lemma shows that the error we make by taking only $h+3$ points is also bounded.

Lemma 4 *The vector set $\mathbf{V}$ is an $(h+3)$-dimensional representative vector set for task set $\mathbf{T}$, i.e., for any subset $\mathbf{T}'$ of $\mathbf{T}$ and the corresponding vector set $\mathbf{V}'$, the inequality in (3) holds.*

Proof. The inequality $\gamma(\mathbf{T}') \geq ||\mathbf{V}'||_\infty$ holds as

$$\begin{aligned}\gamma(\mathbf{T}') =& \max_{t>0} \sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, t)}{t} \\ \geq& \max_{t \in \{z_0, z_1, \ldots, z_h\} \cup \{\infty, D_N\}} \sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, t)}{t} \\ =& ||\mathbf{V}'||_\infty,\end{aligned}$$

where the inequality comes from the fact that $\{z_0, z_1, \ldots, z_h\} \cup \{\infty, D_N\}$ is only a subset for the set of all possible positive real numbers and the second equality comes from the definition of $||\mathbf{V}'||_\infty$.

For the rest of the proof, we only focus on the proof of $||\mathbf{V}'||_\infty \geq \frac{1}{1+\eta}\gamma(\mathbf{T}')$. By definition, the demand bound function is 0 for all the tasks when t is less than D_1. We only have to consider t in the time interval $[z_0, \infty]$. Suppose that t^* is the time $t > 0$, in which $\sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, t)}{t}$ is maximized. There are two cases: (1) t^* is in time interval $[z_{j-1}, z_j)$ for some $1 \leq j \leq h$ or (2) $t^* \geq z_h$.

Case 1 $t^* \in [z_{j-1}, z_j)$: Therefore,

$$\begin{aligned}\sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, t^*)}{t^*} \leq \sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, z_j)}{z_{j-1}} = \sum_{\tau_i \in \mathbf{T}'} \frac{dbf(\tau_i, z_j)}{\frac{1}{1+\eta} z_j} \\ = (1+\eta) \sum_{v_i \in \mathbf{V}'} q_{i,j+1} \leq (1+\eta)||\mathbf{V}'||_\infty,\end{aligned}$$

where the first inequality comes from the fact that $t^* \geq z_{j-1}$ and the demand bound function of a task τ_i is non-decreasing, i.e., $dbf(\tau_i, t^*) \leq dbf(\tau_i, z_j)$.

Case 2 $t^* \geq z_h \geq \frac{D_N}{\eta}$: We have

$$\begin{aligned}&\frac{\sum_{\tau_i \in \mathbf{T}'} dbf(\tau_i, t^*)}{t^*} \leq \frac{\sum_{\tau_i \in \mathbf{T}'} (C_i + \frac{(t^* - D_i)}{T_i} C_i)}{t^*} \\ &\leq \frac{\sum_{\tau_i \in \mathbf{T}'} (C_i + \frac{t^*}{T_i} C_i)}{t^*} = \frac{\sum_{\tau_i \in \mathbf{T}'} (C_i + u_i t^*)}{t^*} \\ &= \sum_{\tau_i \in \mathbf{T}'} (\frac{C_i}{t^*} + u_i) \leq \sum_{\tau_i \in \mathbf{T}'} (\frac{\eta C_i}{D_N} + u_i) \\ &\leq_1 \sum_{\tau_i \in \mathbf{T}'} \eta \frac{dbf(\tau_i, D_N)}{D_N} + \frac{dbf(\tau_i, \infty)}{\infty} \\ &\leq \sum_{v_i \in \mathbf{V}'} \eta q_{i,h+2} + q_{i,h+3} \leq (1+\eta)||\mathbf{V}'||_\infty,\end{aligned}$$

where $\leq_1$ is because $dbf(\tau_i, D_N) \geq C_i$ since $D_i \leq D_N$ for any task $\tau_i \in \mathbf{T}'$.

Therefore, we reach the inequality described by (3). □

Now, we can conclude the existence of polynomial-time approximation schemes for the multiprocessor partitioned scheduling problem when the ratio of the maximum relative deadline to the minimum relative deadline is a constant.

