Packing Sporadic Real-Time Tasks on Identical Multiprocessor Systems

Jian-Jia Chen

Department of Computer Science, TU Dortmund University, Germany
jian-jia.chen@cs.uni-dortmund.de
https://orcid.org/0000-0001-8114-9760

Nikhil Bansal

Eindhoven University of Technology, The Netherlands
n.bansal@tue.nl

Samarjit Chakraborty

Technical University of Munich (TUM), Germany
samarjit@tum.de
https://orcid.org/0000-0002-0503-6235

Georg von der Brüggen

Department of Computer Science, TU Dortmund University, Germany
georg.von-der-brueggen@tu-dortmund.de
https://orcid.org/0000-0002-8137-3612

Abstract

In real-time systems, in addition to the functional correctness recurrent tasks must fulfill timing constraints to ensure the correct behavior of the system. Partitioned scheduling is widely used in real-time systems, i.e., the tasks are statically assigned onto processors while ensuring that all timing constraints are met. The decision version of the problem, which is to check whether the deadline constraints of tasks can be satisfied on a given number of identical processors, has been known $\mathcal{NP}$-complete in the strong sense. Several studies on this problem are based on approximations involving resource augmentation, i.e., speeding up individual processors. This paper studies another type of resource augmentation by allocating additional processors, a topic that has not been explored until recently. We provide polynomial-time algorithms and analysis, in which the approximation factors are dependent upon the input instances. Specifically, the factors are related to the maximum ratio of the period to the relative deadline of a task in the given task set. We also show that these algorithms unfortunately cannot achieve a constant approximation factor for general cases. Furthermore, we prove that the problem does not admit any asymptotic polynomial-time approximation scheme (APTAS) unless $\mathcal{P} = \mathcal{NP}$ when the task set has constrained deadlines, i.e., the relative deadline of a task is no more than the period of the task.

2012 ACM Subject Classification Computer systems organization → Real-time systems

Keywords and phrases multiprocessor partitioned scheduling, approximation factors

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2018.71

Related Version A full version of the paper is available at [9], `https://arxiv.org/abs/1809.04355`.

1 Introduction

The sporadic task model has been widely adopted to model recurring executions of tasks in real-time systems [28]. A sporadic real-time task τ_i is defined with a *minimum inter-arrival time* T_i, its timing constraint or *relative deadline* D_i, and its (worst-case) *execution time* C_i. A sporadic task represents an infinite sequence of task instances, also called *jobs*, that arrive with the minimum inter-arrival time constraint. That is, any two consecutive jobs of task τ_i should be temporally separated by at least T_i. When a job of task τ_i arrives at time t, the job must finish no later than its *absolute deadline* $t + D_i$. According to the Liu and Layland task model [27], the minimum inter-arrival time of a task can also be interpreted as the *period* of the task.

To schedule real-time tasks on multiprocessor platforms, there have been three widely adopted paradigms: partitioned, global, and semi-partitioned scheduling. A comprehensive survey of multiprocessor scheduling in real-time systems can be found in [15]. In this paper, we consider *partitioned scheduling*, in which tasks are statically partitioned onto processors. This means that all the jobs of a task are executed on a specific processor, which reduces the online scheduling overhead since each processor can schedule the sporadic tasks assigned on it without considering the tasks on the other processors. Moreover, we consider preemptive scheduling on each processor, i.e, a job may be preempted by another job on the processor. For scheduling sporadic tasks on one processor, the (preemptive) earliest-deadline-first (EDF) policy is optimal [27] in terms of meeting timing constraints, in the sense that if the task set is schedulable then it will also be schedulable under EDF. In EDF, the job (in the ready queue) with the earliest absolute deadline has the highest priority for execution. Alternatively, another widely adopted scheduling paradigm is (preemptive) fixed-priority (FP) scheduling, where all jobs released by a sporadic task have the same priority level.

The complexity of testing whether a task set can be feasibly scheduled on a uniprocessor depends on the relations between the relative deadlines and the minimum inter-arrival times of tasks. An input task set is said to have (1) *implicit deadlines* if the relative deadlines of sporadic tasks are equal to their minimum inter-arrival times, (2) *constrained deadlines* if the minimum inter-arrival times are no less than their relative deadlines, and (3) *arbitrary deadlines*, otherwise.

On a uniprocessor, checking the feasibility for an implicit-deadline task set is simple and well-known: the timing constraints are met by EDF if and only if the total utilization $\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i}$ is at most 100% [27]. Moreover, if every task τ_i on the processor is with $D_i \geq T_i$, it is not difficult to see that testing whether the total utilization is less than or equal to 100% is also a necessary and sufficient schedulability test. This can be achieved by considering a more stringent case which sets D_i to T_i for every τ_i. Hence, this special case of arbitrary-deadline task sets can be reformulated to task sets with implicit deadlines without any loss of precision. However, determining the schedulability for task sets with constrained or arbitrary deadlines in general is much harder, due to the complex interactions between the deadlines and the periods, and in particular is known to be co$\mathcal{NP}$-hard or co$\mathcal{NP}$-complete [17, 19, 18].

In this paper, we consider partitioned scheduling in homogeneous multiprocessor systems. Deciding if an implicit-deadline task set is schedulable on multiple processors is already $\mathcal{NP}$-complete in the strong sense under partitioned scheduling. To cope with these $\mathcal{NP}$-hardness issues, one natural approach is to focus on approximation algorithms, i.e., polynomial time algorithms that produce an approximate solution instead of an exact one. In our setting, this translates to designing algorithms that can find a feasible schedule using either (i) faster or (ii) additional processors. The goal, of course, is to design an algorithm that uses the

least speeding up or as few additional processors as possible. In general, this approach is referred to as resource augmentation and is used extensively to analyze and compare scheduling algorithms. See for example [29] for a survey and motivation on why this is a useful measure for evaluating the quality of scheduling algorithms in practice. However, such a measure also has its potential pitfalls as recently studied and reported by Chen et al. [12]. Interestingly, it turns out that there is a huge difference regarding the approximation factors depending on whether it is possible to increase the processor speed or the number of processors. As already discussed in [11], approximation by speeding up is known as the *multiprocessor partitioned scheduling problem*, and by allocating more processors is known as the *multiprocessor partitioned packing problem*. We study the latter one in this paper.

