Real-Time Syst (2013) 49:475–516
DOI 10.1007/s11241-013-9181-5

Resource augmentation for uniprocessor and multiprocessor partitioned scheduling of sporadic real-time tasks

Jian-Jia Chen · Samarjit Chakraborty

Published online: 10 April 2013

Abstract Although the earliest-deadline-first (EDF) policy is known to be optimal for preemptive real-time task scheduling in uniprocessor systems, the schedulability analysis problem has recently been shown to be $co\mathcal{NP}$-hard. Therefore, approximation algorithms, and in particular, approximations based on *resource augmentation* have attracted a lot of attention for both uniprocessor and multiprocessor systems. Resource augmentation based approximations assume a certain speedup of the processor(s). Using the notion of *approximate demand bound function* (DBF), in this paper we show that for uniprocessor systems the resource augmentation factor is at most $\frac{2e-1}{e} \approx 1.6322$, where e is the Euler number. We approximate the DBF using a linear approximation when the analysis interval length of interest is larger than the relative deadline of the task. For identical multiprocessor systems with M processors and constrained-deadline task sets, we show that the deadline-monotonic partitioning (that has been proposed by Baruah and Fisher) with the approximate DBF leads to an approximation factor of $\frac{3e-1}{e} - \frac{1}{M} \approx 2.6322 - \frac{1}{M}$ with respect to resource augmentation. We also show that the corresponding factor is $3 - \frac{1}{M}$ for arbitrary-deadline task sets. The best known results so far were $3 - \frac{1}{M}$ for constrained-deadline tasks and $4 - \frac{2}{M}$ for arbitrary-deadline ones. Our tighter analysis exploits the structure of the approximate DBF directly and uses the processor utilization violations (which were ignored in all previous analysis) for analyzing resource augmentation factors. We also provide concrete input instances to show that the lower bound on the resource augmentation factor for uniprocessor systems—using the above approximate

J.-J. Chen (✉)
Department of Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany
e-mail: jian-jia.chen@kit.edu

S. Chakraborty
Institute for Real-Time Computer Systems, Technical University of Munich (TUM), Munich, Germany
e-mail: samarjit@tum.de

DBF—is 1.5, and the corresponding bound is 2.5 for identical multiprocessor systems with an arbitrary order of fitting and a large number of processors. Further, we also provide a polynomial-time approximation scheme (PTAS) to derive near-optimal solutions under the assumption that the ratio of the maximum relative deadline to the minimum relative deadline of tasks is a constant, which is a more relaxed assumption compared to the assumptions required for deriving such a PTAS in the past.

Keywords Approximate demand bound function · DBF · Resource augmentation · Approximation · PTAS · Schedulability analysis

1 Introduction

The sporadic task model has been widely adopted in the real-time systems domain (Mok 1983). In this model, 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 historical reasons, the minimum inter-arrival time is also referred to as the *period* of the task. A sporadic task is an infinite sequence of task instances, referred to as *jobs*, where two consecutive job arrivals from a task should be separated by at least the minimum inter-arrival time of the task. When a job of task τ_i arrives at the system at time t, its *absolute deadline* is set to $t + D_i$, and the job should finish no later than its absolute deadline. Moreover, a job of task τ_i is said to be *legally* released (or arrives *legally*) at time t if no other job of task τ_i arrived in time interval $(t - T_i, t]$. A (sporadic) task set consists of several sporadic tasks. A scheduling policy (algorithm) is said to *feasibly* schedule task set $\mathbf{T}$ if, for any combination of the legally-arrived jobs of all the tasks in the set, it produces a corresponding schedule that meets their absolute deadlines. A task set $\mathbf{T}$ is said to be *feasible* if such schedules exist. To test for feasibility under a given scheduling policy, the schedulability test (analysis) needs to verify whether the (sporadic) task set can meet its deadline constraints under this policy.

To feasibly schedule a preemptive task set on a uniprocessor system, the earliest-deadline-first (EDF) policy has been shown to be optimal (Liu and Layland 1973). Here, at any point in time, the job with the earliest *absolute deadline* has the highest priority for execution. That is, a task set is feasible if and only if the schedule obtained by applying EDF does not violate any deadline constraints. For a task set $\mathbf{T}$ with *implicit* deadlines, in which the relative deadline of a task is equal to its minimum inter-arrival time for each sporadic task in $\mathbf{T}$, the schedulability test under EDF boils down to verifying whether the total utilization $\sum_{\tau_i \in \mathbf{T}} \frac{C_i}{T_i}$ is at most 100 % (Liu and Layland 1973), in which the utilization of task τ_i is defined as $\frac{C_i}{T_i}$.

However, the periodicity and the relative deadline requirements need not always necessarily be the same. When the relative deadline of a task is no more than its minimum inter-arrival time, such a task is said to have *constrained* deadlines. If the relative deadline of a task has no specific relation with its minimum inter-arrival time, such a task is said to have *arbitrary* deadlines. Specifically, when all the tasks are with constrained deadlines, the task set is said to have constrained deadlines. If there is (at least) a task with relative deadline larger than the period, such a task set is said to have arbitrary deadlines.

Although EDF is optimal for schedulability, verifying schedulability under EDF is not an easy problem for task sets with constrained or arbitrary deadlines, as the

problem has been shown to be $co\mathcal{NP}$-hard (Eisenbrand and Rothvoß 2010). To provide an exact schedulability test, Baruah et al. (1990) show that the critical instance is when the first job of all tasks are released synchronously (say, at time $t = 0$), and the subsequent jobs arrive as rapidly as legally possible. Based on this observation, the *demand bound function* $\text{DBF}(\tau_i, t)$, as in Baruah et al. (1990), of a task τ_i within any time interval of length t can be defined as the maximum demand arising from jobs of task τ_i that needs to be satisfied for all deadlines to be met. In other words, this demand includes all jobs of τ_i whose release times and deadlines span across a time interval of length t. This is given by:

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

The schedulability test now involves checking whether $\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, t) \leq t, \forall t \geq 0$. However, as the demand bound function is not a continuous function, this schedulability test of verifying that the summation of the demand bound functions of all the tasks for all $t \geq 0$ requires a pseudo-polynomial number of checks in the worst case (if the condition is violated for a certain t then such a t has a pseudo-polynomial upper bound and it is not required to check for all $t \geq 0$).

To reduce this time complexity for the schedulability test, approximations of the demand bound function in (1) have been proposed by Chakraborty et al. (2002) and Albers and Slomka (2004, 2005). Their approaches consider only a limited (constant or polynomial-bounded) number of time interval lengths for which the demand bound function is computed and approximates it with a linear function for the other intervals. The simplest approximation of the demand bound function in (1) of task τ_i for a given t is to take a linear approximation when t is larger than the relative deadline D_i. We denote such an approximation as the *approximate demand bound function* in this paper.

This schedulability analysis problem is further complicated in modern computing systems with multiprocessor platforms. For scheduling sporadic tasks on multiprocessors, there are three alternative paradigms: *partitioned*, *global*, and *semi-partitioned* scheduling. The partitioned scheduling approach partitions tasks statically among the available processors. That is, each task is statically assigned to a processor and executes all its jobs on that processor. The global scheduling approach allows a job to migrate from one processor to another. The semi-partitioned scheduling approach decides whether a task is to be divided into subtasks statically and how each task/subtask is then assigned to a processor statically. A comprehensive survey of multiprocessor scheduling in real-time systems can be found in Davis and Burns (2011).

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 (Mok 1983). As a result, efficient algorithms will need to rely on approximations, unless $\mathcal{P} = \mathcal{NP}$. For task sets with implicit deadlines, this problem is equivalent to the Makespan problem (Graham 1969), in which the objective is to minimize the maximum utilization among the processors after task partitioning. Therefore, the largest-utilization-task-first strategy proposed in Graham (1969) leads to a $\frac{4}{3}$-approximation, i.e., the processor with the maximum

utilization has an utilization of at most $\frac{4}{3}$ times the optimal solution. Also, Hochbaum and Shmoys (1987) provided a polynomial-time approximation scheme.

Baruah and Fisher (2005) provided the first non-trivial solution for partitioned scheduling on identical multiprocessor systems with arbitrary- or constrained-deadlined task sets. The problem, of course, is still $\mathcal{NP}$-hard in the strong sense. Baruah and Fisher (2005, 2006) proposed using deadline-monotonic task partitioning by using the approximate demand bound function (a linear approximation when t is larger than the relative deadline D_i).

Please note that, for the clarity of presentations, we will use *deadline-monotonic* ordering to represent that the tasks are considered iteratively from the shortest relative deadline for task assignment. The *deadline-monotonic partitioning* algorithm assigns the task under consideration to a processor which is EDF schedulable when using the approximate demand bound function for the schedulability test. Moreover, Fisher also discussed the difficulty of partitioned scheduling of sporadic real-time tasks in Fisher (2009). Baruah and Fisher proved that deadline-monotonic partitioning based on the approximate demand bound function for task partitioning has a $4 - \frac{2}{M}$ (respectively, $3 - \frac{1}{M}$) resource augmentation factor for arbitrary-deadline (constrained-deadline, respectively) task sets (Baruah and Fisher 2005, 2006), where M is the number of processors. That is, for a ρ resource augmentation factor of an algorithm $\mathcal{A}$ for the studied problem, it guarantees that

> *If a sporadic task set is feasible on M identical processors, the task partition derived from the algorithm $\mathcal{A}$ is feasible on each processor when speeding it up to ρ times its speed in the original platform.*

The analysis in Baruah and Fisher (2005, 2006) is based on the observation that the approximate demand bound function leads to a 2-approximation for any $t \geq 0$, which is tight with respect to the approximation on the demand bound function. However, it is not tight when considering resource augmentation, which will be explained in detail in Sect. 2.6.

Some schedulability conditions can also be used to derive resource augmentation factors. For example, the well-known Liu & Layland bound of $N(2^{\frac{1}{N}} - 1)$ for the rate-monotonic scheduling algorithm in Liu and Layland (1973) for uniprocessor systems guarantees that a task set with N tasks can be feasibly scheduled using rate-monotonic scheduling if the total utilization is up to $N(2^{\frac{1}{N}} - 1)$. This also implies that rate-monotonic scheduling has a resource augmentation factor of $\frac{1}{N(2^{\frac{1}{N}}-1)}$.

Moreover, Baruah (2011) also proposed 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, assuming $\frac{1}{\epsilon}$ is a constant, for $\epsilon > 0$. However, the approach in Baruah (2011) only works if the three parameters $\frac{C_{\max}}{C_{\min}}$, $\frac{D_{\max}}{D_{\min}}$, $\frac{T_{\max}}{T_{\min}}$ are upper-bounded by a constant, where $C_{\max}$ and $C_{\min}$ are the maximum and minimum worst-case execution times, $D_{\max}$ and $D_{\min}$ are the maximum and minimum relative deadlines, and $T_{\max}$ and $T_{\min}$ are the maximum and minimum inter-arrival times. Therefore, for a task with a small workload, i.e., small C_i, or a very large T_i, the resulting $C_{\min}$ is very small or $T_{\max}$ is very large, and, hence, the approach in Baruah (2011) cannot lead to polynomial-time approximation schemes any more.

Our contributions This paper explores resource augmentation (i) for uniprocessor schedulability test by adopting the approximate demand bound function and (ii) for multiprocessor partitioned scheduling, and advances the state-of-the-art as follows:

- For uniprocessor systems, we present a tighter analysis of the resource augmentation factor by considering the structure of the approximate demand bound function directly, and prove that the resource augmentation factor is at most $\frac{2e-1}{e} \approx 1.6322$ for the approximate demand bound function (by taking a linear approximation when t is larger than the relative deadline D_i), where e is the Euler number. We also show that the (worst-case) resource augmentation factor when using the approximate demand bound function is at least 1.5 asymptotically.
- We show that deadline-monotonic partitioning with the approximate demand bound function, indeed, has a tighter resource augmentation factor than the factors provided in Baruah and Fisher (2005, 2006). This is again achieved by considering the structure of the approximate demand bound function directly and exploring the utilization violations (which were ignored in Baruah and Fisher 2005) for analyzing the resource augmentation factors. This is described in more detail later in the paper. We show that the resource augmentation factor is $\frac{3e-1}{e} - \frac{1}{M} \approx 2.6322 - \frac{1}{M}$ for task sets with constrained deadlines and $3 - \frac{1}{M}$ for task sets with arbitrary deadlines. Moreover, we provide input instances to show that the analytical lower bound of the resource augmentation factor for the above approach (deadline-monotonic partitioning with the approximate demand bound function) on identical multiprocessor systems with a large number of processors under non-specified fitting preference is 2.5.
- We also provide a proper reduction from the partitioned scheduling problem to the vector scheduling problem, studied in Chekuri and Khanna (2004). Such a scheme allows a $(1 + \epsilon)$ resource augmentation factor for speeding up, for $\epsilon > 0$. The scheme has polynomial-time complexity 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 Baruah (2011). Therefore, it can be applicable to more general task sets.