Theorem 2 *If $\frac{D_N}{D_1}$ is a constant, there is a polynomial-time approximation scheme for such a task set $\mathbf{T}$ for the multiprocessor partitioned scheduling problem.*

Proof. By Lemma 4, we can construct an $(h+3)$-dimensional representative vector set for $\mathbf{T}$ under a user-defined tolerable error $0 < \eta < 1$ in polynomial time, as $h = \left\lceil \frac{\ln(1+\eta)}{\ln \frac{\lambda}{\eta}} \right\rceil$ is a constant, provided that $\frac{D_N}{D_1} = \lambda$ and $\frac{1}{\eta}$ are both constants. Therefore, by Lemma 4 and Theorem 1, we reach the conclusion. □

3.1.2 σ clusters of the relative deadlines

Suppose that the N tasks in $\mathbf{T}$ are clustered into σ groups. Let w_i be the largest index of the task that has the maximal relative deadline in the i-th cluster. Let w_0 be 0 and $D_{N+1} = \infty$ for notational brevity. Therefore, by definition, w_σ is N. We also define $\lambda_1 = \frac{D_{w_1}}{D_1}$, $\lambda_2 = \frac{D_{w_2}}{D_{w_1+1}}$, $\lambda_3 = \frac{D_{w_3}}{D_{w_2+1}}$, $\ldots$, $\lambda_\sigma = \frac{D_{w_\sigma}}{D_{w_{\sigma-1}+1}}$ and assume these numbers are upper-bounded by a constant λ, i.e., $\lambda_k \leq \lambda$ for $1 \leq k \leq \sigma$. Moreover, for the simplicity of presentation, for the k-the cluster, we further assume that $\frac{D_{w_k+1}}{D_{w_{k-1}+1}}$ is larger than $\frac{\lambda}{\eta}$, which is a constant. Otherwise, we can reorganize the clusters without increasing the number of clusters.[2]

Let h be defined as $\left\lceil \frac{\ln(1+\eta)}{\ln \frac{\lambda}{\eta}} \right\rceil$, as in Section 3.1.1. For the k-th cluster, similar to Section 3.1.1, we can discretize the time interval $[D_{w_{k-1}+1}, \frac{\lambda D_{w_{k-1}+1}}{\eta}]$ to into h sub-intervals: $[D_{w_{k-1}+1}, (1+\eta)D_{w_{k-1}+1})$, $[(1+\eta)D_{w_{k-1}+1}, (1+\eta)^2 w_{k-1}+1)$, $\ldots$ $[(1+\eta)^{h-1} w_{k-1}+1, (1+\eta)^h w_{k-1}+1]$, denoted by $[z_{k,0}, z_{k,1})$, $[z_{k,1}, z_{k,2})$, $\ldots$, $[z_{k,h-1}, z_{k,h})$. Then, we create a vector v_i with $\sigma(h+2)$ +1 dimensions for task τ_i by setting

- $q_{i,(k-1)\cdot(h+1)+j} = \frac{dbf(\tau_i, z_{k,j-1})}{z_{k,j-1}}$ for $j = 1, 2, \ldots, h+1$, $k = 1, 2, \ldots, \sigma$,
- $q_{i,\sigma\cdot(h+1)+k} = \frac{dbf(\tau_i, D_{w_k})}{D_{w_k}}$, for $k = 1, 2, \ldots, \sigma$, and
- $q_{i,\sigma\cdot(h+2)+1} = \frac{dbf(\tau_i, \infty)}{\infty} = u_i$.

Therefore, we have created a corresponding vector set $\mathbf{V}$ for task set $\mathbf{T}$.

[2]This is simply for the ease of presentation. The formation of clusters can be easily adjusted.