Formally, an algorithm $\mathcal{A}$ for the multiprocessor partitioned packing problem is said to have an approximation factor ρ, if given any task set $\mathbf{T}$, it can find a feasible partition of $\mathbf{T}$ on ρM^* processors, where M^* is the minimum (optimal) number of processors required to schedule $\mathbf{T}$. However, it turns out that the approximation factor is not the best measure in our setting (it is not fine-grained enough). For example, it is $\mathcal{NP}$-complete to decide if an implicit-deadline task set is schedulable on 2 processors or whether 3 processors are necessary. Assuming $\mathcal{P} \neq \mathcal{NP}$, this rules out the possibility of any efficient algorithm with approximation factor better than 3/2, as shown in [11]. (This lower bound is further lifted to 2 for sporadic tasks in Section 5.) The problem with this example is that it does not rule out the possibility of an algorithm that only needs $M^* + 1$ processors. Clearly, such an algorithm is almost as good as optimum when M^* is large and would be very desirable.[1] To get around this issue, a more refined measure is the so-called asymptotic approximation factor. An algorithm $\mathcal{A}$ has an *asymptotic* approximation factor ρ if we can find a schedule using at most $\rho M^* + \alpha$ processors, where α is a constant that does not depend on M^*. An algorithm is called an asymptotic polynomial-time approximation scheme (APTAS) if, given an arbitrary accuracy parameter $\epsilon > 0$ as input, it finds a schedule using $(1+\epsilon)M^* + O(1)$ processors and its running time is polynomial assuming ϵ is a fixed constant.

For implicit-deadline task sets, the multiprocessor partitioned scheduling problem, by speeding up, is equivalent to the Makespan problem [21], and the multiprocessor partitioned packing problem, by allocating more processors, is equivalent to the bin packing problem [20]. The Makespan problem admits polynomial-time approximation schemes (PTASes), by Hochbaum and Shmoys [22], and the bin packing problem admits asymptotic polynomial-time approximation schemes (APTASes), by de la Vega and Lueker [16, 25].

When considering sporadic task sets with constrained or arbitrary deadlines, the problem becomes more complicated. When adopting speeding-up for resource augmentation, the deadline-monotonic partitioning proposed by Baruah and Fisher [3, 4] has been shown to have a $3 - \frac{1}{M}$ speed-up factor in [10], where M is the given number of identical processors. The studies in [2, 11, 1] provide polynomial-time approximation schemes for some special cases when speeding-up is possible. The PTAS by Baruah [2] requires that $\frac{D_{\max}}{D_{\min}}, \frac{C_{\max}}{C_{\min}}, \frac{T_{\max}}{T_{\min}}$ are constants, where $D_{\max}$ ($C_{\max}$ and $T_{\max}$, respectively) is the maximum relative deadline (worst-case execution time and period, respectively) in the task set and $D_{\min}$ ($C_{\min}$ and $T_{\min}$, respectively) is the minimum relative deadline (worst-case execution time and period, respectively) in the task set. It was later shown in [11, 1] that the complexity only depends on $\frac{D_{\max}}{D_{\min}}$. If $\frac{D_{\max}}{D_{\min}}$ is a constant, there exists a PTAS developed by Chen and Chakraborty [11], which admits feasible task partitioning by speeding up the processors by $(1+\epsilon)$. The

[1] Indeed, there are (very ingenious) algorithms known for the implicit-deadline partitioning problem that use only $M^* + O(\log^2 M^*)$ processors [25], based on the connection to the bin-packing problem.

Table 1 Summary of the multiprocessor partitioned scheduling and packing problems, unless $\mathcal{P} = \mathcal{NP}$, where $\gamma = \max_{\tau_i \in \mathbf{T}} \frac{C_i}{\min\{T_i, D_i\}}$, $\lambda = \max_{\tau_i \in \mathbf{T}} \max\{\frac{T_i}{D_i}, 1\}$, and $D_{\max}$ ($D_{\min}$) is the task set's maximum (minimum) relative deadline. A $^\sharp$ marks results from this paper.

	implicit deadlines	constrained deadlines	arbitrary deadlines	arbitrary deadlines (dependent on $\frac{D_{\max}}{D_{\min}}$)
partitioned EDF scheduling	PTAS [22]	2.6322-speed up [10]	3-speed up [10]	PTAS [11] for constant $\frac{D_{\max}}{D_{\min}}$ qPTAS [1] for polynomial $\frac{D_{\max}}{D_{\min}}$
partitioned FP scheduling	$\frac{7}{4}$ [6], 1.5 [26] (extended from packing)	2.84306 speed-up [8]	3-speed up[8]	
partitioned packing	APTAS [16]	non-existence of APTAS$^\sharp$	non-existence of APTAS [11]	
		2λ-approximation$^\sharp$, asymptotic $\frac{2}{1-\gamma}$-approximation$^\sharp$, non-existence of $(2-\epsilon)$-approximation$^\sharp$		

approach in [11] deals with the multiprocessor partitioned scheduling problem as a vector scheduling problem [7] by constructing (roughly) $(1/\epsilon) \log \frac{D_{\max}}{D_{\min}}$ dimensions and then applies the PTAS of the vector scheduling problem developed by Chekuri and Khanna [7] in a black-box manner. Bansal et al. [1] exploit the special structure of the vectors and give a faster vector scheduling algorithm that is a quasi-polynomial-time approximation scheme (qPTAS) even if $\frac{D_{\max}}{D_{\min}}$ is polynomially bounded.