The rest of this paper is organized as follows. Section 2 specifies the task and platform model and discusses preliminary results. Section 3 focuses on the special case of uniprocessor systems, as the basis of the analysis. Section 4 shows the resource augmentation analysis for multiprocessor systems. Section 5 presents our polynomial-time approximation scheme to have a $(1+\epsilon)$ resource augmentation factor. Section 6 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, reviews the deadline-monotonic partitioning algorithm proposed in Baruah and Fisher (2005, 2006) for completeness, 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. The execution platform has M identical processors, and a task has the same execution and timing property on all of these M processors. We denote the utilization of task τ_i as $u_i = \frac{C_i}{T_i}$, and the density of task τ_i as $\delta_i = \frac{C_i}{\min\{T_i, D_i\}}$.

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 $i < j$. Such an ordering is called *deadline-monotonic ordering*. As already mentioned, a task set $\mathbf{T}$ is called an *implicit-deadline task set* if the relative deadline D_i of each task τ_i in $\mathbf{T}$ is equal to its minimum inter-arrival time T_i. A task set $\mathbf{T}$ is called a *constrained-deadline task set* if the relative deadline D_i of each task τ_i in $\mathbf{T}$ is no more than its minimum inter-arrival time T_i. A task set $\mathbf{T}$ is called an *arbitrary-deadline task set* if there exists a task τ_i in $\mathbf{T}$ with relative deadline D_i larger than its minimum inter-arrival time T_i.

2.2 Demand bound function

Based on the demand bound function defined in (1), it has been shown in Baruah et al. (1990) that a task set $\mathbf{T}$ can be feasibly scheduled under EDF on one processor if and only if

$$\forall t \geq 0, \quad \sum_{\tau_i \in \mathbf{T}} \mathrm{DBF}(\tau_i, t) \leq t. \tag{2}$$

However, as it requires to consider all $t \geq 0$ for evaluating the schedulability condition in (2), in the worst case, we may have to evaluate at least a pseudo-polynomial number of t for deciding whether a task set can be feasibly scheduled under EDF.

Based on the demand bound function, we can also define the *density of a task set* $\mathbf{T}$ as $\max_{t \geq 0} \frac{\sum_{\tau_i \in \mathbf{T}} \mathrm{DBF}(\tau_i, t)}{Mt}$. It is clear that for allowing a feasible partitioned schedule, the density of a task set $\mathbf{T}$ must be at most 1.

To obtain a polynomial-time schedulability test, Albers and Slomka (2004) proposed a linear approximation, defined as the *approximate demand bound function* in this paper, as follows:

$$\mathrm{DBF}^*(\tau_i, t) = \begin{cases} 0 & \text{if } t < D_i \\ (\frac{t - D_i}{T_i} + 1)C_i & \text{otherwise.} \end{cases} \tag{3}$$

Note that the approach in Albers and Slomka (2004) allows us to choose the number of discrete points before starting the linear approximation. However, as we will use this for task assignment with deadline-monotonic partitioning, we will only use the special case of linear approximation starting from the relative deadline of the task τ_i. Based on the definition in (3), the following inequality holds:

$$\forall t \geq 0, \quad \mathrm{DBF}(\tau_i, t) \leq \mathrm{DBF}^*(\tau_i, t) \leq 2\mathrm{DBF}(\tau_i, t). \tag{4}$$

The above inequality for a factor of 2 is actually tight. For example, when $t = T_i + D_i - \epsilon$ with a small and positive ϵ, $\mathrm{DBF}^*(\tau_i, T_i + D_i - \epsilon) \approx 2\mathrm{DBF}(\tau_i, T_i + D_i - \epsilon)$.

2.3 Multiprocessor partitioned scheduling

Given task set $\mathbf{T}$ and M identical processors, the objective of the *multiprocessor partitioned scheduling* problem 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,
- $\bigcup_{i=1}^{M} \mathbf{T}_i$ 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.

2.4 Resource augmentation

As the schedulability analysis based on testing the demand bound function in (2) is *co*$\mathcal{NP}$-hard (Eisenbrand and Rothvoß 2010) and the multiprocessor partitioned scheduling is $\mathcal{NP}$-hard in the strong sense (Mok 1983), approximation is needed to have efficient algorithms, unless $\mathcal{P} = \mathcal{NP}$.

If an algorithm $\mathcal{A}$ for the studied problem has a *resource augmentation factor* of ρ, it guarantees that *the task partition derived from the algorithm $\mathcal{A}$ is always feasible on each processor by speeding it up to ρ times as fast as the speed in the original platform, if task set $\mathbf{T}$ is feasible on the original M identical processors*. In other words, by taking the negation of the above statement, we know that *if the algorithm $\mathcal{A}$ fails to feasibly partition the task set $\mathbf{T}$ on M identical processors, there is no feasible task partition when each processor runs $\frac{1}{\rho}$ times slower than the original platform speed.*

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 (Garey and Johnson 1979), 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. We will use this definition of PTAS with respect to resource augmentation in Sect. 5.

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. (2010) have shown that there is no polynomial-time algorithm to test the feasibility of a task set within a constant resource augmentation factor with respect to speeding up, unless $\mathcal{P} = \mathcal{NP}$.

2.5 Simpler cases by reducing to makespan

For task sets with constrained deadlines or arbitrary deadlines on multiprocessor systems, one safe approach is to transform a task to another task with an implicit deadline by setting both the relative deadline and the minimum inter-arrival time of the task to the minimum of these two parameters, and, then, the schedulability test and partitioning strategy for implicit-deadline task sets can be applied.

Suppose that $\max_{\tau_i \in \mathbf{T}} \frac{T_i}{\min\{T_i, D_i\}}$ is ψ. We can transform the input task set $\mathbf{T}$ to $\mathbf{T}^\flat$ by constructing a task $\tau_i^\flat$ for each task $\tau_i \in \mathbf{T}$ as follows:

- Both $T_i^\flat$ and $D_i^\flat$ are set to $\min\{T_i, D_i\}$, and
- $C_i^\flat$ is set to C_i.

As we only make the task set more stringent in terms of meeting its timing constraints, it is not difficult to see that if a task partition of $\mathbf{T}^\flat$ is feasible, the corresponding task partition with respect to $\mathbf{T}$ is also feasible. Moreover, since $\mathbf{T}^\flat$ is a task set with implicit deadlines, we can now adopt the known polynomial-time algorithms with $\frac{4}{3} - \frac{1}{3M}$-approximation factor (Graham 1969) or the polynomial-time approximation scheme (Hochbaum and Shmoys 1987) with $(1+\epsilon)$-approximation factor for any $\epsilon > 0$. Since the construction of $\mathbf{T}^\flat$ takes linear time, the entire procedure takes polynomial time.

Similar to the argument in Sect. 2.4, when the adopted Makespan algorithm returns a task partition, then that can be feasibly scheduled by EDF. The main issue here is also to answer the speed-up factor to ensure the timing guarantees.

Suppose that $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$ is a *feasible* task partition that can be feasibly scheduled by EDF. Clearly, we know that $\sum_{\tau_i \in \mathbf{T}_m} \frac{C_i}{T_i} \leq 1$ for all $m = 1, 2, \ldots, M$. Therefore, according to the construction of $\mathbf{T}^\flat$, we also know that the corresponding task partition $\mathbf{T}_1^\flat, \mathbf{T}_2^\flat, \ldots, \mathbf{T}_M^\flat$ of $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$ has $\sum_{\tau_i^\flat \in \mathbf{T}_m^\flat} \frac{C_i^\flat}{T_i^\flat} \leq \psi$ for all $m = 1, 2, \ldots, M$.

Suppose that $\mathbf{T}_{1,\rho}^\flat, \mathbf{T}_{2,\rho}^\flat, \ldots, \mathbf{T}_{M,\rho}^\flat$ is a task partition derived by the Makespan algorithm with a ρ-approximation factor. Therefore, we know that

$$\max_{m=1,2,\ldots,M} \left\{ \sum_{\tau_i^\flat \in \mathbf{T}_{m,\rho}^\flat} \frac{C_i^\flat}{T_i^\flat} \right\} \leq \rho \max_{m=1,2,\ldots,M} \left\{ \sum_{\tau_i^\flat \in \mathbf{T}_m^\flat} \frac{C_i^\flat}{T_i^\flat} \right\} \leq \rho\psi. \tag{5}$$

As a result, we have the following theorem.

Theorem 1 *Suppose that* $\max_{\tau_i \in \mathbf{T}} \frac{T_i}{\min\{T_i, D_i\}}$ *is* ψ. *By transforming the task set* $\mathbf{T}$ *to an implicit-deadline task set* $\mathbf{T}^\flat$ *and adopting a polynomial-time* ρ*-approximation algorithm of the Makespan problem on* $\mathbf{T}^\flat$, *the resource augmentation factor is* $\rho\psi$.

Proof This is based on (5). □

The following corollary shows that the problem is actually simpler when the relative deadline is no less than the minimum inter-arrival time for every task in $\mathbf{T}$.

Corollary 1 *Suppose that* $D_i \geq T_i$ *for all tasks* τ_i *in* $\mathbf{T}$. *By transforming the task set* $\mathbf{T}$ *to an implicit-deadline task set* $\mathbf{T}^\flat$ *and adopting a polynomial-time* ρ*-approximation algorithm of the Makespan problem on* $\mathbf{T}^\flat$, *the resource augmentation factor is* ρ.

Unfortunately, for general cases, the above approach may lead to a very bad task partition, as the task transformation introduces too much pessimism. For example, when $T_i \gg D_i$, the above approach will have to pessimistically set T_i to D_i. However,

Algorithm 1 Deadline-Monotonic Partitioning

Input: set $\mathbf{T}$ of N tasks, M identical processors;
1: order tasks for the deadline-monotonic ordering, i.e., $D_i \leq D_j$ for $i < j$;
2: $\mathbf{T}_m \leftarrow \emptyset, \forall m = 1, 2, \ldots, M$;
3: **for** $i = 1$ to N **do**
4: **if** there exists m such that both (6) and (7) hold **then**
5: choose $m \in \{1, 2, \ldots, M\}$ **by preference** such that both (6) and (7) hold;
6: assign τ_i to processor m with $\mathbf{T}_m \leftarrow \mathbf{T}_m \cup \{\tau_i\}$;
7: **else**
8: return "the task assignment failed";
9: **end if**
10: **end for**
11: return feasible task assignment $\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_M$;

the above approach may work well when the ratio of D_i to T_i for every τ_i is bounded by a very small constant.

2.6 Deadline-monotonic partitioning

Baruah and Fisher (2005, 2006) adopted the approximate demand bound function $\text{DBF}^*(\tau_i, t)$ and deadline-monotonic partitioning to assign tasks. For completeness, the algorithm in Baruah and Fisher (2005, 2006) is presented in Algorithm 1. The basic idea is to consider tasks from the shortest relative deadline to the longest relative deadline for assignment. When a task τ_i is considered, a processor m with

$$C_i + \sum_{\tau_j \in \mathbf{T}_m} \text{DBF}^*(\tau_j, D_i) \leq D_i \tag{6}$$

and

$$u_i + \sum_{\tau_j \in \mathbf{T}_m} u_j \leq 1 \tag{7}$$

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 for which both (6) and (7) holds, the procedure fails to find task assignment.

The algorithm in Baruah and Fisher (2005, 2006) uses *first-fit*, in which the preference in Step 5 in Algorithm 1 is to choose the minimum index m. However, the analysis in Baruah and Fisher (2005, 2006) holds for any other preference too. For implicit-deadline task sets, it has been shown that the *worst-fit strategy based on utilization* by choosing the minimum $\sum_{\tau_j \in \mathbf{T}_m} u_j$ is a good strategy to minimize the maximum utilization (Graham 1969).

This paper does not restrict the analysis to any particular preference on choosing the index m in a deadline-monotonic partitioning. The analysis for the upper bound on resource augmentation is independent of the adopted fitting strategy. The fitting

strategy matters when considering the analytical lower bound of the resource augmentation factor for the deadline-monotonic partitioning in Sect. 4.3.

If Algorithm 1 returns a feasible task assignment, all the tasks are guaranteed to meet the timing constraints since the DBF^* function over-approximates the DBF function. Note that, if DBF^* is replaced by DBF in (6) when Algorithm 1 is applied, the resulting task partition may not be feasible even though $C_i + \sum_{\tau_j \in \mathbf{T}_m} \text{DBF}(\tau_j, D_i) \leq D_i$ holds.