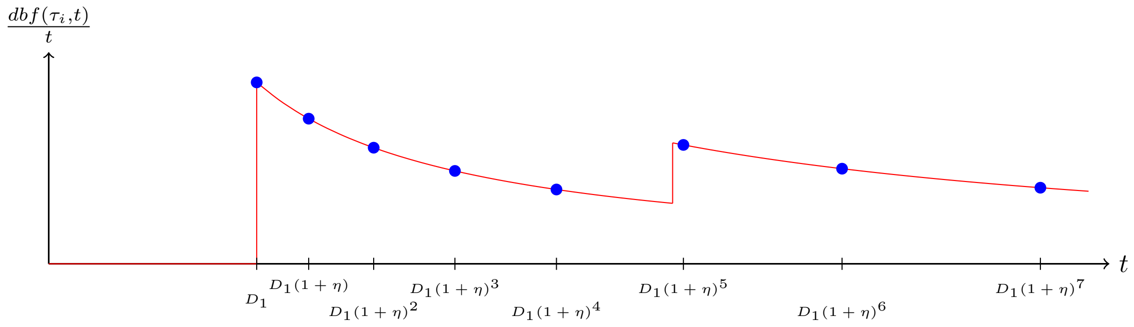

Figure 1: An example for the first $h+1$ dimensions, where $h=7$. The circles are used to construct the dimensions along with the demand bound functions when $t=\infty$ and $t=D_N$.

Lemma 5 *The vector set $\mathbf{V}$ is a $\sigma(h+2)+1$-dimensional representative vector set for task set $\mathbf{T}$, i.e., for any subset $\mathbf{T}'$ of $\mathbf{T}$ and the corresponding vector set $\mathbf{V}'$, the inequality in (3) holds.*

Proof. The proof is very similar to that for Lemma 4. The proof for $\gamma(\mathbf{T}') \geq ||\mathbf{V}'||_\infty$ is the same. There are now three cases for t^*: (1) t^* is in time interval $[z_{k,j-1}, z_{k,j})$ for some $j \leq h$, (2) $t^* \geq z_{k,h}$, or (3) t^* is in time interval $[z_{k,h}, z_{k+1,0})$ for $k=1,2,\ldots,\sigma-1$. The proof for the first two cases are the same as in the proof in Lemma 4.

We only focus on the last case, in which $z_{k,h} \leq t < z_{k+1,0}$. Suppose that $\mathbf{T}'^k$ is the subset of tasks in $\mathbf{T}'$, in which task τ_i in $\hat{\mathbf{T}}$ has relative deadline less than or equal to the maximal relative deadline of the tasks in the k-th cluster. Then, for $\frac{D_{w_k}}{\eta} \leq z_{k,h} \leq t^* < z_{k+1,0}$, we have

$$\frac{\sum_{\tau_i \in \mathbf{T}'} dbf(\tau_i, t^*)}{t^*} = \frac{\sum_{\tau_i \in \mathbf{T}'^k} dbf(\tau_i, t^*)}{t^*}$$
$$\leq \sum_{\tau_i \in \mathbf{T}'^k} (\frac{C_i}{t} + u_i) \leq \sum_{\tau_i \in \mathbf{T}'^k} (\frac{\eta C_i}{D_{w_k}} + u_i)$$
$$\leq \sum_{v_i \in \mathbf{V}'} \eta q_{i,\sigma\cdot(h+1)+k} + q_{i,\sigma\cdot(h+2)+1} \leq (1+\eta)||\mathbf{V}'||_\infty.$$

□

Theorem 3 *For a given task set, if there are σ clusters of tasks, in which σ is a constant and the maximum relative deadline of the tasks divided by the minimum relative deadline of the tasks in a cluster is upper-bounded by a constant, then there is a polynomial-time approximation scheme for the multiprocessor partitioned scheduling problem.*

Proof. This comes directly from Theorem 1 and Lemma 5. □

3.2 Efficiency and Reducing Unnecessary Dimensions

Even though we have errors both by applying the PTAS proposed in [8] and by taking only the constant dimensional representative vector set, we have shown in Section 3.1 that the errors can be both considered to achieve our PTAS for some settings. For example, by taking η as ϵ, we can achieve a $1+3\epsilon$ resource augmentation factor. However, some dimensions we considered in Section 3.1 may be unnecessary. As the time complexity strongly depends on the number of dimensions in the representative vector set, reducing the unnecessary dimensions will significantly reduce the complexity. For the rest of the discussions, suppose that the demand bound functions are considered at time $z_0, z_1, z_2, \ldots z_{\hat{h}}$, in which $z_0 < z_1 < z_2 < \cdots < z_{\hat{h}} = \infty$.