However, augmentation by allocating additional processors, i.e., the multiprocessor partitioned packing problem, has not been explored until recently in real-time systems. Our previous work in [11] has initiated the study for minimizing the number of processors for real-time tasks. While [11] mostly focuses on approximation algorithms for resource augmentation via speeding up, it also showed that for the multiprocessor partitioned packing problem there does not exist any APTAS for arbitrary-deadline task sets, unless $\mathcal{P} = \mathcal{NP}$. However, the proof in [11] for the non-existence of APTAS only works when the input task set $\mathbf{T}$ has *exactly* two types of tasks in which one type consists of tasks with relative deadline less than or equal to its period (i.e., $D_i \leq T_i$ for some τ_i in $\mathbf{T}$) and another type consists of tasks with relative deadline larger than its period (i.e., $D_j > T_j$ for some τ_j in $\mathbf{T}$). Therefore, it cannot be directly applied for constrained-deadline task sets.

For the results, from the literature and also this paper, related to the multiprocessor partitioned scheduling and packing problems, Table 1 provides a short summary.

Our Contributions. This paper studies the multiprocessor partitioned packing problem in much more detail. On the positive side, when the ratio of the period of a constrained-deadline task to the relative deadline of the task is at most $\lambda = \max_{\tau_i \in \mathbf{T}} \max\{\frac{T_i}{D_i}, 1\}$, in Section 3, we provide a simple polynomial-time algorithm with a 2λ-approximation factor. In Section 4, we show that the deadline-monotonic partitioning algorithm in [3, 4] has an asymptotic $\frac{2}{1-\gamma}$-approximation factor for the packing problem, where $\gamma = \max_{\tau_i \in \mathbf{T}} \frac{C_i}{\min\{T_i, D_i\}}$. In particular, when γ and λ are not constant, adopting the worst-fit or best-fit strategy in the deadline-monotonic partitioning algorithm is shown to have an $\Omega(N)$ approximation factor, where N is the number of tasks. In contrast, from [10], it is known that both strategies have a speed-up factor 3, *when the resource augmentation is to speed up processors.* We also show that speeding up processors can be much more powerful than allocating more processors. Specifically, in Section 5, we provide input instances, in which the only feasible schedule is to run each task on an individual processor but the system requires only one processor with a speed-up factor of $(1 + \epsilon)$, where $0 < \epsilon < 1$.

On the negative side, in Section 6, we show that there does not exist any asymptotic polynomial-time approximation scheme (APTAS) for the multiprocessor partitioned packing problem for task sets with constrained deadlines, unless $\mathcal{P} = \mathcal{NP}$. As there is already an APTAS for the implicit deadline case, this together with the result in [11] gives a complete picture of the approximability of multiprocessor partitioned packing for different types of task sets, as shown in Table 1.

2 System Model

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. Each of these tasks releases an infinite number of task instances, called jobs. A task τ_i is defined by (C_i, T_i, D_i), where D_i is its relative deadline, T_i is its minimum inter-arrival time (period), and C_i is its (worst-case) execution time. For a job released at time t, the next job must be released no earlier than $t + T_i$ and it must finish (up to) C_i amount of execution before the jobs absolute deadline at $t + D_i$. The *utilization* of task τ_i is denoted by $u_i = \frac{C_i}{T_i}$. We consider platforms with identical processors, i.e., the execution and timing property remains no matter which processor a task is assigned to. According to the relations of the relative deadlines and the minimum inter-arrival times of the tasks in $\mathbf{T}$, the task set can be identified to be with (1) implicit deadlines, i.e., $D_i = T_i \; \forall \tau_i$, (2) constrained deadlines, i.e., $D_i \leq T_i \; \forall \tau_i$, or (3) arbitrary deadlines, otherwise. The cardinality of a set $\mathbf{X}$ is denoted by $|\mathbf{X}|$.

In this paper we focus on partitioned scheduling, i.e., each task is statically assigned to a fixed processor and all jobs of the task is executed on the assigned processor. On each processor, the jobs related to the tasks allocated to that processor are scheduled using preemptive earliest deadline first (EDF) scheduling. This means that at each point the job with the shortest absolute deadline is executed, and if a new job with a shorter absolute deadline arrives the currently executed job is preempted and the new arriving job starts executing. A task set can be feasibly scheduled by EDF (or EDF is a feasible schedule) on a processor if the timing constraints can be fulfilled by using EDF.

2.2 Problem Definition

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

- $\mathbf{T}_j \cap \mathbf{T}_{j'} = \emptyset$ for all $j \neq j'$,
- $\cup_{j=1}^{M} \mathbf{T}_j$ is equal to the input task set $\mathbf{T}$, and
- set $\mathbf{T}_j$ can meet the timing constraints by EDF scheduling on a processor j.

▶ **Definition 1.** *The multiprocessor partitioned packing problem:* The objective is to find a feasible task partition on M identical processors with the minimum M.

We assume that $u_i \leq 100\%$ and $\frac{C_i}{D_i} \leq 100\%$ for any task τ_i since otherwise there cannot be a feasible partition.

2.3 Demand Bound Function

This paper focuses on the case where the arrival times of the sporadic tasks are not specified, i.e., they arrive according to their interarrival constraint and not according to a pre-defined pattern. Baruah et al. [5] have shown that in this case the worst-case pattern is to release the first job of tasks synchronously (say, at time 0 for notational brevity), and all subsequent jobs as early as possible. Therefore, as shown in [5], the *demand bound function* $\text{DBF}(\tau_i, t)$ of a task τ_i that specifies the maximum demand of task τ_i to be released and finished within any time interval with length t is defined as

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

The exact schedulability test of EDF, to verify whether EDF can feasibly schedule the given task set on a processor, is to check whether the summation of the demand bound functions of all the tasks is always less than t for all $t \geq 0$ [5].