The main issue here is to answer what can be guaranteed when Algorithm 1 returns failure in task partitioning. The *resource augmentation* technique helps quantify such failure, and provides the worst-case guarantees. Baruah and Fisher (2005, 2006) prove that Algorithm 1 has a $4 - \frac{2}{M}$ (respectively, $3 - \frac{1}{M}$) resource augmentation factor for arbitrary-deadline (constrained-deadline, respectively) task sets.

Even though a factor 2 in (4) is tight for bounding DBF and DBF^*, it is not tight for resource augmentation even for a uniprocessor system. Let us consider the following simple example on a uniprocessor system. Among the N tasks with a small and positive number ϵ, there are $N-1$ tasks with relative deadline equal to $T-\epsilon$, minimum inter-arrival time T, and execution time $\frac{T-\epsilon}{N-1}$. Task τ_N is with $D_N = 2T - 2\epsilon$, $C_N = \epsilon$, and $T_N = \infty$. It is not difficult to see that $\sum_{\tau_i \in \mathbf{T}} \text{DBF}^*(\tau_i, 2T - 2\epsilon) = 2T - 2\epsilon + \epsilon^2/T > 2T - 2\epsilon$, which leads to the failure of the schedulability test by using the approximate demand bound function. It is clear that $\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, 2T - 2\epsilon)$ is T. Therefore, for this case, $\sum_{\tau_i \in \mathbf{T}} \text{DBF}^*(\tau_i, 2T - 2\epsilon) \approx 2 \sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, 2T - 2\epsilon)$. However, since $\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, T - \epsilon) = (N-1)\frac{T-\epsilon}{N-1} = T - \epsilon$, any slowing down of the system is actually not feasible. That is, even though $\frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}^*(\tau_i, 2T-2\epsilon)}{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, 2T-2\epsilon)} \leq 2$, if we consider t as $T - \epsilon$, the task set indeed requests a high demand requirement as well. In other words, it is pessimistic to use the approximation factor of the approximate demand bound function to bound the resource augmentation factor. Therefore, by the above example, using (4) for analyzing the resource augmentation factor is correct, but not tight. This also explains why the analysis in Baruah and Fisher (2005, 2006) is not tight for multiprocessor systems.

As the necessary condition for being feasibly schedulable by EDF is to have $\frac{\sum_{\tau_i \in \mathbf{T}} u_i}{M} \leq 1$, $\max_{\tau_i \in \mathbf{T}} \delta_i \leq 1$, and $\frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, t)}{M} \leq t$ for all $t \geq 0$, the following lemma gives a constraint on the slowing down factors.

Lemma 1 *A task set is not feasible (not schedulable by EDF) by running at any speed factor s of the original platform speed on each processor if*

$$\max\left\{\max_{t \geq 0} \frac{\sum_{\tau_i \in \mathbf{T}} \text{DBF}(\tau_i, t)}{Mt}, \frac{\sum_{\tau_i \in \mathbf{T}} u_i}{M}, \max_{\tau_i \in \mathbf{T}} \delta_i\right\} > s. \tag{8}$$

For any s that satisfies (8), the resource augmentation factor is at most $\frac{1}{s}$ when Algorithm 1 fails to derive feasible task partitioning.

Proof The first part comes from the necessary condition for EDF scheduling. The second part follows directly from Lemma 2 in Baruah and Fisher (2005) (a negation

statement is used in this lemma) and the definition of the resource augmentation factor. □

3 Analysis for uniprocessors

This section presents the analysis for uniprocessor systems, in which $M = 1$, when the approximate demand bound function in (3) is used for schedulability analysis. Note that this is used as the basis for analyzing multiprocessor systems. The main reason why the analysis in Baruah and Fisher (2005, 2006) is not tight is because the inequalities in (4) are adopted for analyzing resource augmentation. We provide a tighter analysis in this section.

For uniprocessor systems, we consider that the total utilization is no more than 100 %. Otherwise, the task set is not feasible by testing for total utilization. Suppose that τ_ℓ is the first task with $\sum_{i=1}^{\ell} \mathrm{DBF}^*(\tau_i, D_\ell) > D_\ell$. Note that we already sort the tasks according to their relative deadlines in a non-decreasing order (i.e., deadline-monotonic ordering). We are going to show that

$$\begin{aligned} &\max\left\{\sum_{i=1}^{\ell}\frac{C_i}{T_i}, \max_{t\geq 0}\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i,t)}{t}, \max_{i=1}^{\ell}\delta_i\right\} \\ &\geq \max\left\{\sum_{i=1}^{\ell-1}\frac{C_i}{T_i}, \max_{t\geq 0}\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i,t)}{t}\right\} > \frac{e}{2e-1}, \end{aligned} \tag{9}$$

where e is the Euler number and $\geq$ is because some terms are dropped and reduced in max{}.

The proof strategy, illustrated in Fig. 1, consists of the following steps:

1. We perform task transformation to construct another task set $\mathbf{T}'$ such that the analysis on the transformed task set $\mathbf{T}'$ is equivalent to that on the original task set $\mathbf{T}$. The main concept is to maintain the approximate demand bound function when $t \geq D_\ell$ after the task transformation. This is presented in Sect. 3.1.
2. Based on the transformed task set $\mathbf{T}'$, we define variables $\beta, k, k^\dagger$. This is presented in Sect. 3.2.
3. Sect. 3.3 shows the analysis based on these variables to prove that (9) holds

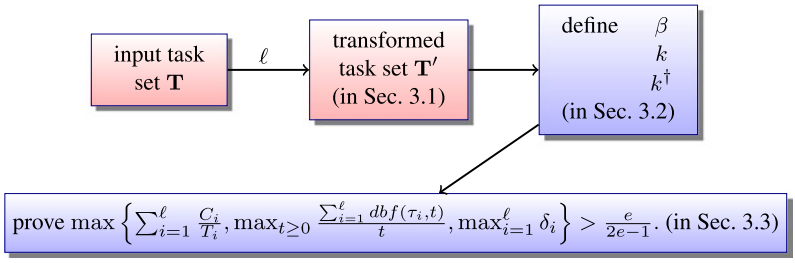


Fig. 1 The overall proof strategy for uniprocessor systems

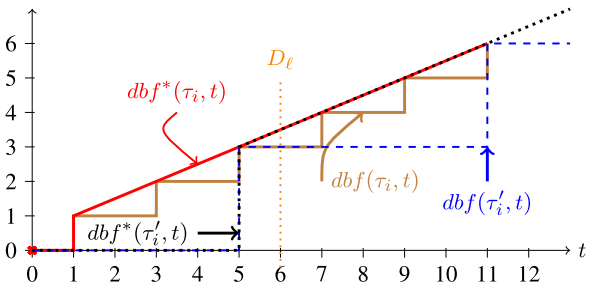


Fig. 2 An example of task transformation: $D_\ell = 6$, $C_i = 1, T_i = 2, D_i = 1$. Therefore, $C'_i = 3$, $T'_i = 6, D'_i = 5$

In Sect. 3.4, we will also present the analytical lower bound when the approximate demand bound function is used for schedulability analysis.

3.1 Task transformation

We construct another task set $\mathbf{T}' = \{\tau'_1, \tau'_2, \ldots, \tau'_\ell\}$, in which for τ'_i with $i = 1, 2, \ldots, \ell$,

$$C'_i = \left(\left\lfloor \frac{D_\ell - D_i}{T_i} \right\rfloor + 1 \right) \times C_i$$

$$T'_i = \left(\left\lfloor \frac{D_\ell - D_i}{T_i} \right\rfloor + 1 \right) \times T_i$$

$$D'_i = \left\lfloor \frac{D_\ell - D_i}{T_i} \right\rfloor \times T_i + D_i$$

Please note that $D'_\ell = D_\ell$, $T'_\ell = T_\ell$, and $C'_\ell = C_\ell$. That is, by releasing all the tasks synchronously (at time 0) and issuing subsequent jobs as early as possible whenever it is valid, we count the number χ_i of jobs with absolute deadlines less than or equal to D_ℓ. Therefore, $\chi_i = \lfloor \frac{D_\ell - D_i}{T_i} \rfloor + 1$. The relative deadline is changed from D_i to $D'_i = (\chi_i - 1)T_i + D_i$ which is the latest absolute deadline no later than D_ℓ, the new minimum inter-arrival time T'_i is $\chi_i T_i$, and the execution time C'_i is $\chi_i C_i$.

Figure 2 gives an example for this case. Originally, $C_i = 1, T_i = 2, D_i = 1$. When $D_\ell = 6$, we have $C'_i = 3, T'_i = 6, D'_i = 5$. The above task set $\mathbf{T}'$ enforces the following properties, which can also be easily observed in Fig. 2:

Lemma 2 *For any task τ'_i in task set $\mathbf{T}'$ and its original (corresponding) task τ_i in task set $\mathbf{T}$,*

- $\mathrm{DBF}(\tau'_i, t) \leq \mathrm{DBF}(\tau_i, t)$ *for any* $t \geq 0$.
- $\mathrm{DBF}^*(\tau'_i, t) = \mathrm{DBF}^*(\tau_i, t)$ *for any* $t \geq D'_i$.
- $\frac{C'_i}{T'_i} = \frac{C_i}{T_i}$.
- $D'_i + T'_i > D'_\ell, \forall i < \ell$.

Proof These follow from the definition of $\mathbf{T}'$. □

Lemma 3

$$\max_{t\geq 0}\frac{\sum_{i=1}^{\ell-1}\mathrm{DBF}(\tau_i,t)}{t}\geq\max_{t\geq 0}\frac{\sum_{i=1}^{\ell-1}\mathrm{DBF}(\tau_i',t)}{t} \tag{10}$$

Proof This follows from $\mathrm{DBF}(\tau_i',t)\leq\mathrm{DBF}(\tau_i,t)$ for any $t\geq 0$. □

3.2 Terminologies for analyzing DBF*

From the failure of the schedulability test with the approximate demand bound function at time D_ℓ and $\mathrm{DBF}^*(\tau_i',t)=\mathrm{DBF}^*(\tau_i,t)$ for any $t\geq D_i'$, we have

$$\sum_{i=1}^{\ell}\mathrm{DBF}^*\big(\tau_i',D_\ell\big)=\sum_{i=1}^{\ell}\left(\frac{D_\ell'-D_i'}{T_i'}+1\right)C_i'>D_\ell=D_\ell'. \tag{11}$$

For notational simplicity, we define β, $k^\dagger$, and k as follows:

$$C_\ell'\stackrel{\mathrm{def}}{=}\beta\sum_{i=1}^{\ell-1}C_i', \tag{12}$$

$$\sum_{i=1}^{\ell-1}\left(\frac{D_\ell'-D_i'}{T_i'}\right)C_i'\stackrel{\mathrm{def}}{=}k^\dagger\sum_{i=1}^{\ell-1}C_i', \tag{13}$$

$$k\sum_{i=1}^{\ell-1}C_i'\stackrel{\mathrm{def}}{=}D_\ell'-(1+\beta)\sum_{i=1}^{\ell-1}C_i'. \tag{14}$$

That is, by taking $\sum_{i=1}^{\ell-1}C_i'$ as the basis, β defines the ratio of C_ℓ' to $\sum_{i=1}^{\ell-1}C_i'$, $k^\dagger$ defines the ratio of the partially released workload $\sum_{i=1}^{\ell-1}(\frac{D_\ell'-D_i'}{T_i'})C_i'$ in time interval length D_ℓ' to $\sum_{i=1}^{\ell-1}C_i'$, and k is defined to set $D_\ell'=(1+k+\beta)\sum_{i=1}^{\ell-1}C_i'$.

For example, if $\ell=2$ and $C_\ell=2.6$ for the example in Fig. 2, then $\sum_{i=1}^{\ell-1}C_i'=3$, $\beta=\frac{2.6}{3}$, $k^\dagger=\frac{0.5}{3}$, and $k=\frac{0.4}{3}$.

By Lemma 2, we also know that, for any $i<\ell$, $\frac{D_\ell'-D_i'}{T_i'}<1$. Therefore, we have

$$k\sum_{i=1}^{\ell-1}C_i'<k^\dagger\sum_{i=1}^{\ell-1}C_i'<\sum_{i=1}^{\ell-1}C_i'\Rightarrow k<1.$$

Moreover, we have

$$\max_{t\geq 0}\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i',t)}{t}\geq\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i',D_\ell')}{D_\ell'}=\frac{\sum_{i=1}^{\ell}C_i'}{D_\ell'}=\frac{1+\beta}{1+k+\beta}. \tag{15}$$

When k is small, (15) gives a good bound for estimating the density. Specifically, when $k=0$, the estimation is exact, i.e., the resource augmentation factor is 1. But, when k is close to 1 and β is small, the above analysis gives a resource augmentation factor close to 2, which is not tight as illustrated at the end of Sect. 2.4.