The following lemmas provide some conditions to remove unnecessary dimensions when constructing the representative vector set $\mathbf{V}$.

Lemma 6 *If, for any task τ_i, there does not exist any non-negative integer n with $z_{j-1} < D_i + nT_i \leq z_j$ for some $\hat{h} > j \geq 1$, the dimension constructed by considering z_j can be removed without sacrificing the guaranteed resource augmentation factor.*

Proof. This comes from Lemma 1. We do not have to check time t in time interval $(z_{j-1}, z_j]$ for schedulability tests for such a case. For every task τ_i with $z_{j-1} < t \leq z_j$, we know that $\frac{dbf(\tau_i,t)}{t} = \frac{dbf(\tau_i,z_{j-1})}{t} \leq \frac{dbf(\tau_i,z_{j-1})}{z_{j-1}}$. □

Lemma 7 *For any task τ_i with $D_i \leq z_j$, if $\frac{C_i}{z_{j-1}} < \eta \cdot u_i$, the dimension constructed by considering z_j can be removed without sacrificing the guaranteed resource augmentation factor.*

Proof. Due to the assumption $\frac{C_i}{z_{j-1}} \leq \eta \cdot u_i$ for any $D_i \leq z_j$, we know that for any $z_{j-1} < t \leq z_j$,

$$\frac{dbf(\tau_i, t)}{t} \leq \frac{C_i + (t-D_i)u_i}{t} \leq (1+\eta)u_i.$$

Therefore, the demand bound function of the any subset of tasks at time t with $z_{j-1} < t \leq z_j$ is bounded by $t \cdot (1+\eta) u_i$. We reach the conclusion of the lemma. □

Lemma 8 *For a given j, if there exists another dimension k with $q_{i,j} \leq q_{i,k}$ for every task τ_i in $\mathbf{T}$ or $q_{i,j} \leq (1+\eta)u_i$ for every task τ_i in $\mathbf{T}$, the dimension constructed by considering z_j can be removed without sacrificing the guaranteed resource augmentation factor.*

Proof. This comes from the fact that the removal does not affect the optimality at all and a similar argument in the proof in Lemma 7. □

As the examinations for the conditions in Lemmas 6, 7, and 8 only take polynomial time, removing unnecessary dimensions should be performed so that the time complexity can be reduced. This procedure will not affect the worst-case time complexity and resource augmentation analysis.

4 Partitioned Packing Problem

This section presents the non-existence of asymptotic polynomial-time approximation schemes for the multiprocessor partitioned packing problem for sporadic task sets with arbitrary deadlines, unless $\mathcal{P} = \mathcal{NP}$. The bin packing problem has no polynomial-time approximation algorithm with approximation factors less than $\frac{3}{2}$ unless $\mathcal{P} = \mathcal{NP}$. Such a special case happens when the optimal solution requires only two processors but any approximation uses three processors.

Theorem 4 *For any $\epsilon > 0$, there is no polynomial-time approximation algorithm with approximation factor of $\frac{3}{2} - \epsilon$ for the multiprocessor partitioned packing problem, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. This property comes directly from the hardness of the bin-packing problem [19, Chapter 8]. □

Therefore, asymptotic approximation algorithms can be adopted to deal with such a case. If an algorithm $\mathcal{A}$ for the multiprocessor partitioned packing problem has a ρ asymptotic approximation factor, it guarantees that the task partition derived from the algorithm $\mathcal{A}$ is always feasible on each processor by using at most $\rho M^* + \alpha$ processors, where M^* is the minimum (optimal) number of processors to schedule the task set $\mathbf{T}$ and α is a constant. For task sets with implicit deadlines, as the problem is equivalent to the bin packing problem, it has been shown in [11] for the existence of polynomial-time asymptotic approximation schemes by using at most $(1+2\epsilon)M^* + 1$ processors for a given $0 < \epsilon \leq \frac{1}{2}$.