3 Reduction to Bin Packing

When considering tasks with implicit deadlines, the multiprocessor partitioned packing problem is equivalent to the bin packing problem [20]. Therefore, even though the packing becomes more complicated when considering tasks with arbitrary or constrained deadlines, it is pretty straightforward to handle the problem by using existing algorithms for the bin packing problem if the maximum ratio λ of the period to the relative deadline among the tasks, i.e., $\lambda = \max_{\tau_i \in \mathbf{T}} \max\{\frac{T_i}{D_i}, 1\}$, is not too large.

For a given task set $\mathbf{T}$, we can basically transform the input instance to a related task instance $\mathbf{T}^\dagger$ by creating task $\tau_i^\dagger$ based on task τ_i in $\mathbf{T}$ such that

- $T_i^\dagger$ is D_i, $C_i^\dagger$ is C_i, and $D_i^\dagger$ is D_i when $T_i \geq D_i$ for τ_i, and
- $D_i^\dagger$ is $T_i^\dagger$, $C_i^\dagger$ is C_i and $T_i^\dagger$ is T_i when $T_i < D_i$ for τ_i.

Now, we can adopt any greedy fitting algorithms (i.e., a task is assigned to "one" allocated processor that is feasible; otherwise, a new processor is allocated and the task is assigned to the newly allocated processor) for the bin packing problem by considering only the utilization of transformed tasks in $\mathbf{T}^\dagger$ for the multiprocessor partitioned packing problem, as presented in [30, Chapter 8]. The construction of $\mathbf{T}^\dagger$ has a time complexity of $O(N)$, and the greedy fitting algorithm has a time complexity of $O(NM)$.

▶ **Theorem 2.** *Any greedy fitting algorithm by considering $\mathbf{T}^\dagger$ for task assignment is a 2λ-approximation algorithm for the multiprocessor partitioned packing problem.*

Proof. Clearly, as we only reduce the relative deadline and the periods, the timing parameters in $\mathbf{T}^\dagger$ are more stringent than in $\mathbf{T}$. Hence, a feasible task partition for $\mathbf{T}^\dagger$ on M processors also yields a corresponding feasible task partition for $\mathbf{T}$ on M processors. As $\mathbf{T}^\dagger$ has implicit deadlines, we know that any task subset in $\mathbf{T}^\dagger$ with total utilization no more than 100% can be feasibly scheduled by EDF on a processor, and therefore the original tasks in that subset as well. For any greedy fitting algorithms that use M processors, using the same proof as in [30, Chapter 8], we get $\sum_{\tau_i \in \mathbf{T}^\dagger} \frac{C_i^\dagger}{T_i^\dagger} > \frac{M}{2}$.

By definition, we know that $\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i} \geq \sum_{\tau_i^\dagger \in \mathbf{T}^\dagger} \frac{C_i^\dagger}{\lambda T_i^\dagger} > \frac{M}{2\lambda}$. Therefore, any feasible solution for $\mathbf{T}$ uses at least $\frac{M}{2\lambda}$ processors and the approximation factor is hence proved. ◀

4 Deadline-Monotonic Partitioning under EDF Scheduling

This section presents the worst-case analysis of the deadline-monotonic partitioning strategy, proposed by Baruah and Fisher [4, 3], for the multiprocessor partitioned packing problem. Note that the underlying scheduling algorithm is EDF but the tasks are considered in the *deadline-monotonic* (DM) order. Hence, in this section, we index the tasks accordingly from the shortest relative deadline to the longest, i.e., $D_i \leq D_j$ if $i < j$. Specifically, in the DM partitioning, the approximate demand bound function $\text{DBF}^*(\tau_i, t)$ is used to approximate Eq. (1), where

$$\text{DBF}^*(\tau_i, t) = \begin{cases} 0 & \text{if } t < D_i \\ \left(\frac{t - D_i}{T_i} + 1\right) C_i & \text{otherwise.} \end{cases} \tag{2}$$

Algorithm 1 Deadline-Monotonic Partitioning.

Input: set $\mathbf{T}$ of N tasks;
re-index (sort) tasks such that $D_i \leq D_j$ for $i < j$;
$M \leftarrow 1$, $\mathbf{T}_1 \leftarrow \{\tau_1\}$;
for $i = 2$ to N **do**
  if $\exists m \in \{1, 2, \ldots, M\}$ such that both (3) and (4) hold **then**
    choose $m \in \{1, 2, \ldots, M\}$ **by preference** such that both (3) and (4) hold;
    assign τ_i to processor m with $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$;
  else
    $M \leftarrow M + 1$; $\mathbf{T}_M \leftarrow \{\tau_i\}$;
  end if
end for
return feasible task partition $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$;

Even though the DM partitioning algorithm in [4, 3] is designed for the multiprocessor partitioned scheduling problem, it can be easily adapted to deal with the multiprocessor partitioned packing problem. For completeness, we revise the algorithm in [4, 3] for the multiprocessor partitioned packing problem and present the pseudo-code in Algorithm 1. As discussed in [4, 3], when a task τ_i is considered, a processor m among the allocated processors where both the following conditions hold

$$C_i + \sum_{\tau_j \in \mathbf{T}_m} \text{DBF}^*(\tau_j, D_i) \leq D_i \tag{3}$$

$$u_i + \sum_{\tau_j \in \mathbf{T}_m} u_j \leq 1 \tag{4}$$

is selected to assign task τ_i, where $\mathbf{T}_m$ is the set of the tasks (as a subset of $\{\tau_1, \tau_2, \ldots, \tau_{i-1}\}$), which have been assigned to processor m before considering τ_i. If there is no m where both Eq. (3) and Eq. (4) hold, a new processor is allocated and task τ_i is assigned to the new processor. The order in which the already allocated processors are considered depends on the fitting strategy:

- *first-fit* (FF) strategy: choosing the feasible m with the minimum index;
- *best-fit* (BF) strategy: choosing, among the feasible processors, m with the maximum approximate demand bound at time D_i;
- *worst-fit* (WF) strategy: choosing m with the minimum approximate demand bound at time D_i.