Table 1 Symbols used in the proofs in Sect. 3.3

Symbol	Description	Mathematical definition
β	ratio of C'_ℓ to $\sum_{i=1}^{\ell-1} C'_i$	$\frac{C'_\ell}{\sum_{i=1}^{\ell-1} C'_i}$
k	ratio of $D'_\ell - \sum_{i=1}^{\ell} C'_i$ to $\sum_{i=1}^{\ell-1} C'_i$	$\frac{D'_\ell - (1+\beta)\sum_{i=1}^{\ell-1} C'_i}{\sum_{i=1}^{\ell-1} C'_i}$
W	partial released workload	$\sum_{i=1}^{\ell-1} \frac{D_\ell - D'_i}{T'_i} C'_i$
$k^\dagger$	ratio of W to $\sum_{i=1}^{\ell-1} C'_i$	$\frac{W}{\sum_{i=1}^{\ell-1} C'_i}$
$D_i^\dagger$	new relative deadline for τ_i in Lemma 4	
$T_i^\sharp$	new period for τ_i in Lemma 4	
$T_i^\dagger$	new period for τ_i in Lemma 5	
α	auxiliary variable	$\alpha = \frac{\frac{e}{e-1}k}{1+k+\beta}$ from Lemma 6

3.3 Analysis for DBF*

As $0 \le k < 1$ depends on the input instance, to analyze the resource augmentation factor, we need to consider the worst case of k. Apparently, for providing tighter analysis, we should seek other tighter bounds instead of adopting the bound in (15) when k is large. Therefore, the analysis in this subsection is mainly to find another tighter bound when k is large.

For the ease of presentation, we reorder the tasks $\tau'_1, \tau'_2, \ldots, \tau'_{\ell-1}$ so that $D'_i \le D'_j$ if $i < j$. Note that task τ_ℓ is identical to task τ'_ℓ. Suppose that both $\max_{0 \le t \le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t}$ and $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}$ are less than or equal to α. Clearly, $\alpha > 0$; otherwise, the task set has no workload or τ_ℓ can not be feasibly scheduled itself.

Let W be the *partially released workload* of the tasks in time interval length D'_ℓ by the linear approximation after the relative deadline of tasks in $\mathbf{T}'$, in which

$$W \stackrel{\text{def}}{=} \sum_{i=1}^{\ell-1} \frac{D_\ell - D'_i}{T'_i} C'_i = k^\dagger \sum_{i=1}^{\ell-1} C'_i. \tag{16}$$

As shown in Sect. 3.2, when k is 0, the resource augmentation factor is 1. For the rest of this section, unless specified, we will implicitly focus on the case that $k > 0$ to find the resource augmentation factor. We are going to show, for contradiction, that W is no more than $k \sum_{i=1}^{\ell-1} C'_i$ when both the total utilization $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}$ and the maximum density $\max_{0 \le t \le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t}$ are small, i.e., both are less than or equal to α when α is set to $\frac{\frac{e}{e-1}k}{1+k+\beta}$.

Before we start to present the analysis, in Table 1 we summarize the symbols that are going to be widely used in this subsection. The proof strategy is illustrated in

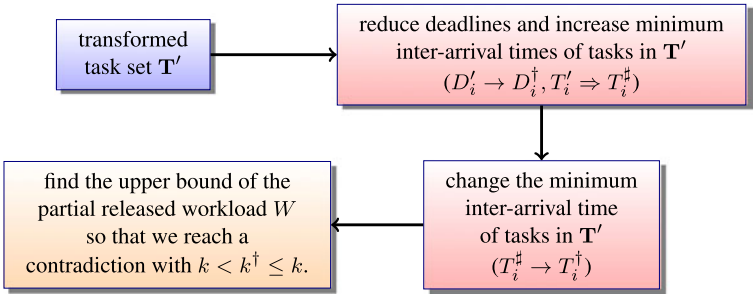


Fig. 3 The proof strategy for contradiction in Sect. 3.3

Fig. 3: We first reduce the relative deadlines from D_i' to $D_i^\dagger$ and increase the minimum inter-arrival time for $i = 1, 2, \ldots, \ell - 1$ to increase the value of W. Then, we will change the minimum inter-arrival time to $T_i^\dagger$ for $i = 1, 2, \ldots, \ell - 1$ to increase the value of W.

Lemma 4 *For any* $\alpha > 0$, *when* $\max_{0 \le t \le D_\ell'} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau_i', t)}{t} \le \alpha$, $\sum_{i=1}^{\ell-1} \frac{C_i'}{T_i'} \le \alpha$, *and* $D_i^\dagger = \frac{1}{\alpha} \cdot \sum_{j=1}^{i} C_j'$, $T_i^\sharp = T_i' + D_i' - D_i^\dagger$, *we have*

$$W \le \sum_{i=1}^{\ell-1} \frac{D_\ell - D_i^\dagger}{T_i^\sharp} C_i' \quad \textit{and} \quad \alpha \ge \sum_{i=1}^{\ell-1} \frac{C_i'}{T_i^\sharp}.$$

Proof Since we assume $\max_{0 \le t \le D_\ell'} \frac{\sum_{j=1}^{\ell-1} \mathrm{DBF}(\tau_j', t)}{t} \le \alpha$, and tasks are ordered non-decreasingly according to the relative deadlines ($D_i' \le D_j'$ if $i < j$), we know that $\sum_{j=1}^{i} C_j' \le \alpha D_i'$ and $\frac{C_i'}{T_i'}(D_\ell - D_i') \le \frac{C_i'}{T_i'}(D_\ell - \frac{1}{\alpha}\sum_{j=1}^{i} C_j') = \frac{C_i'}{T_i'}(D_\ell - D_i^\dagger)$. Therefore, we have $D_i^\dagger \le D_i'$. According to the definition $T_i^\sharp = T_i' + D_i' - D_i^\dagger$ in this lemma, we know that $T_i^\sharp \ge T_i'$ and $D_\ell < T_i^\sharp + D_i^\dagger = T_i' + D_i'$. By $T_i^\sharp \ge T_i'$ for $i = 1, 2, \ldots, \ell - 1$, we know that

$$\alpha \ge \sum_{i=1}^{\ell-1} \frac{C_i'}{T_i'} \ge \sum_{i=1}^{\ell-1} \frac{C_i'}{T_i^\sharp}.$$

By $D_\ell < T_i^\sharp + D_i^\dagger = T_i' + D_i'$ and $T_i^\sharp \ge T_i'$, for $i = 1, 2, \ldots, \ell - 1$, we have

$$C_i' \frac{D_\ell - D_i'}{T_i'} = C_i' \left(1 - \frac{T_i' + D_i' - D_\ell}{T_i'}\right) = C_i' \left(1 - \frac{T_i^\sharp + D_i^\dagger - D_\ell}{T_i'}\right)$$
$$\le C_i' \left(1 - \frac{T_i^\sharp + D_i^\dagger - D_\ell}{T_i^\sharp}\right) = \frac{C_i'}{T_i^\sharp}(D_\ell - D_i^\dagger).$$

Table 2 An example of task transformation

	C'_i	D'_i	T'_i	$D_i^{\dagger}$	$T_i^{\sharp}$	$T_i^{\dagger}$
$i=1$	1	2	6	$\frac{1}{0.6327} \approx 1.5806$	≈ 6.4194	≈ 4.4194
$i=2$	1	4	6	$\frac{2}{0.6327} \approx 3.1611$	≈ 6.8389	≈ 2.8389
$i=3$	0.5	5	6	$\frac{2.5}{0.6327} \approx 3.9514$	≈ 7.0486	≈ 13.1087
$i=4$	0.7	5.5	6	$\frac{3.2}{0.6327} \approx 5.0577$	≈ 6.4423	∞
$i=5$	0.5	5.9	6	$\frac{3.7}{0.6327} \approx 5.8480$	≈ 6.0520	∞

Therefore, by the above arguments, we know that the workload

$$W = \sum_{i=1}^{\ell-1} \frac{D_\ell - D'_i}{T'_i} C'_i \le \sum_{i=1}^{\ell-1} \frac{D_\ell - D_i^{\dagger}}{T_i^{\sharp}} C'_i \quad \text{and} \quad \alpha \ge \sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} \ge \sum_{i=1}^{\ell-1} \frac{C'_i}{T_i^{\sharp}}. \qquad \square$$

We use the example presented in Table 2 to illustrate the transformations in the proof presented in this subsection. Suppose that $\alpha = 0.6327$ and $D'_\ell = 6$. Then, we know that $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} = \frac{3.7}{6} \le \alpha$ and $\max_{0 \le t \le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t} = \frac{3.7}{5.9} \le \alpha$, and $W = 1.14\bar{9}$. By the transformation in Lemma 4, we have $\sum_{i=1}^{\ell-1} \frac{C'_i}{T_i^{\sharp}} < 0.6327$ and $W < \sum_{i=1}^{\ell-1} \frac{D_\ell - D_i^{\dagger}}{T_i^{\sharp}} C'_i \approx 1.3638$. Moreover, the transformation from Fig. 4(a) to Fig. 4(b) provides an illustrative example of the transformation described above.

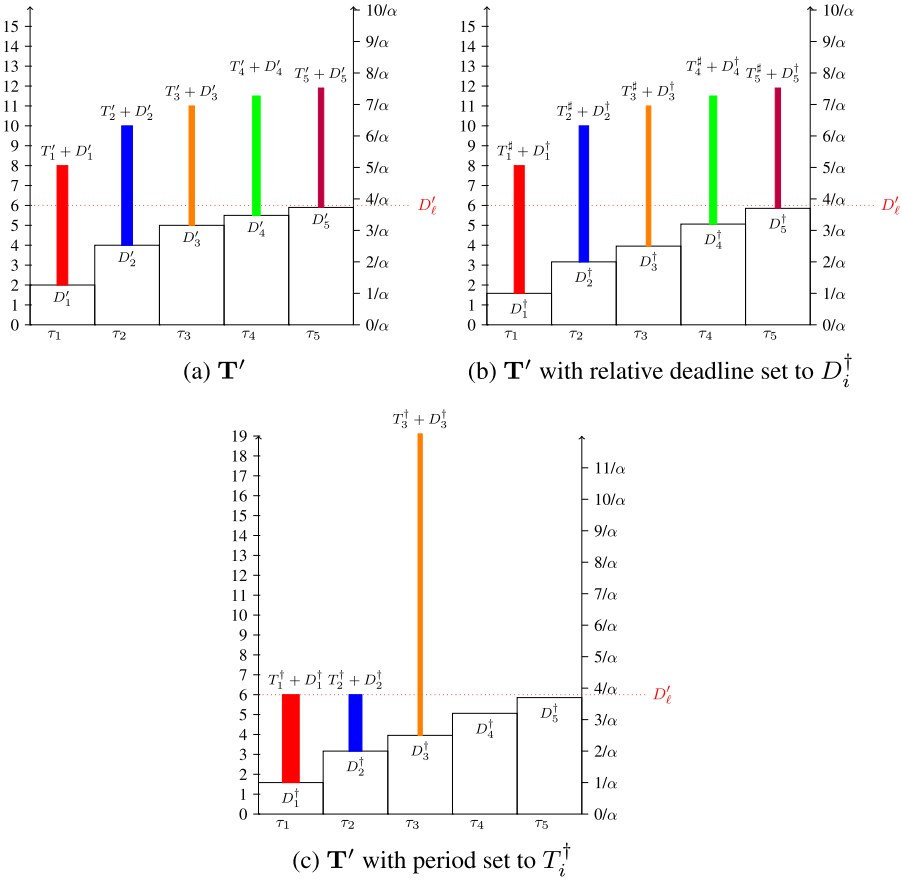
For the rest of this subsection, we replace the value of the variable T'_i by $T_i^{\sharp}$ as defined in Lemma 4. Therefore, $D_i^{\dagger} + T'_i > D_\ell, \forall i = 1, \ldots, \ell - 1$ and $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} \le \alpha$ both hold.

Now, we do a further transformation after Lemma 4. *The key observation is to change the minimum inter-arrival time of the tasks to maximize the partially released workload without increasing the total utilization.* The following lemma shows that W is upper-bounded by the partially released workload in the time interval length D_ℓ of the following (transformed) task set: (1) If $T_i^{\dagger}$ is $D_\ell - D_i^{\dagger}$, $T_j^{\dagger}$ is $D_\ell - D_j^{\dagger}$ if $j < i$. (2) If $T_i^{\dagger}$ is larger than $D_\ell - D_i^{\dagger}$, $T_j^{\dagger}$ is ∞ if $j > i$.