However, when considering sporadic real-time tasks with arbitrary deadlines, we will prove the non-existence of asymptotic polynomial-time approximation schemes unless $\mathcal{P} = \mathcal{NP}$. The proof is based on a reduction from the *vector packing problem*, defined as follows:

The vector packing problem: Given a set $\mathbf{V}$ of vectors $< v_1, v_2, \ldots, v_N >$ with d dimensions, in which $q_{i,j}$ is the value for vector v_i in the j-th dimension, the problem is to partition $\mathbf{V}$ into M subsets such that M is minimized and each of the dimensions in the summation of the vectors in each subset is no more than 1.

The vector packing problem is more generalized than the bin packing problem, in which the bin packing problem is a special case of the vector packing problem with $d = 1$. Unfortunately, the vector packing problem does not admit any polynomial-time asymptotic approximation scheme unless $\mathcal{P} = \mathcal{NP}$ [20].

Before introducing the reduction, we will need the following lemma which gives the schedulability analysis for a special case in which all the tasks are with the same relative deadline.

Lemma 9 *Suppose that the tasks assigned on a processor m are with the same relative deadline, i.e., $D_i = D$ for any task $\tau_i \in \mathbf{T}_m$, EDF schedule is feasible for a set $\mathbf{T}_m$ of tasks on a processor if and only if*

$$\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} \leq 1 \textit{ and } \sum_{\tau_i \in \mathbf{T}_m} u_i \leq 1.$$

Proof. **Only if**: This direction is straightforward as the task set is not able to meet the timing constraint when $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} > 1$ or $\sum_{\tau_i \in \mathbf{T}_m} u_i > 1$.

If: If $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} \leq 1$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i \leq 1$, we know that when $t < D$ $\sum_{\tau_i \in \mathbf{T}_m} dbf(\tau_i, t) = 0$ and when $t \geq D$

$$\begin{aligned}
\sum_{\tau_i \in \mathbf{T}_m} dbf(\tau_i, t) &= \sum_{\tau_i \in \mathbf{T}_m} \left(\left\lfloor \frac{t-D}{T_i} \right\rfloor + 1 \right) \times C_i \\
&\leq \sum_{\tau_i \in \mathbf{T}_m} \left(\frac{t-D}{T_i} + 1 \right) \times C_i \\
&\leq \sum_{\tau_i \in \mathbf{T}_m} C_i + (t-D)u_i \\
&\leq D + (t-D) = t. \qquad (7)
\end{aligned}$$

Therefore, we know that if $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} \leq 1$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i \leq 1$, there task set $\mathbf{T}_m$ can meet the timing constraint by EDF scheduling. □

Now we can introduce the hardness of the multiprocessor partitioned packing problem.

Theorem 5 *Unless $\mathcal{P} = \mathcal{NP}$, there does not exist any asymptotic polynomial-time approximation scheme (APTAS) for the multiprocessor partitioned packing problem.*

Proof. The hardness is proved by a reduction from the two-dimensional vector packing problem to the multiprocessor partitioned packing problem: Given a set $\mathbf{V}$ of vectors $< v_1, v_2, \ldots, v_N >$ with 2 dimensions, let $q_{i,1}$ and $q_{i,2}$ be the values in the two dimensions for v_i. Even when $d = 2$, the vector packing problem does not admit any APTAS unless $\mathcal{P} = \mathcal{NP}$ [20].

For a vector v_i, we construct a task τ_i, in which its relative deadline D_i is set to 1, its execution time C_i is set to $q_{i,1}$, and its minimum inter-arrival time T_i is set to $\frac{q_{i,1}}{q_{i,2}}$. Therefore, the utilization of constructed task τ_i is $q_{i,2}$. Clearly, the above reduction is in polynomial time.

For a task subset $\mathbf{T}'$ of $\mathbf{T}$, suppose that $\mathbf{V}(\mathbf{T}')$ is the set of the corresponding vectors that are used to create the task subset $\mathbf{T}'$. Therefore, we know that $\sum_{\tau_i \in \mathbf{T}'} \frac{C_i}{D} = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}')} q_{i,1}$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}')} q_{i,2}$. By Lemma 9, the subset $\mathbf{T}_m$ of the constructed tasks can be feasibly scheduled by EDF on a processor if and only if $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}_m)} q_{i,1} \leq 1$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}_m)} q_{i,2} \leq 1$.