For a given number of processors, it has been proved in [10] that the speed-up factor of the DM partitioning is at most 3, independent from the fitting strategy. However, if the objective is to minimize the number of allocated processors, we will show that DM partitioning has an approximation factor of at least $\frac{N}{4}$ (in the worst case) when the best-fit or worst-fit strategy is adopted. We will prove this by explicitly constructing two concrete task sets with this property. Afterwards, we show that the asymptotic approximation factor of DM partitioning is at most $\frac{2}{1-\gamma}$ for packing, where $\gamma = \max_{\tau_i \in \mathbf{T}} \frac{C_i}{\min\{T_i, D_i\}}$.

▶ **Theorem 3.** *The approximation factor of the deadline-monotonic partitioning algorithm with the best-fit strategy is at least* $\frac{N}{4}$ *when* $N \geq 8$ *and the schedulability test is based on Eq.* (3) *and Eq.* (4)*.*

Proof. The theorem is proven by providing a task set that can be scheduled on two processors but where Algorithm 1 when applying the best-fit strategy uses $\frac{N}{2}$ processors. Under the assumption that $K \geq 4$ is an integer, N is $2K$, and H is sufficiently large, i.e., $H \gg K^K$, such a task set can be constructed as:

- Let $D_1 = 1$, $C_1 = 1/K$, and $T_1 = H$.
- For $i = 2, 4, \ldots, 2K$, let $D_i = K^{\frac{i}{2}-1}$, $C_i = K^{\frac{i}{2}-2}$, and $T_i = D_i$.
- For $i = 3, 5, \ldots, 2K-1$, let $D_i = K^{\frac{i-1}{2}}$, $C_i = K^{\frac{i-1}{2}} - K^{\frac{i-1}{2}-1}$, and $T_i = H$.

The task set can be scheduled on two processors under EDF if all tasks with an odd index are assigned to processor 1 and all tasks with an even index are assigned to processor 2. On the other hand, the best-fit strategy assigns τ_i to processor $\lceil \frac{i}{2} \rceil$. The resulting solution uses K processors. Details are in the Appendix in [9]. ◀

▶ **Theorem 4.** *The approximation factor of the deadline-monotonic partitioning algorithm with the worst-fit strategy is at least* $\frac{N}{4}$ *when the schedulability test is based on Eq.* (3) *and Eq.* (4)*.*

Proof. The proof is very similar to the proof of Theorem 3, considering the task set:

- Let $D_1 = 1$, $C_1 = 1$, and $T_1 = H$.
- For $i = 2, 4, \ldots, 2K$, let $D_i = K^{\frac{i}{2}}$, $C_i = K^{\frac{i}{2}-1}$, and $T_i = D_i$.
- For $i = 3, 5, \ldots, 2K-1$, let $D_i = K^{\frac{i-1}{2}}$, $C_i = K^{\frac{i-1}{2}} - K^{\frac{i-1}{2}-1}$, and $T_i = H$.

Odd tasks are assigned to processor 1 and even tasks to processor 2 the task set is schedulable while τ_i is assigned to processor $\lceil \frac{i}{2} \rceil$ using the worst-fit strategy. Details are in the Appendix in [9]. ◀

▶ **Theorem 5.** *The DM partitioning algorithm is an asymptotic* $\frac{2}{1-\gamma}$*-approximation algorithm for the multiprocessor partitioned packing problem, when* $\gamma = \max_{\tau_i \in \boldsymbol{T}} \frac{C_i}{\min\{T_i, D_i\}}$ *and* $\gamma < 1$*.*

Proof. We consider the task τ_l which is the task that is responsible for the last processor that is allocated by Algorithm 1. The other processors are categorized into two disjoint sets $\mathbf{M}_1$ and $\mathbf{M}_2$, depending on whether Eq. (3) or Eq. (4) is violated when Algorithm 1 tries to assign τ_l (if both conditions are violated, the processor is in $\mathbf{M}_1$). The two sets are considered individually and the maximum number of processors in both sets is determined based on the minimum utilization for each of the processors. Afterwards, a necessary condition for the amount of processors that is at least needed for a feasible solution is provided and the relation between the two values proves the theorem. Details can be found in the Appendix in [9]. ◀

5 Hardness of Approximations

It has been shown in [11, 2] that a PTAS exists for augmenting the resources by speeding up. A straightforward question is to see whether such PTASes will be helpful for bounding the lower or upper bounds for multiprocessor partitioned packing. Unfortunately, the following theorem shows that using speeding up to get a lower bound for the number of required processors is not useful.

▶ **Theorem 6.** *There exists a set of input instances, in which the number of allocated processors is up to* N*, while the task set can be feasibly scheduled by EDF with a speed-up factor* $(1+\epsilon)$ *on a processor, where* $0 < \epsilon < 1$*.*

Proof. We provide a set of input instances, with the property described in the statement:

- Let $D_1 = 1$, $C_1 = 1$, and $T_1 = \frac{(1+\epsilon)^{N-2}}{\epsilon^{N-1}}$.

- For any $i = 2, 3, \ldots, N$, let $D_i = \frac{(1+\epsilon)^{i-2}}{\epsilon^{i-1}}$, $C_i = D_i$, and $T_i = \frac{(1+\epsilon)^{N-2}}{\epsilon^{N-1}}$.