Lemma 5 *Let i^* be the index such that*

$$\sum_{i=1}^{i^*-1} \frac{C'_i}{D_\ell - D_i^{\dagger}} < \sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} \le \sum_{i=1}^{i^*} \frac{C'_i}{D_\ell - D_i^{\dagger}},$$

where $\max_{0 \le t \le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t} \le \alpha$, $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} \le \alpha$, *and* $\alpha > 0$. *Moreover, for every task* τ'_i, $D_i^{\dagger} = \frac{1}{\alpha} \cdot \sum_{j=1}^{i} C'_j$, *and* $T'_i + D_i^{\dagger} > D'_\ell$. *Let*



(a) $\mathbf{T}'$ (b) $\mathbf{T}'$ with relative deadline set to $D_i^\dagger$

(c) $\mathbf{T}'$ with period set to $T_i^\dagger$

Fig. 4 An example by task transformation in Table 2, where the partial workload is represented by the (*colored*) area below D'_ℓ

- $T_j^\dagger = D'_\ell - D_j^\dagger$ *for* $j < i^*$;
- $T_j^\dagger = \dfrac{C'_{i^*}}{\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} - \sum_{i=1}^{i^*-1} \frac{C'_i}{D_\ell - D_i^\dagger}}$ *for* $j = i^*$;
- $T_j^\dagger = \infty$ *for* $j > i^*$.

Then, we have $\sum_{i=1}^{i^*} \frac{C'_i}{T_i^\dagger} \le \alpha$ *and*

$$W \le \left(\sum_{i=1}^{i^*-1} C'_i \right) + \frac{C'_{i^*}}{T_{i^*}^\dagger} (D_\ell - D_{i^*}^\dagger). \tag{17}$$

Proof Suppose that there are two tasks τ'_i and τ'_j with the following properties: $T'_i > D_\ell - D_i^\dagger$, $T'_j > D_\ell - D_j^\dagger$, and $i < j$. Let σ be $\min\{\frac{C'_j}{T'_j}, \frac{C'_i}{T'_i} - \frac{C'_i}{D_\ell - D_i^\dagger}\}$, in which σ is

either the utilization of task τ'_j or the maximal increased utilization by changing the minimum inter-arrival time of τ'_i to $D_\ell - D_i^\dagger$. Since $D_i^\dagger < D_j^\dagger$ for $i < j$, we know that

$$\frac{C'_i}{T'_i}\left(D_\ell - D_i^\dagger\right) + \frac{C'_j}{T'_j}\left(D_\ell - D_j^\dagger\right) < \left(\frac{C'_i}{T'_i} + \sigma\right)\left(D_\ell - D_i^\dagger\right) + \left(\frac{C'_j}{T'_j} - \sigma\right)\left(D_\ell - D_j^\dagger\right). \tag{18}$$

It is not difficult to see the existence of index i^*. According to the definition of i^*, we know that assigning each task τ'_i from τ'_1 to τ'_{i^*} with the minimum inter-arrival time equal to $D_\ell - D_i^\dagger$ will result in total utilization no less than the total utilization $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}$. Therefore, the inequality $\sum_{i=1}^{i^*} \frac{C'_i}{T_i^\dagger} \le \alpha$ holds by the definition of $T_j^\dagger$, while the inequality in (17) holds due to (18). □

We again use the example in Table 2 to illustrate the transformations in Lemma 5. The index i^* in Lemma 5 for this example is 3. By changing the periods to $T^\dagger$, as shown in Table 2, we also know that $(\sum_{i=1}^{i^*-1} C'_i) + \frac{C'_{i^*}}{T_{i^*}^\dagger}(D_\ell - D_{i^*}^\dagger) \approx 2.0781$, which is larger than W (which is 1.3638 after the transformation in Lemma 4) for this example. The transformation from Fig. 4(b) to Fig. 4(c) provides an illustrative example for the above transformation in Lemma 5. The following lemma is needed as a property for proving the upper bound of the right-hand side of (17).

Lemma 6 *Suppose* $\alpha = \frac{\frac{e}{e-1}k}{1+k+\beta} > 0$. *Let* $U(t)$ *be* $\frac{1}{1+k+\beta-\frac{t}{\alpha}}$ *for* $0 \le t \le 1$. *Then,* $\int_0^k U(t)dt$ *is equal to* α.

Proof This is from the following integration result and the definition of $\alpha = \frac{\frac{e}{e-1}k}{1+k+\beta}$, i.e.,

$$\int_0^k \frac{1}{1+k+\beta-\frac{t}{\alpha}}dt = \alpha \ln \frac{1+\beta+k}{1+\beta+k-\frac{k}{\alpha}} = \alpha.$$ □

We can now prove the upper bound of the right-hand side of (17), by considering the corresponding problem in the continuous space in the following lemma.

Lemma 7 *Suppose* $\alpha = \frac{\frac{e}{e-1}k}{1+k+\beta} > 0$. *With* $\max_{0\le t\le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i,t)}{t} \le \alpha$ *and* $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i} \le \alpha$, *we have*

$$W \le k\sum_{i=1}^{\ell-1} C'_i. \tag{19}$$

Proof Suppose t_i is $\frac{\sum_{j=1}^{i} C'_j}{\sum_{j=1}^{\ell-1} C'_j}$ for $i = 1, 2, \ldots, i^* - 1$ and let t_0 be 0. Therefore, C'_i is equal to $(t_i - t_{i-1})\sum_{j=1}^{\ell-1} C'_j$. This leads to $\sum_{i=1}^{i^*-1} C'_i = t_{i^*-1}\sum_{j=1}^{\ell-1} C'_j$. Moreover,

for $i < i^*$,

$$\frac{C'_i}{T_i^\dagger} = \frac{C'_i}{D_\ell - D_i^\dagger} = \frac{(t_i - t_{i-1})(\sum_{j=1}^{\ell-1} C'_j)}{(1+k+\beta-\frac{t_i}{\alpha})\sum_{j=1}^{\ell-1} C'_j} = \frac{t_i - t_{i-1}}{1+k+\beta-\frac{t_i}{\alpha}}. \tag{20}$$

For any $t \le t_i$ with $i < i^*$, we know that $\frac{1}{1+k+\beta-\frac{t_i}{\alpha}} \ge \frac{1}{1+k+\beta-\frac{t}{\alpha}} \ge 0$, which implies $\frac{t_i - t_{i-1}}{1+k+\beta-\frac{t_i}{\alpha}} \ge \int_{t_{i-1}}^{t_i} \frac{dt}{1+k+\beta-\frac{t}{\alpha}}$. Therefore, we have

$$\sum_{i=1}^{i^*-1} \frac{C'_i}{T_i^\dagger} \ge \int_0^{t_{i^*-1}} \frac{dt}{1+k+\beta-\frac{t}{\alpha}}.$$

Moreover, let t_{i^*} be $t_{i^*-1} + \frac{C'_{i^*}(D_\ell - D^\dagger_{i^*})}{T^\dagger_{i^*}\sum_{j=1}^{\ell-1} C'_j}$. By the definition of W, $T_i^\dagger$, $D_i^\dagger$, and (17) in Lemma 5, we know that $W \le \sum_{i=1}^{i^*} \frac{C'_i(D'_\ell - D_i^\dagger)}{T_i^\dagger} = t_{i^*}\sum_{j=1}^{\ell-1} C'_j$. Similarly, by the definition of D_ℓ and t_{i^*}, we have $\frac{C'_{i^*}}{T^\dagger_{i^*}}(D_\ell - D^\dagger_{i^*})$ equals to $(t_{i^*} - t_{i^*-1})\sum_{j=1}^{\ell-1} C'_j$ and

$$\frac{C'_{i^*}}{T^\dagger_{i^*}} \ge \int_{t_{i^*-1}}^{t_{i^*}} \frac{dt}{1+k+\beta-\frac{t}{\alpha}}.$$

Therefore, $\int_0^{t_{i^*}} \frac{dt}{1+k+\beta-\frac{t}{\alpha}} \le \sum_{j=1}^{i^*} \frac{C'_j}{T_j^\dagger} \le \alpha$.

By Lemma 6, we know that $\int_0^k \frac{dt}{1+k+\beta-\frac{t}{\alpha}} = \alpha$. Therefore, $k \ge t_{i^*} \ge \frac{W}{\sum_{j=1}^{\ell-1} C'_j}$, which results in $W \le k\sum_{i=1}^{\ell-1} C'_i$. □

By the above analysis, we conclude the contradiction for the assumption that both $\max_{0\le t\le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t}$ and $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}$ are less than or equal to $\alpha = \frac{\frac{e}{e-1}k}{1+k+\beta}$.

Lemma 8 *If* $k \ge \frac{e-1}{e}(1+\beta)$, *then*

$$\max\left\{\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}, \max_{0\le t\le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t}\right\} > \frac{\frac{e}{e-1}k}{1+k+\beta} \tag{21}$$

Proof If $\sum_{i=1}^{\ell-1} \frac{C'_i}{T'_i}$ and $\max_{0\le t\le D'_\ell} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau'_i, t)}{t}$ are less than or equal to $\frac{\frac{e}{e-1}k}{1+k+\beta}$, we reach the contradiction that

$$k\sum_{i=1}^{\ell-1} C'_i <_1 k^\dagger \sum_{i=1}^{\ell-1} C'_i =_2 W \le_3 k\sum_{i=1}^{\ell-1} C'_i, \tag{22}$$

where $<_1$ comes from the definition of k and $k^{\dagger}$, $=_2$ comes from the definition of W, $\leq_3$ is from (19). Therefore, either $\sum_{i=1}^{\ell-1} \frac{C_i'}{T_i'} > \alpha$ or $\max_{0 \leq t \leq D_\ell'} \frac{\sum_{i=1}^{\ell-1} \mathrm{DBF}(\tau_i', t)}{t} > \alpha$ must hold, which implies the inequality in (21). □

We can conclude this subsection with the following theorem.

Theorem 2 *The resource augmentation factor for Algorithm 1 is* $\frac{2e-1}{e}$, *which is about* 1.6322, *when* $M = 1$.

Proof Depending on the input-dependent value k, there are two cases:

- When $0 \leq k < \frac{e-1}{e}(1+\beta)$, this is an easier case, which has been analyzed in Sect. 3.2. By Lemmas 1 and 3 and (15), the inequality in (9) holds, and the resource augmentation factor is upper-bounded by $\frac{1+\frac{e-1}{e}(1+\beta)+\beta}{1+\beta} = \frac{2e-1}{e}$.
- When $\frac{e-1}{e}(1+\beta) \leq k < 1$, this is the case analyzed in Lemma 8. Therefore, by Lemmas 1, 2, 3, and 8, we also know that (9) holds, and the resource augmentation factor is at most $\frac{(1+\beta)\frac{2e-1}{e}}{1+\beta} = \frac{2e-1}{e}$.

Therefore, as $0 \leq k < 1$, we conclude that the resource augmentation factor for Algorithm 1 is $\frac{2e-1}{e}$, in which the worst case based on the above analysis happens when $k = \frac{e-1}{e}(1+\beta)$. □

3.4 Analytical lower bound for the approximate demand bound function

We now show the lower bound of the resource augmentation factors (in the worst cases) for Algorithm 1. The strategy here is to design task sets such that Algorithm 1 fails, but there exists a feasible schedule by slowing down the processors to $\frac{1}{\rho'}$ of the original speeds. Then, the resource augmentation factor is at least ρ'.

Theorem 3 *The resource augmentation factor of Algorithm 1 is at least* $\frac{3}{2}$ *when* $M = 1$.

Proof We prove this theorem by providing a concrete input instance. There are $N = \lambda + 1$ tasks in $\mathbf{T}$, in which for $i = 1, 2, \ldots, \lambda$ task τ_i has relative deadline D_i equal to $1.5i$, minimum inter-arrival time T_i equal to 1.5λ, execution time C_i equal to 1. Task τ_N is with $D_N = 1.5\lambda$ and $C_N = 0.5 + \epsilon$ and $T_N = \infty$ where $\epsilon > 0$ and close to 0.

Algorithm 1 fails when considering task τ_N since

$$C_N + \sum_{i=1}^{N-1} \mathrm{DBF}^*(\tau_i, D_N)$$

$$= 0.5 + \epsilon + \sum_{i=1}^{\lambda} \left(1 + \frac{(1.5\lambda - 1.5i)}{1.5\lambda}\right) = 1.5\lambda + \epsilon > D_N.$$

The utilization of the task set is

$$\sum_{i=1}^{N} \frac{C_i}{T_i} = \frac{\lambda}{1.5\lambda} + \frac{C_N}{\infty} = \frac{2}{3}.$$

Moreover, the maximum density is

$$\max_t \frac{\sum_{i=1}^{N} \mathrm{DBF}(\tau_i, t)}{t} = \max_{t \geq 1.5\lambda} \frac{\lambda + \lfloor \frac{t-1.5\lambda}{1.5} \rfloor + 0.5 + \epsilon}{1.5\lambda + (t - 1.5\lambda)} = \frac{2}{3} + \frac{0.5+\epsilon}{1.5\lambda}.$$

As a result, when $\lambda \to \infty, \epsilon \to 0$, if the system is slowed down to run at factor $\frac{2}{3} + \frac{0.5+\epsilon}{1.5\lambda} \to \frac{2}{3}$ of the original speed, the task set is feasible by using EDF. Therefore, the resource augmentation factor of Algorithm 1 is at least $\frac{3}{2}$ when $M = 1$. □

4 Analysis for deadline-monotonic partitioning on multiprocessors

This section presents the analysis for resource augmentation factors for identical multiprocessor systems. We will first present the analysis for arbitrary-deadline task sets and then prove the resource augmentation factor for constrained-deadline task sets.