Therefore, if there exists an asymptotic polynomial-time approximation scheme for the multiprocessor partitioned packing problem, such an algorithm is also an asymptotic polynomial-time approximation scheme for the two-dimensional vector packing problem, which reaches the contradiction, unless $\mathcal{P} = \mathcal{NP}$. □

As shown above, minimizing the required number of processors to meet the timing constraints for sporadic real-time tasks is actually not easier than the resource augmentation by speeding up. The special case for tasks with the same relative deadline actually allows a PTAS when we augment the processor speeds, as we only need to consider two dimensions due to Lemma 3 and Lemma 9. However, when we have to minimize the number of processors, as shown in the proof in Theorem 5, such a special case even does not admit an APTAS.

Please note that the reduction in the proof of Theorem 5 only works when arbitrary-deadline tasks are considered. If all the tasks are with constrained deadlines, i.e., $D_i \leq T_i$ for any τ_i in $\mathbf{T}$, the reduction does not work, as the schedulability is only to verify whether $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{D} \leq 1$ in Lemma 9. The non-existence or existence of asymptotic polynomial-time approximation schemes is still open for tasks with constrained deadlines.

Moreover, it is also not difficult to see that a ρ-approximation algorithm for the multiprocessor partitioned packing problem is also useful for the multiprocessor partitioned problem as shown in the following theorem.

Theorem 6 *If there exists a polynomial-time algorithm $\mathcal{A}$ which guarantees a ρ-approximation factor for the multiprocessor partitioned packing problem, the multiprocessor partitioned scheduling problem also admits a polynomial-time algorithm with a $\lceil \rho \rceil$ resource augmentation factor, where $\rho \geq 1$.*

Proof. Suppose that the resulting solution from $\mathcal{A}$ uses $\hat{M}$ processors. If $\hat{M} \leq M$, where M is given for the multiprocessor partitioned scheduling problem, it is clear that the argument holds. For the rest of the proof, we only consider the case that $\hat{M} > M$. Let $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{\hat{M}}$ be the resulting solution by applying $\mathcal{A}$. As these tasks are feasibly scheduled on each processor, we know that

$$\forall t > 0 \sum_{\tau_i \in \mathbf{T}_m} dbf(\tau_i, t) \leq t.$$

Therefore, for any $k = 1, 2, \ldots, \hat{M} - \lceil \rho \rceil$, we have

$$\forall t > 0 \sum_{m=k}^{k+\lceil \rho \rceil} \sum_{\tau_i \in \mathbf{T}_m} dbf(\tau_i, t) \leq t \lceil \rho \rceil,$$

by merging $\lceil \rho \rceil$ consecutive task sets onto a processor. Therefore, if the system is speeded up by a factor $\lceil \rho \rceil$, we know that we can merge tasks on any $\lceil \rho \rceil$ processors and the resulting schedule is still feasible. As $\mathcal{A}$ is a ρ-approximation algorithm, we know that $M \geq \frac{\hat{M}}{\rho}$; otherwise, there is no feasible solution on M processors for the multiprocessor partitioned scheduling problem. Therefore, by $\rho \geq 1$, the resulting schedule after merging will use at most

$$\left\lceil \frac{\hat{M}}{\lceil \rho \rceil} \right\rceil \leq \left\lceil \frac{\rho M}{\lceil \rho \rceil} \right\rceil \leq M$$

processors, and each processor can feasibly schedule the tasks assigned on it by speeding up to run at $\lceil \rho \rceil$ of the original speed.

The above procedure for merging tasks can be easily done in polynomial time. Hence, the theorem is proved. □

5 Conclusion

This paper presents how to partition sporadic real-time tasks onto identical multiprocessors. We present a polynomial-time reduction to the vector scheduling problem and adopt the PTAS in [8]. The proposed scheme allows a $(1 + \epsilon)$ resource augmentation factor for speeding up for $\epsilon > 0$. The scheme has polynomial-time complexity by assuming $\frac{1}{\epsilon}$ is a constant when (1) $\frac{D_{\max}}{D_{\min}}$ is a constant or (2) there is a constant number of clusters with bounded relative deadlines. The results presented here for resource augmentation by speeding up improve the study by Baruah in [2], which assumes more constraints to obtain polynomial-time resource augmentation schemes. Therefore, our PTAS is applicable for wider task settings. Moreover, when considering the resource augmentation by allocating more processors, we show that there does not exist any asymptotic polynomial-time approximation scheme (APTAS), unless $\mathcal{P} = \mathcal{NP}$, and provide a connection to resource augmentation by speeding up.