Since $C_i = D_i$ for any task τ_i, assigning any two tasks on the same processor is infeasible without speeding up. Therefore, the only feasible processor allocation is N processors and to assign each task individually on one processor. However, by speeding up the system by a factor $1 + \epsilon$, the tasks can be feasibly scheduled on one processor due to $\sum_{i=1}^{N} \frac{dbf(\tau_i,t)}{1+\epsilon} \leq t$ for any $t > 0$. A proof is in the Appendix in [9]. Hence, the gap between these two types of resource augmentation is up to N. ◀

Moreover, the following theorem shows the inapproximability for a factor 2 without adopting asymptotic approximation.

▶ **Theorem 7.** *For any $\epsilon > 0$, there is no polynomial-time approximation algorithm with an approximation factor of $2 - \epsilon$ for the multiprocessor partitioned packing problem, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Suppose that there exists such a polynomial-time algorithm $\mathcal{A}$ with approximation factor $2 - \epsilon$. $\mathcal{A}$ can be used to decide if a task set $\mathbf{T}$ is schedulable on a uniprocessor, which would contradict the *co*$\mathcal{NP}$-hardness [17] of this problem. Indeed, we simply run $\mathcal{A}$ on the input instance. If $\mathcal{A}$ returns a feasible schedule using one processor, we already have a uniprocessor schedule. On the other hand, if $\mathcal{A}$ requires at least two processors, then we know that any optimum solution needs $\geq \left\lceil \frac{2}{2-\epsilon} \right\rceil = 2$ processors, implying that the task set $\mathbf{T}$ is not schedulable on a uniprocessor. ◀

6 Non-Existence of APTAS for Constrained Deadlines

We now show that there is no APTAS when considering constrained-deadline task sets, unless $\mathcal{P} = \mathcal{NP}$. The proof is based on an L-reduction (informally an approximation preserving reduction) from a special case of the *vector packing problem*, i.e., the 2D dominated vector packing problem.

6.1 The 2D Dominated Vector Packing Problem

The *vector packing problem* is defined as follows:

▶ **Definition 8.** *The vector packing problem:* Given a set $\mathbf{V}$ of vectors $[v_1, v_2, \ldots, v_N]$ with d dimensions, in which $1 \geq v_{i,j} \geq 0$ is the value for vector v_i in the j-th dimension, the problem is to partition $\mathbf{V}$ into M parts $\mathbf{V}_1, \ldots, \mathbf{V}_M$ such that M is minimized and each part $\mathbf{V}_m$ is feasible in the sense that $\sum_{v_i \in \mathbf{V}_m} v_{i,j} \leq 1$ for all $1 \leq j \leq d$. That is, for each dimension j, the sum of the j-th coordinates of the vectors in $\mathbf{V}_m$ is at most 1.

We say that a subset $\mathbf{V}'$ of $\mathbf{V}$ can be *feasibly packed in a bin* if $\sum_{v_i \in \mathbf{V}'} v_{i,j} \leq 1$ for all j-th dimensions. Note that for $d = 1$ this is precisely the bin-packing problem. The vector packing problem does not admit any polynomial-time asymptotic approximation scheme even in the case of $d = 2$ dimensions, unless $\mathcal{P} = \mathcal{NP}$ [31].

Specifically, the proof in [11] for the non-existence of APTAS for task sets with arbitrary deadlines comes from an L-reduction from the 2-dimensional vector packing problem as follows: For a vector v_i in $\mathbf{V}$, a task τ_i is created with $D_i = 1$, $C_i = v_{i,2}$, and $T_i = \frac{v_{i,2}}{v_{i,1}}$. However, a trivial extension from [11] to constrained deadlines does not work, since for the transformation of the task set we need to assume that $v_{i,1} \leq v_{i,2}$ for any $v_i \in \mathbf{V}$ so that $T_i \geq 1 = D_i$ for every reduced task τ_i. This becomes problematic, as one dimension in the vectors in such input instances for the two-dimensional vector packing problem can be

totally ignored, and the input instance becomes a special case equivalent to the traditional bin-packing problem, which admits an APTAS. We will show that the hardness is equivalent to a special case of the two-dimensional vector packing problem, called the *two-dimensional dominated vector packing* (2D-DVP) problem, in Section 6.2.

▶ **Definition 9.** The *two-dimensional dominated vector packing* (2D-DVP) problem is a special case of the two-dimensional vector packing problem with following conditions for each vector $v_i \in \mathbf{V}$:

- $v_{i,1} > 0$, and
- if $v_{i,2} \neq 0$, then $v_{i,1}$ is dominated by $v_{i,2}$, i.e., $v_{i,2} > v_{i,1}$.

Moreover, we further assume that $v_{i,1}$ and $v_{i,2}$ are rational numbers for every $v_i \in \mathbf{V}$.

Here, some tasks are created with implicit deadlines (based on vector v_i if $v_{i,2}$ is 0) and some tasks with strictly constrained deadlines (based on vector v_i if $v_{i,2}$ is not 0). However, the 2D-DVP problem is a special case of the two-dimensional vector packing problem, and the implication for $v_{i,2} > v_{i,1}$ when $v_{i,2} \neq 0$ does not hold in the proof in [31]. We note, that the proof for the non-existence of an APTAS for the two-dimensional vector packing problem in [31] is erroneous. However, the result still holds. Details are in the Appendix in [9]. Therefore, we will provide a proper L-reduction in Section 6.3 to show the non-existence of APTAS for the multiprocessor partitioned packing problem for tasks with constrained deadlines.

6.2 2D-DVP Problem and Packing Problem

We now show that the packing problem is at least as hard as the 2D-DVP problem from a complexity point of view. For vector v_i with $v_{i,2} > v_{i,1}$, we create a corresponding task τ_i with

$$D_i = 1, \quad C_i = v_{i,2}, \quad T_i = \frac{v_{i,2}}{v_{i,1}}.$$

Clearly, $D_i < T_i$ for such tasks. Let H be a *common multiple*, not necessary the least, of the periods T_i of the tasks constructed above. By the assumption that all the values in the 2D-DVP problem are rational numbers and $v_{i,1} > 0$ for every vector v_i, we know that H exists and can be calculated in $O(N)$. For vector v_i with $v_{i,2} = 0$, we create a corresponding implicit-deadline task τ_i with

$$T_i = D_i = H, \quad C_i = v_{i,1} T_i.$$

The following lemma shows the related schedulability condition.