Similar to Sect. 3, suppose that τ_ℓ could not be assigned on any of the M processors by the deadline-monotonic partitioning algorithm (Algorithm 1). We are going to find a feasible upper bound ρ for such an input instance, where

$$\max\left\{\max_t \frac{\sum_{\tau_i \in \mathbf{T}} \mathrm{DBF}(\tau_i, t)}{Mt}, \sum_{\tau_i \in \mathbf{T}} \frac{C_i}{MT_i}, \max_{\tau_i \in \mathbf{T}} \delta_i\right\} > \frac{1}{\rho}.$$

Therefore, the resource augmentation factor of the deadline-monotonic partitioning algorithm is at most ρ.

Similar to Sect. 3, we first transform tasks $\tau_1, \tau_2, \ldots, \tau_\ell$ to $\tau'_1, \tau'_2, \ldots, \tau'_\ell$ without increasing the demand bound function or the utilization of each of the tasks. Then, if τ_i is in $\mathbf{T}_m$, the corresponding transformed task τ'_i is assigned in $\mathbf{T}'_m$. For the rest of this section, our explanations will be based on the transformed tasks. No matter which fitting preference is used in Algorithm 1, the M processors are categorized into two disjoint sets $\mathbf{M}_1$ and $\mathbf{M}_2$, in which:

- processor m in $\mathbf{M}_1$ violates the constraint on the sum of the approximate demand bound functions at time D_ℓ, described in (6), i.e.,

$$C'_\ell + \sum_{\tau'_i \in \mathbf{T}'_m} \mathrm{DBF}^*\left(\tau'_i, D'_\ell\right) > D'_\ell. \tag{23}$$

Table 3 An example for task assignment when $M > 1$

	C'_i	D'_i	T'_i
$i = 1$	1	1	6
$i = 2$	1	2	2
$i = 3$	1.05	2.1	2

- processor m in $\mathbf{M}_2$ *only* violates the total utilization constraint when considering τ_ℓ, described in (7), i.e.,

$$\frac{C'_\ell}{T'_\ell} + \sum_{\tau'_i \in \mathbf{T}'_m} \frac{C'_i}{T'_i} > 1. \tag{24}$$

Consider the example in Table 3 with $M = 2$. According to Algorithm 1, task τ_1 will be assigned on processor 1 and task τ_2 will be assigned on processor 2. When considering task τ_3, the assignment to processor 1 is not feasible due to (23), since $\mathbf{T}_1$ is $\{\tau_1\}$, and the assignment to processor 2 is not feasible *only* due to (24), since $\mathbf{T}_2$ is $\{\tau_2\}$.

Note that by the above definition, a processor m is either in $\mathbf{M}_1$ or in $\mathbf{M}_2$. Hence, $|\mathbf{M}_1| + |\mathbf{M}_2|$ is equal to M, since τ_ℓ is not able to be assigned on any processor by the deadline-monotonic partitioning algorithm, where $|\mathbf{X}|$ is the cardinality of set $\mathbf{X}$.

For the analysis of multiprocessor systems in this section, we re-defined β, $k^\dagger$, and k as follows:

$$C'_\ell \stackrel{\text{def}}{=} \frac{\beta}{|\mathbf{M}_1|} \sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} C'_i, \tag{25}$$

$$\sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} \left(\frac{D'_\ell - D'_i}{T'_i} \right) C'_i \stackrel{\text{def}}{=} k^\dagger \sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} C'_i, \tag{26}$$

$$|\mathbf{M}_1| D'_\ell - (1 + \beta) \sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} C'_i \stackrel{\text{def}}{=} k \sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} C'_i. \tag{27}$$

4.1 Tasks with arbitrary deadlines

We first analyze the case when the test based on the approximate demand bound function fails. By (23), we know that $k^\dagger > k$ since

$$|\mathbf{M}_1| C'_\ell + \sum_{m \in \mathbf{M}_1} \sum_{\tau'_i \in \mathbf{T}'_m} \left(\frac{D'_\ell - D'_i}{T'_i} + 1 \right) C'_i > |\mathbf{M}_1| D'_\ell.$$

By definition, we know that the maximum density of the task set should be at least

$$\max_{t\geq 0}\frac{\sum_{\tau_i\in\mathbf{T}}\mathrm{DBF}(\tau_i,t)}{Mt}\geq\frac{\sum_{\tau_i'\in\mathbf{T}'}\mathrm{DBF}(\tau_i',D_\ell')}{MD_\ell'}\geq\frac{|\mathbf{M}_1|}{|\mathbf{M}_1|+|\mathbf{M}_2|}\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{1+k+\beta}. \tag{28}$$

Moreover, the tasks assigned to processors in $\mathbf{M}_1$ should also have a certain amount of utilization. Based on (26), we know

$$\sum_{m\in\mathbf{M}_1}\sum_{\tau_i'\in\mathbf{T}_m'}\frac{D_\ell'}{T_i'}C_i'\geq k^\dagger\sum_{m\in\mathbf{M}_1}\sum_{\tau_i'\in\mathbf{T}_m'}C_i', \tag{29}$$

which, along with $k^\dagger > k$, implies that the total utilization of these tasks should be lower-bounded as follows:

$$\sum_{m\in\mathbf{M}_1}\sum_{\tau_i'\in\mathbf{T}_m'}\frac{C_i'}{T_i'}>\frac{k\sum_{m\in\mathbf{M}_1}\sum_{\tau_i'\in\mathbf{T}_m'}C_i'}{D_\ell'}=\frac{|\mathbf{M}_1|k}{1+k+\beta}. \tag{30}$$

For the second case where the test only fails for the utilization constraint, by (24), we know that the total utilization of these tasks assigned to processors in $\mathbf{M}_2$ should be lower-bounded as follows:

$$|\mathbf{M}_2|\frac{C_\ell'}{T_\ell'}+\sum_{m\in\mathbf{M}_2}\sum_{\tau_i'\in\mathbf{T}_m'}\frac{C_i'}{T_i'}>|\mathbf{M}_2|. \tag{31}$$

As a result, we can get a lower bound on the total utilization over M processors by taking both utilization lower bounds in (30) and (31) as follows:

$$\sum_{\tau_i\in\mathbf{T}}u_i\geq\sum_{m\in\mathbf{M}_1}\sum_{\tau_i'\in\mathbf{T}_m'}\frac{C_i'}{T_i'}+\sum_{m\in\mathbf{M}_2}\sum_{\tau_i'\in\mathbf{T}_m'}\frac{C_i'}{T_i'}>\frac{|\mathbf{M}_1|k}{1+k+\beta}+|\mathbf{M}_2|\left(1-\frac{C_\ell}{T_\ell}\right). \tag{32}$$

The main reason why our analysis is tighter than the analysis by Baruah and Fisher (2005) for tasks with arbitrary deadlines is due to the additional consideration of the utilization lower bound in (32), whereas the analysis in Baruah and Fisher (2005) only uses the approximate demand bound function at time D_ℓ' in (28) for analyzing the resource augmentation factor. The following lemma provides a lower bound based on the average utilization and the maximum density among the tasks.

Lemma 9

$$\max\left\{\frac{\sum_{\tau_i\in\mathbf{T}}u_i}{M},\max_{\tau_i\in\mathbf{T}}\delta_i\right\}>\inf\left\{\max\left\{\frac{\frac{|\mathbf{M}_1|k}{1+k+\beta}+|\mathbf{M}_2|}{|\mathbf{M}_1|+2|\mathbf{M}_2|},\frac{\beta}{1+k+\beta}\right\}\right\}.$$

Proof This is based on (32), the definition of $\frac{C_\ell}{D_\ell}$, and solving the following equation by taking $\frac{C_\ell}{T_\ell}$ as a variable:

$$\frac{\frac{|\mathbf{M}_1|k}{1+k+\beta} + |\mathbf{M}_2|(1-\frac{C_\ell}{T_\ell})}{|\mathbf{M}_1|+|\mathbf{M}_2|} = \frac{C_\ell}{T_\ell}.$$

Therefore,

$$\begin{aligned}\max\left\{\frac{\sum_{\tau_i\in\mathbf{T}} u_i}{M}, \max_{\tau_i\in\mathbf{T}}\delta_i\right\} &\geq \max\left\{\frac{\sum_{\tau_i\in\mathbf{T}} u_i}{M}, \frac{C_\ell}{T_\ell}, \frac{C_\ell}{D_\ell}\right\}\\ &> \inf\left\{\max\left\{\frac{\frac{|\mathbf{M}_1|k}{1+k+\beta}+|\mathbf{M}_2|}{|\mathbf{M}_1|+2|\mathbf{M}_2|}, \frac{\beta}{1+k+\beta}\right\}\right\}.\end{aligned}$$

□

By putting everything together, we have the following lemma.

Lemma 10

$$\max\left\{\max_t \frac{\sum_{\tau_i\in\mathbf{T}}\mathrm{DBF}(\tau_i,t)}{Mt}, \sum_{\tau_i\in\mathbf{T}}\frac{C_i}{MT_i}, \max_{\tau_i\in\mathbf{T}}\delta_i\right\} > \frac{1}{3-\frac{1}{M}}.$$

Proof Let $|\mathbf{M}_2|$ be $x|\mathbf{M}_1|$. By (28) and Lemma 9, we know that

$$\begin{aligned}&\max\left\{\max_t \frac{\sum_{\tau_i\in\mathbf{T}}\mathrm{DBF}(\tau_i,t)}{Mt}, \sum_{\tau_i\in\mathbf{T}}\frac{C_i}{MT_i}, \max_{\tau_i\in\mathbf{T}}\delta_i\right\}\\ &> \inf\left\{\max_{0\leq k<1}\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{(1+x)(1+k+\beta)}, \frac{\beta}{1+k+\beta}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}\right\}\\ &= \inf\left\{\max_{0\leq k<1, x\geq\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{\beta}{1+k+\beta}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\},\right.\\ &\qquad\left.\max_{0\leq k<1, x<\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{(1+x)(1+k+\beta)}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}\right\}\\ &\geq^* \frac{1}{3-\frac{1}{|\mathbf{M}_1|}} \geq \frac{1}{3-\frac{1}{M}},\end{aligned} \tag{33}$$

where $\geq^*$ is proved in the Appendix. □

Therefore, we have the following theorem.

Theorem 4 *The resource augmentation factor for the deadline-monotonic partitioning algorithm is* $3-\frac{1}{M}$, *no matter which fitting strategy is adopted in Step* 5 *in Algorithm* 1.

Proof This comes directly from the analysis in Lemma 10. □

The following theorem shows that when $\max_{\tau_i \in T} \delta_i = \max_{\tau_i \in \mathbf{T}} \frac{C_i}{\min\{T_i, D_i\}}$ is less than $\frac{1}{3-\frac{1}{M}}$, the resource augmentation factor is actually $\frac{2}{1-\max_{\tau_i \in T} \delta_i}$.

Theorem 5 *No matter which fitting strategy is adopted in Step 5 in Algorithm 1, the resource augmentation factor for the deadline-monotonic partitioning algorithm is*

$$\min\left\{\frac{2}{1-\max_{\tau_i \in T} \delta_i}, 3-\frac{1}{M}\right\}. \tag{34}$$

Proof For notational brevity, we denote $\max_{\tau_i \in T} \delta_i$ by $\delta_{\max}$. Clearly, when $\delta_{\max} \geq \frac{1}{3-\frac{1}{M}}$, we can use Theorem 4 for bounding the resource augmentation factor.