Polynomial-time approximation schemes are mainly for studying the complexity for multiprocessor partitioned scheduling. We have presented some methods to reduce the number of dimensions. There are more arguments in [2] on when and how to apply these schemes. Please refer to [2] for more detailed discussions about efficiency.

The PTAS for multiprocessor partitioned scheduling can also be adopted to deal with more cases. When assigning a sporadic task τ_i onto a processor, suppose that the task also requires $\psi_{i,j}$ percentage from the local resource (e.g., memory) j. This problem has also been recently considered in [1]. We can consider each local resource on a processor as a dimension in the vector scheduling problem. Therefore, if the number of the local resources of a processor is a constant, the PTAS developed in this paper for multiprocessor partitioned scheduling can also be directly applied.

In addition, it is open whether there exists a polynomial-time resource augmentation scheme for the multiprocessor scheduling problem when there is no specific constraint on the relative deadlines of the sporadic real-time tasks. For tasks with constrained deadlines, the existence of asymptotic polynomial-time approximation schemes for the multiprocessor partitioned packing problem also remains open.

Acknowledgements

We are grateful to Sanjoy Baruah from University of North Carolina at Chapel Hill for the discussions on the complexity of partitioned multiprocessor scheduling. This work is sponsored in part by Baden Wuerttemberg MWK Juniorprofessoren-Programms.

References

[1] S. Baruah. Partitioning sporadic task systems upon memory-constrained multiprocessors. *to appear in ACM Transactions in Embedded Computing Systems*.

[2] S. Baruah. The partitioned edf scheduling of sporadic task systems. In *Real-Time Systems Symposium (RTSS)*, pages 116 –125, 2011.

[3] S. K. Baruah and N. Fisher. The partitioned multiprocessor scheduling of sporadic task systems. In *RTSS*, pages 321–329, 2005.

[4] S. K. Baruah and N. Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Trans. Computers*, 55(7):918–923, 2006.

[5] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *IEEE Real-Time Systems Symposium*, pages 182–190, 1990.

[6] A. Bastoni, B. B. Brandenburg, and J. H. Anderson. Is semi-partitioned scheduling practical? In *Proceedings of the 2011 23rd Euromicro Conference on Real-Time Systems*, ECRTS '11, pages 125–135, 2011.

[7] V. Bonifaci, H.-L. Chan, A. Marchetti-Spaccamela, and N. Megow. Algorithms and complexity for periodic real-time scheduling. In *SODA*, pages 1350–1359, 2010.

[8] C. Chekuri and S. Khanna. On multidimensional packing problems. *SIAM J. Comput.*, 33(4):837–851, 2004.

[9] J.-J. Chen and S. Chakraborty. Resource augmentation bounds for approximate demand bound functions. In *IEEE Real-Time Systems Symposium*, pages 272 – 281, 2011.

[10] R. I. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011.

[11] W. F. de la Vega and G. S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981.

[12] F. Eisenbrand and T. Rothvoß. Edf-schedulability of synchronous periodic task systems is coNP-hard. In *SODA*, pages 1029–1034, 2010.

[13] N. W. Fisher. How hard is partitioning for the sporadic task model? In *Proceedings of the 2009 International Conference on Parallel Processing Workshops*, ICPPW '09, pages 2–5, 2009.

[14] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W. H. Freeman and Co., 1979.

[15] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

[16] D. S. Hochbaum and D. B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987.

[17] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[18] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Cambridge, MA, USA, 1983.

[19] V. V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[20] G. J. Woeginger. There is no asymptotic ptas for two-dimensional vector packing. *Inf. Process. Lett.*, 64(6):293–297, 1997.