▶ **Lemma 10.** *Suppose that the set $\boldsymbol{T}_m$ of tasks assigned on a processor consists of (1) strictly constrained-deadline tasks, denoted by $\boldsymbol{T}_m^<$, with a common relative deadline $1 = D$ and (2) implicit-deadline tasks, i.e., $\boldsymbol{T}_m \setminus \boldsymbol{T}_m^<$, in which the period is a common integer multiple H of the periods of the strictly constrained-deadline tasks. EDF schedule is feasible for the set $\boldsymbol{T}_m$ of tasks on a processor if and only if*

$$\sum_{\tau_i \in \boldsymbol{T}_m^<} C_i \leq 1 \text{ and } \sum_{\tau_i \in \boldsymbol{T}_m} u_i \leq 1.$$

Proof.

Only If. This is straightforward as the task set cannot 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 $t < D$, then $\sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, t) = 0$. When $D \leq t < H$, we have

$$\sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, t) = \sum_{\tau_i \in \mathbf{T}_m^<} \left(\left\lfloor \frac{t-D}{T_i} \right\rfloor + 1 \right) \times C_i \leq \sum_{\tau_i \in \mathbf{T}_m^<} \left(\frac{t-D}{T_i} + 1 \right) \times C_i$$
$$\leq \sum_{\tau_i \in \mathbf{T}_m^<} C_i + (t-D)u_i \leq D + (t-D) = t. \tag{5}$$

Moreover, with $\sum_{\tau_i \in \mathbf{T}_m} u_i \leq 1$, we know that when $t = H$

$$\sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, H) = \sum_{\tau_i \in \mathbf{T}_m^<} \left(\left\lfloor \frac{H-D}{T_i} \right\rfloor + 1 \right) \times C_i + \sum_{\tau_i \in \mathbf{T}_m \setminus \mathbf{T}_m^<} \frac{H}{T_i} C_i$$
$$=_1 \sum_{\tau_i \in \mathbf{T}_m^<} \frac{H}{T_i} C_i + \sum_{\tau_i \in \mathbf{T}_m \setminus \mathbf{T}_m^<} \frac{H}{T_i} C_i = H \left(\sum_{\tau_i \in \mathbf{T}_m} u_i \right) \leq H,$$

where $=_1$ comes from the fact that $\frac{H}{T_i}$ is an integer for any τ_i in $\mathbf{T}_m^<$ and $T_i > D > 0$ so that $\left\lfloor \frac{H-D}{T_i} \right\rfloor + 1$ is equal to $\frac{H}{T_i}$.

For any value $t > H$, the value of $\sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, t)$ is equal to $\sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, t-H) + \sum_{\tau_i \in \mathbf{T}_m} \text{DBF}(\tau_i, H)$. Therefore, we know that if $\sum_{\tau_i \in \mathbf{T}_m^<} \frac{C_i}{D} \leq 1$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i \leq 1$, the task set $\mathbf{T}_m$ can be feasibly scheduled by EDF. ◀

▶ **Theorem 11.** *If there does not exist any APTAS for the 2D-DVP problem, unless $\mathcal{P} = \mathcal{NP}$, there also does not exist any APTAS for the multiprocessor partitioned packing problem with constrained-deadline task sets.*

Proof. Clearly, the reduction in this section from the 2D-DVP problem to the multiprocessor partitioned packing problem with constrained deadlines 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 $\mathbf{T}'$. By Lemma 10, the subset $\mathbf{T}_m$ of the constructed tasks can be feasibly scheduled by EDF on a processor if and only if $\sum_{\tau_i \in \mathbf{T}_m^<} C_i = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}_m)} v_{i,2} \leq 1$ and $\sum_{\tau_i \in \mathbf{T}_m} u_i = \sum_{\tau_i \in \mathbf{V}(\mathbf{T}_m)} v_{i,1} \leq 1$.

Therefore, it is clear that the above reduction is a perfect approximation preserving reduction. That is, an algorithm with a ρ (asymptotic) approximation factor for the multiprocessor partitioned packing problem can easily lead to a ρ (asymptotic) approximation factor for the 2D-DVP problem. ◀

6.3 Hardness of the 2D-DVP problem

Based on Theorem 11, we are going to show that there does not exist APTAS for the 2D-DVP problem, which also proves the non-existence of APTAS for the multiprocessor partitioned packing problem with constrained deadlines.

▶ **Theorem 12.** *There does not exist any APTAS for the 2D-DVP problem, unless $\mathcal{P} = \mathcal{NP}$.*

Proof. This is proved by an L-reduction, following a similar strategy in [31] by constructing an L-reduction from the Maximum Bounded 3-Dimensional Matching (MAX-3-DM), which is MAX SNP-complete [24]. Details are in the Appendix in [9], where a short comment regarding an erroneous observation in [31] is also provided. ◀

The following theorem results from Theorems 11 and 12.

▶ **Theorem 13.** *There does not exist any APTAS for the multiprocessor partitioned packing problem for constrained-deadline task sets, unless $\mathcal{P} = \mathcal{NP}$.*

7 Concluding Remarks

This paper studies the partitioned multiprocessor packing problem to minimize the number of processors needed for multiprocessor partitioned scheduling. Interestingly, there turns out to be a huge difference (technically) in whether one is allowed faster processors or additional processors. Our results are summarized in Table 1. For general cases, the upper bound and lower bound for the first-fit strategy in the deadline-monotonic partitioning algorithm are both open. The focus of this paper is the multiprocessor partitioned packing problem. If *global scheduling* is allowed, in which a job can be executed on different processors, the problem of minimizing the number of processors has been also recently studied in a more general setting by Chen et al. [14, 13] and Im et al. [23]. They do not explore any periodicity of the job arrival patterns. Among them, the state-of-the-art online competitive algorithm has an approximation factor (more precisely, competitive factor) of $O(\log \log M)$ by Im et al. [23]. These results are unfortunately not applicable for the multiprocessor partitioned packing problem since the jobs of a sporadic task may be executed on different processors.