Suppose $\delta_{\max} \leq \frac{1}{3-\frac{1}{M}}$, the resource augmentation factor is dominated by the workload resulting from the total utilization or the demand bound function of all the tasks. For the lower bound of these two, by (28) and (32), we have

$$\begin{aligned}
&\max\left\{\max_t \frac{\sum_{\tau_i \in \mathbf{T}} \mathrm{DBF}(\tau_i, t)}{Mt}, \sum_{\tau_i \in \mathbf{T}} \frac{C_i}{MT_i}\right\} \\
&> \inf\left\{\max\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{(1+x)(1+k+\beta)}, \frac{x(1-\delta_{\max})+\frac{k}{1+k+\beta}}{1+x}\right\}\right\} \\
&\geq \inf\left\{\max\left\{\frac{1}{(1+x)(1+k+\beta)}, \frac{x(1-\delta_{\max})+\frac{k}{1+k+\beta}}{1+x}\right\}\right\} \\
&=_1 \inf\left\{\frac{1-\delta_{\max}}{2+\beta-\delta_{\max}(1+k+\beta)}\right\} \geq_2 \frac{1-\delta_{\max}}{2},
\end{aligned}$$

where $=_1$ is due to the only intersection by taking $x = \frac{1-k}{(1-\delta_{\max})(1+k+\beta)}$ and $\geq_2$ comes from the fact that $\frac{\beta}{1+k+\beta}$ is no more than $\max_{\tau_i \in \mathbf{T}} \frac{C_i}{\min\{T_i, D_i\}}$, defined as $\delta_{\max}$.

Therefore, we conclude this theorem. □

4.2 Tasks with constrained deadlines

For constrained-deadline task sets, as shown in Lemma 2 in Baruah and Fisher (2006), Algorithm 1 can be simplified by removing the test with (7) in Steps 4 and 5. That is, the test with the approximate demand bound function in (6) is sufficient for schedulability guarantees of the derived partitioning.

Therefore, $|\mathbf{M}_1|$ is equal to M and $\mathbf{M}_2$ is an empty set. The main reason why our analysis is tighter than the analysis by Baruah and Fisher (2006) for tasks with constrained deadlines is due to the considerations of the utilization and the approximate demand bound function at any time $t \leq D'_\ell$, whereas the analysis in Baruah and Fisher (2006) only uses the approximate demand bound function at time D'_ℓ in (28) for analyzing the resource augmentation factor. Similar to Lemma 8, we know

that, for constrained-deadline task sets, the maximum density of the task set and the utilization should be lower-bounded by the following lemma.

Lemma 11 *If* $k \geq \frac{e-1}{e}(1+\frac{\beta}{M})$, *then*

$$\max\left\{\sum_{i=1}^{\ell-1}\frac{C_i'}{MT_i'}, \max_{0\leq t\leq D_\ell'}\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i',t)}{Mt}\right\} > \frac{\frac{e}{e-1}k}{1+k+\beta}.$$

Proof By reordering the tasks so that $D_i' \leq D_j'$ if $i<j$, the procedure of the proof is the same as that in the proofs from Lemma 4 to Lemma 8 in Sect. 3.3 by adjusting the relative deadlines and periods accordingly. □

Therefore, we can derive the resource augmentation factor for constrained-deadline task sets.

Theorem 6 *The resource augmentation factor for the deadline-monotonic partitioning algorithm is* $\frac{3e-1}{e}-\frac{1}{M}$, *which is about* $2.6322-\frac{1}{M}$, *when all the tasks have relative deadline shorter than or equal to the minimum-inter-arrival time, no matter which fitting strategy is adopted in Step 5 in Algorithm 1.*

Proof By Lemma 11, we know that

$$\begin{aligned}
&\max\left\{\sum_{\tau_i\in\mathbf{T}}\frac{C_i}{MT_i}, \max_t\frac{\sum_{\tau_i\in\mathbf{T}}\mathrm{DBF}(\tau_i,t)}{Mt}, \max_{\tau_i\in\mathbf{T}}\delta_i\right\}\\
&\geq \max\left\{\sum_{i=1}^{\ell-1}\frac{C_i'}{MT_i'}, \max_{0\leq t\leq D_\ell'}\frac{\sum_{i=1}^{\ell}\mathrm{DBF}(\tau_i',t)}{Mt}, \frac{C_\ell'}{D_\ell'}\right\}\\
&> \inf\left\{\max_{k\geq\frac{e-1}{e}(1+\frac{\beta}{M})}\left\{\frac{\frac{e}{e-1}k}{1+k+\beta}, \frac{\beta}{1+k+\beta}\right\},\right.\\
&\qquad\left.\max_{0\leq k<\frac{e-1}{e}(1+\frac{\beta}{M})}\left\{\frac{1+\frac{\beta}{M}}{1+k+\beta}, \frac{\beta}{1+k+\beta}\right\}\right\}\\
&= \frac{1}{\frac{3e-1}{e}-\frac{1}{M}}. \qquad (35)
\end{aligned}$$

□

For constrained-deadline task sets, the following theorem shows that when $\max_{\tau_i\in T}\delta_i = \max_{\tau_i\in\mathbf{T}}\frac{C_i}{D_i}$ is less than $\frac{1}{\frac{3e-1}{e}-\frac{1}{M}}$, the resource augmentation factor is actually $\frac{2e-1}{e}/(1-\max_{\tau_i\in T}\delta_i)$.

Theorem 7 *No matter which fitting strategy is adopted in Step 5 in Algorithm 1, for a constrained-deadline task set, the resource augmentation factor for the deadline-*

monotonic partitioning algorithm is

$$\min\left\{\frac{\frac{2e-1}{e}}{1-\max_{\tau_i\in T}\delta_i}, \frac{3e-1}{e}-\frac{1}{M}\right\}. \tag{36}$$

Proof For notational brevity, we denote $\max_{\tau_i\in T}\delta_i$ by $\delta_{\max}$. Clearly, when $\delta_{\max}\geq \frac{1}{\frac{3e-1}{e}-\frac{1}{M}}$, we can use Theorem 6 for bounding the resource augmentation factor.

Suppose $\delta_{\max}\leq \frac{1}{\frac{3e-1}{e}-\frac{1}{M}}$, the resource augmentation factor is dominated by the workload resulting from the total utilization or the demand bound function of all the tasks. We know that $\delta_{\max}\geq \frac{\beta}{1+k+\beta}$, which implies $\beta\leq \frac{(1+k)\delta_{\max}}{1-\delta_{\max}}$. By Lemma 11 and $\beta\leq \frac{(1+k)\delta_{\max}}{1-\delta_{\max}}$, we have

$$\begin{aligned}
&\max\left\{\max_t \frac{\sum_{\tau_i\in\mathbf{T}}\mathrm{DBF}(\tau_i,t)}{Mt}, \sum_{\tau_i\in\mathbf{T}}\frac{C_i}{MT_i}\right\}\\
&> \inf\left\{\inf_{0\leq k<\frac{e-1}{e}}\frac{1}{1+k+\beta}, \inf_{\frac{e-1}{e}\leq k<1}\frac{\frac{e}{e-1}k}{1+k+\beta}\right\}\\
&\geq \inf\left\{\inf_{0\leq k<\frac{e-1}{e}}\frac{1}{1+k+\frac{(1+k)\delta_{\max}}{1-\delta_{\max}}}, \inf_{\frac{e-1}{e}\leq k<1}\frac{\frac{e}{e-1}k}{1+k+\frac{(1+k)\delta_{\max}}{1-\delta_{\max}}}\right\}\\
&= \inf\left\{\inf_{0\leq k<\frac{e-1}{e}}\frac{1-\delta_{\max}}{1+k}, \inf_{\frac{e-1}{e}\leq k<1}\frac{\frac{e}{e-1}(1-\delta_{\max})k}{1+k}\right\}\\
&= \frac{1-\delta_{\max}}{\frac{2e-1}{e}}.
\end{aligned}$$

Hence, we conclude this theorem. □

4.3 Lower bound on resource augmentation for deadline-monotonic partitioning

Similar to Sect. 3.4, we now present the lower bound on the resource augmentation factors (in the worst cases) based on the deadline-monotonic partitioning.

Theorem 8 *When $M\geq 2$, the resource augmentation factor of deadline-monotonic partitioning is at least $\frac{2.5-\frac{1}{M}}{1+\frac{1.5}{M}-\frac{2}{M^2}}$ when Step 5 in Algorithm 1 uses*

- *a random fitting strategy or*
- *the worst-fit strategy based on utilization.*

Proof We prove this theorem by providing a concrete input instance. There are $N=M(M-1)+1$ tasks, in which for $i=1,2,\ldots,M(M-1)$, task τ_i has relative deadline D_i equal to $(2.5-\frac{1}{M})(\lfloor\frac{i-1}{M-1}\rfloor+1)$, minimum inter-arrival time T_i equal to $M(2.5-\frac{1}{M})$, execution time C_i equal to 1. Task τ_N is with $D_N=M(2.5-\frac{1}{M})$ and $C_N=M+1.5-\frac{2}{M}+\epsilon$ and $T_N=\infty$ with $\epsilon>0$.

For the worst-fit strategy based on utilization and a special (worst) case of the random fitting strategy, for $i = 1, 2, \ldots, M(M-1)$, each processor is assigned $M-1$ tasks and at most 2 of the $M-1$ tasks are with the same relative deadline on a processor. The approximate demand bound for each processor when considering τ_N is no less than the case considering the approximate demand bound function for $(M-3)$ tasks with relative deadlines $2(2.5-\frac{1}{M}), 3(2.5-\frac{1}{M}), \ldots, (M-2)(2.5-\frac{1}{M})$ and two tasks both with relative deadline $(M-1)(2.5-\frac{1}{M})$, i.e.,

$$\sum_{\tau_j \in \mathbf{T}_m} \mathrm{DBF}^*(\tau_j, D_N) \geq (M-1) + \frac{1}{M} + \sum_{i=2}^{M-1} \left(1 - \frac{i}{M}\right)$$

$$= 1.5M - 2.5 + \frac{2}{M}.$$

Therefore, since $C_N + \sum_{\tau_j \in \mathbf{T}_m} \mathrm{DBF}^*(\tau_j, D_N) \geq 2.5M - 1 + \epsilon > D_N$ for all $m = 1, 2, \ldots, M$, Algorithm 1 fails to derive a feasible task partitioning.

The utilization of the task set is

$$\sum_{i=1}^{N} \frac{C_i}{T_i} = \frac{M(M-1)}{M(2.5-\frac{1}{M})} + \frac{C_N}{\infty} = \frac{M-1}{2.5-\frac{1}{M}}. \tag{37}$$

The maximum density of a single task is

$$\max_{i=1}^{N} \delta_i = \delta_N = \frac{1 + \frac{1.5+\epsilon}{M} - \frac{2}{M^2}}{2.5 - \frac{1}{M}}. \tag{38}$$

By assigning task τ_i to processor $[(i-1) \mod (M-1)] + 1$ for $i \leq M(M-1)$ and τ_N to processor M, the task set is feasible by running at any speed no less than factor $\frac{1+\frac{1.5+\epsilon}{M}-\frac{2}{M^2}}{2.5-\frac{1}{M}}$ of the original speed. Therefore, the resource augmentation factor of Algorithm 1 is at least $\frac{2.5-\frac{1}{M}}{1+\frac{1.5}{M}-\frac{2}{M^2}}$ when $\epsilon \to 0$. □

Theorem 9 *When $M > 1$, the resource augmentation factor of deadline-monotonic partitioning is at least $2 - \frac{2}{M+1}$ when Step 5 in Algorithm 1 uses the first-fit strategy.*

Proof We prove this theorem by providing a concrete input instance. There are $N = 2M$ tasks. For $i = 1, 2, \ldots, M$, task τ_i has relative deadline D_i equal to $2M - \epsilon$ with $\epsilon > 0$, minimum inter-arrival time $T_i \geq D_i$, execution time C_i equal to 1. For $i = M, M+1, \ldots, 2M$, τ_i has relative deadline $2M$, minimum inter-arrival time equal to the relative deadline, and execution time $M + \epsilon$.

By assigning the first M tasks on the first processor with the first-fit strategy, Algorithm 1 fails to assign task τ_N. By assigning task τ_i and task τ_{i+M} on processor i, the solution is feasible by running at any speed no less than factor $\frac{1+\frac{1+\epsilon}{M}}{2-\frac{\epsilon}{M}}$ of the

original speed. Therefore, the resource augmentation factor is at least

$$\lim_{\epsilon \to 0} \frac{2 - \frac{\epsilon}{M}}{1 + \frac{1+\epsilon}{M}} = \frac{2}{1 + \frac{1}{M}} = 2 - \frac{2}{M+1}. \qquad \square$$

Note that the above theorems on lower bounds of resource augmentation are based on concrete input instances. Even though the lower bound of the first-fit strategy is lower than the others on multiprocessor systems, we do not claim that the first-fit strategy has a better worst-case resource augmentation factor than others. The gap between these lower bounds stems from the difficulty in identifying a set of concrete input instances for the first-fit strategy.