References

1. Nikhil Bansal, Cyriel Rutten, Suzanne van der Ster, Tjark Vredeveld, and Ruben van der Zwaan. Approximating Real-Time Scheduling on Identical Machines. In *LATIN: Theoretical Informatics - 11th Latin American Symposium*, pages 550–561, 2014.
2. Sanjoy Baruah. The Partitioned EDF Scheduling of Sporadic Task Systems. In *Real-Time Systems Symposium (RTSS)*, pages 116–125, 2011.
3. Sanjoy K. Baruah and Nathan Fisher. The Partitioned Multiprocessor Scheduling of Sporadic Task Systems. In *Real-Time Systems Symposium (RTSS)*, pages 321–329, 2005.
4. Sanjoy K. Baruah and Nathan Fisher. The Partitioned Multiprocessor Scheduling of Deadline-Constrained Sporadic Task Systems. *IEEE Trans. Computers*, 55(7):918–923, 2006.
5. Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Real-Time Systems Symposium (RTSS)*, pages 182–190, 1990.
6. Almut Burchard, Jörg Liebeherr, Yingfeng Oh, and Sang Hyuk Son. New Strategies for Assigning Real-Time Tasks to Multiprocessor Systems. *IEEE Trans. Computers*, 44(12):1429–1442, 1995.
7. Chandra Chekuri and Sanjeev Khanna. On Multidimensional Packing Problems. *SIAM J. Comput.*, 33(4):837–851, 2004. doi:10.1137/S0097539799356265.
8. Jian-Jia Chen. Partitioned Multiprocessor Fixed-Priority Scheduling of Sporadic Real-Time Tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 251–261, 2016.
9. Jian-Jia Chen, Nikhil Bansal, Samarjit Chakraborty, and Georg von der Brüggen. Packing Sporadic Real-Time Tasks on Identical Multiprocessor Systems. *Computing Research Repository (CoRR)*, 2018. http://arxiv.org/abs/1809.04355.
10. Jian-Jia Chen and Samarjit Chakraborty. Resource Augmentation Bounds for Approximate Demand Bound Functions. In *IEEE Real-Time Systems Symposium*, pages 272–281, 2011.
11. Jian-Jia Chen and Samarjit Chakraborty. Partitioned Packing and Scheduling for Sporadic Real-Time Tasks in Identical Multiprocessor Systems. In *ECRTS*, pages 24–33, 2012.

12 Jian-Jia Chen, Georg von der Brüggen, Wen-Hung Huang, and Robert I Davis. On the Pitfalls of Resource Augmentation Factors and Utilization Bounds in Real-Time Scheduling. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 9:1–9:25, 2017.

13 Lin Chen, Nicole Megow, and Kevin Schewior. An O(log m)-Competitive Algorithm for Online Machine Minimization. In *Symposium on Discrete Algorithms, SODA*, pages 155–163, 2016.

14 Lin Chen, Nicole Megow, and Kevin Schewior. The Power of Migration in Online Machine Minimization. In *Symposium on Parallelism in Algorithms and Architectures*, pages 175–184, 2016.

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

16 Wenceslas Fernandez de la Vega and George S. Lueker. Bin packing can be solved within 1+epsilon in linear time. *Combinatorica*, 1(4):349–355, 1981. doi:10.1007/BF02579456.

17 Friedrich Eisenbrand and Thomas Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *Symposium on Discrete Algorithms (SODA)*, pages 1029–1034, 2010. URL: http://www.siam.org/proceedings/soda/2010/SODA10_083_eisenbrandf.pdf.

18 Pontus Ekberg and Wang Yi. Uniprocessor Feasibility of Sporadic Tasks Remains coNP-Complete under Bounded Utilization. In *IEEE Real-Time Systems Symposium, RTSS*, pages 87–95, 2015.

19 Pontus Ekberg and Wang Yi. Uniprocessor Feasibility of Sporadic Tasks with Constrained Deadlines Is Strongly coNP-Complete. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 281–286, 2015.

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

21 Ronald L. Graham. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.

22 Dorit S. Hochbaum and David B. Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *J. ACM*, 34(1):144–162, 1987. doi:10.1145/7531.7535.

23 Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Clifford Stein. An O(log log m)-competitive Algorithm for Online Machine Minimization. In *Real-Time Systems Symposium, (RTSS)*, pages 343–350, 2017.

24 Viggo Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inf. Process. Lett.*, 37(1):27–35, January 1991.

25 N. Karmarkar and R. M. Karp. An Efficient Approximation Scheme for the One-Dimensional Bin-Packing Problem. In *Symp. on Foundations of Computer Science (FOCS)*, pages 312–320, 1982.

26 Andreas Karrenbauer and Thomas Rothvoß. A 3/2-Approximation Algorithm for Rate-Monotonic Multiprocessor Scheduling of Implicit-Deadline Tasks. In *International Workshop of Approximation and Online Algorithms WAOA*, pages 166–177, 2010. doi:10.1007/978-3-642-18318-8_15.

27 C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, 1973.

28 A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1983.

29 K. Pruhs, E. Torng, and J. Sgall. Online scheduling. In Joseph Y. T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 15, pages 15:1–15:41. Chapman and Hall/CRC, 2004.

30 Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.

31 Gerhard J. Woeginger. There is no Asymptotic PTAS for Two-Dimensional Vector Packing. *Inf. Process. Lett.*, 64(6):293–297, 1997. doi:10.1016/S0020-0190(97)00179-8.