The lower bounds provided here are different from the lower bound $2 - \frac{2}{M}$ provided in Theorem 1 in Fisher (2009), which comes from the resource augmentation by comparing the *partitioned scheduling* to the *global scheduling*. The lower bounds provided here are for specific fitting strategies with comparison to feasible partitioned scheduling. In fact, there does not exist any feasible partitioned schedule for the task set provided in Theorem 1 in Fisher (2009) at the original speed. Therefore, by the definition of the resource augmentation factor in this paper, that instance has a resource augmentation factor 1, in which the feasibility of a schedule is defined by *partitioned scheduling*.

5 Polynomial-time approximation schemes

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 involved than the deadline-monotonic partitioning algorithm. Therefore, from an implementation perspective, as also suggested in Baruah (2011), it is recommended to use the deadline-monotonic partitioning first, and our proposed PTAS should be adopted when deadline-monotonic partitioning fails to derive feasible task partitions.

The PTAS is based on a reduction to the vector scheduling problem by applying the results by Chekuri and Khanna (2004).

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 (Chekuri and Khanna 2004). When d is not

a constant, the vector scheduling problem can be approximated with a $O(\log d)$ approximation factor (Chekuri and Khanna 2004). We will adopt the polynomial-time approximation scheme by Chekuri and Khanna (2004) for the vector scheduling problem with a constant dimension d.

Lemma 12 (Chekuri and Khanna 2004) *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* $s = O((\frac{\ln(\frac{d}{\epsilon})}{\epsilon})^d)$.

For completeness, we will briefly summarize how the proposed PTAS in Chekuri and Khanna (2004) works. For a vector v_i, the quantity $\|v_i\|_\infty$ denotes the standard ℓ_∞ norm, in which $\|v_i\|_\infty = \max_j\{q_{i,j}\}$. For a set $\mathbf{V}_m$ of vectors, the quantity $\|\mathbf{V}_m\|_\infty$ denotes the standard ℓ_∞ norm for the sum of vectors in the set, in which $\|\mathbf{V}_m\|_\infty = \max_j\{\sum_{v_i \in \mathbf{V}_m} q_{i,j}\}$.

Let I be an instance of the vector scheduling problem. The first step is to construct another instance $I^\flat$ by ruling out some values in the vectors. Suppose that κ is $\frac{\epsilon}{d}$. If $q_{i,j} \le \kappa\|v_i\|_\infty$, we can consider that $q_{i,j}$ is not significant and reduce its value to 0. Let $\mathbf{V}_m$ be a set of vectors in I and let $\mathbf{V}_m^\flat$ be the corresponding set of vectors in $I^\flat$. It can be shown that

$$\|\mathbf{V}_m\|_\infty \le (1+\epsilon)\|\mathbf{V}_m^\flat\|_\infty.$$

Then, the algorithm divides the vectors in $I^\flat$ into two types: *heavy* and *light* vectors. A vector v_i is classified as a heavy vector if $\|v_i\|_\infty > \kappa$; otherwise, v_i is a light vector. It is not difficult to see that the minimum non-zero value in the heavy vectors in $I^\flat$ is at least κ^2.

The algorithm first packs the heavy vectors by rounding the non-zero elements in the vectors and discretizing the interval $[(\frac{\epsilon}{d})^2, 1]$ into consecutive subintervals as follows: The i-th subinterval starts from value x_{i-1} and ends with value x_i defined as $(1+\epsilon)x_{i-1}$. By taking x_0 as κ^2, we need $\lceil \frac{2}{\epsilon}\ln\frac{1}{\kappa}\rceil$ subintervals. If the element $q_{i,j}$ is in the interval $(x_i, x_{i+1}]$, the element is rounded down to x_i.

After the rounding, there are only $(1+\lceil \frac{2}{\epsilon}\ln\frac{1}{\kappa}\rceil)^d$ different types of heavy vectors. By the assumption that $\frac{1}{\epsilon}$ and d are both constants, the number of different types of vectors is also a constant. By applying a dynamic programming approach, the optimal vector scheduling for the heavy vectors after rounding can be derived in polynomial time. Then, after the partition of heavy vectors is done, the algorithm starts to pack the light vectors by using linear programming.

After introducing the vector scheduling problem and its algorithm, we will present our PTAS for multiprocessor partitioned scheduling in Sect. 5.1. Moreover, Section 5.2 will give some remarks about efficiency by reducing unnecessary dimensions. Section 5.3 provides the extensions to some related problems.

5.1 Our framework for PTAS

Our proposed PTAS for the multiprocessor partitioned scheduling problem is based on the polynomial-time approximation scheme in Lemma 12. 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 13, we may be able to reduce the number of tests without incurring errors.

Lemma 13 *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*

$$\forall \tau_i \in \mathbf{T}_m,\ n \in \mathcal{N}, \quad \sum_{\tau_j \in \mathbf{T}_m} \mathrm{DBF}(\tau_j, t_{i,n}) \le t_{i,n}.$$

Proof This lemma comes directly from the demand bound function analysis in Baruah et al. (1990). □

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

$$\Delta_{\max} \stackrel{\text{def}}{=} \max_{m=1,2,\ldots,M, t>0} \frac{\sum_{\tau_i \in \mathbf{T}_m} \mathrm{DBF}(\tau_i, t)}{t}. \tag{39}$$

Therefore, according to the definition of the existence of a feasible task partition on M processors, a feasible task partition requires $\Delta_{\max} \le 1$. Suppose that $\Delta^*_{\max}$ is the minimum $\Delta_{\max}$ among the feasible task partitions. Therefore, the multiprocessor partitioned scheduling problem is equivalent to the minimization of $\Delta_{\max}$. *Therefore, for the rest of this section, when considering the multiprocessor partitioned scheduling problem, we will implicitly target at the polynomial-time approximation schemes for the minimization of* $\Delta_{\max}$. As a result, the optimal task partition $\mathbf{T}^*_1, \mathbf{T}^*_2, \ldots, \mathbf{T}^*_M$ has the minimum $\Delta_{\max}$ among the feasible task partitions.

According to the above analysis, it is very clear that the multiprocessor partitioned scheduling problem is equivalent to the vector scheduling problem with *infinite dimensions* by taking $\frac{\mathrm{DBF}(\tau_j,t)}{t}$ as a corresponding element of time t for a vector representing τ_j, where t is equal to $t_{i,n}$ under any non-negative n. 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 by reducing the problem to the vector scheduling problem with constant dimensions.

Lemma 14 *For a given task set $\mathbf{T}$ of the multiprocessor partitioned scheduling problem and 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*

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

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 $q_{i,j} = \frac{\mathrm{DBF}(\tau_i, t_j)}{t_j}$ for $j = 1, 2, \ldots, d$. Then, we can adopt the polynomial-time approximation scheme in Lemma 12 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 12 and the resource augmentation factor is $1 + \epsilon$. □

The condition in Lemma 14 is not easy to satisfy by selecting only a constant number of time points for exact schedulability tests. 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 $\mathbf{V}$

$$\Delta(\mathbf{T}') \geq \|\mathbf{V}'\|_\infty \geq \left(\frac{1}{1+\eta}\right)\Delta(\mathbf{T}'), \tag{40}$$

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

Theorem 10 *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 12 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 12 and the assumption with the tolerable error in (40),

$$\max_{m=1,2,\ldots,M} \Delta\left(\mathbf{T}_m^\dagger\right) \leq (1+\eta) \max_{m=1,2,\ldots,M} \|\mathbf{V}_m^\dagger\|_\infty \tag{41}$$

$$\leq_1 (1+\eta)(1+\epsilon) \max_{m=1,2,\ldots,M} \|\mathbf{V}_m^*\|_\infty \tag{42}$$

$$\leq_2 (1+\eta)(1+\epsilon) \max_{m=1,2,\ldots,M} \Delta\left(\mathbf{T}_m^*\right) \tag{43}$$

$$= (1+\eta)(1+\epsilon)\Delta_{\max}^* \tag{44}$$

where $\leq$ and $\leq_2$ come from the inequality assumed in (40) for the representative vector set, and $\leq_1$ is from Lemma 12.

Therefore, the derived solution has an approximation factor guarantee $(1+\epsilon) \times (1+\eta)$ 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 12 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 12. 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.

5.1.1 $\frac{D_N}{D_1}$ *is a constant*

Suppose λ is defined as the constant $\frac{D_N}{D_1}$ in $\mathbf{T}$, i.e., $\lambda \stackrel{\text{def}}{=} \frac{D_N}{D_1}$. Let h be $\lceil \frac{\ln(1+\eta)}{\ln \frac{\lambda}{\eta}} \rceil$. 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: $[D_1, (1+\eta)D_1), [(1+\eta)D_1, (1+\eta)^2 D_1), \ldots, [(1+\eta)^{h-1}D_1, (1+\eta)^h D_1)$.

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

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

Figure 5 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 functions. However, the following lemma shows that the error we make by taking only $h+3$ points is also bounded.

Lemma 15 *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* (40) *holds.*

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

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

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 $\|\mathbf{V}'\|_\infty \geq \frac{1}{1+\eta}\Delta(\mathbf{T}')$. 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 $\sum_{\tau_i \in \mathbf{T}'} \frac{\mathrm{DBF}(\tau_i, t)}{t}$ 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,

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

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., $\mathrm{DBF}(\tau_i, t^*) \leq \mathrm{DBF}(\tau_i, z_j)$.

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

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

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

Therefore, we reach the inequality described by (40). □

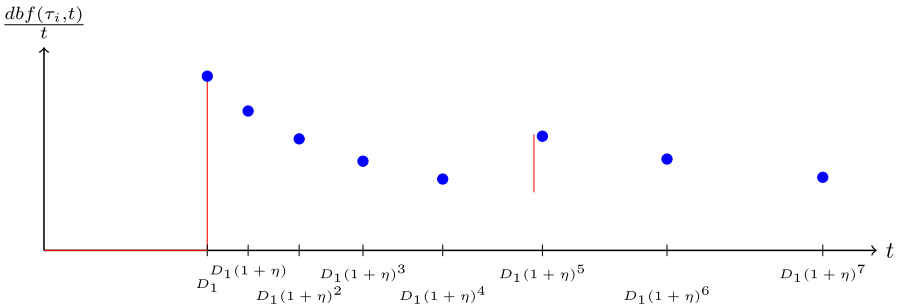


Fig. 5 An example for the first $h+1$ dimensions, where $h=7$. *The circles* are used to construct the dimensions along with the demand bound function when $t=\infty$ and $t=D_N$

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 11 *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 15, 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 $h=\lceil\frac{\ln(1+\eta)}{\ln\frac{\lambda}{\eta}}\rceil$ is a constant, provided that $\frac{D_N}{D_1}=\lambda$ and $\frac{1}{\eta}$ are both constants. Therefore, by Lemma 15 and Theorem 10, we reach the conclusion. □

5.1.2 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., $\lambda_k\le\lambda$ for $1\le k\le\sigma$. 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.[1]

Let h be defined as $\lceil\frac{\ln(1+\eta)}{\ln\frac{\lambda}{\eta}}\rceil$, as in Sect. 5.1.1. For the k-th cluster, similar to Sect. 5.1.1, we can discretize the time interval $[D_{w_{k-1}+1},\frac{\lambda D_{w_{k-1}+1}}{\eta}]$ to into h sub-intervals: $[D_{w_{k-1}+1},(1+\eta)D_{w_{k-1}+1}),[(1+\eta)D_{w_{k-1}+1},(1+\eta)^2w_{k-1}+1),\ldots,$ $[(1+\eta)^{h-1}w_{k-1}+1,(1+\eta)^hw_{k-1}+1]$, denoted by $[z_{k,0},z_{k,1}),\ [z_{k,1},z_{k,2}),\ldots,$

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

$[z_{k,h-1}, z_{k,h})$. Then, we create a vector v_i with $\sigma(h+2)+1$ dimensions for task τ_i by setting

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

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

Lemma 16 *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* (40) *holds.*

Proof The proof is very similar to that for Lemma 15. The proof for $\Delta(\mathbf{T}') \geq \|\mathbf{V}'\|_\infty$ 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 15.

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

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

□

Theorem 12 *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 10 and Lemma 16. □

5.2 Efficiency and reducing unnecessary dimensions

Even though we have errors both by applying the PTAS proposed in Chekuri and Khanna (2004) and by taking only the constant dimensional representative vector set,

we have shown in Sect. 5.1 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 Sect. 5.1 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 function is 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 17 *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 13. 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 $\frac{\mathrm{DBF}(\tau_i,t)}{t} = \frac{\mathrm{DBF}(\tau_i,z_{j-1})}{t} \leq \frac{\mathrm{DBF}(\tau_i,z_{j-1})}{z_{j-1}}$. □

Lemma 18 *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 $z_{j-1} < t \leq z_j$,

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

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 19 *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 18. □

As the examinations for the conditions in Lemmas 17, 18, and 19 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.

5.3 Extensions to related problems

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 Baruah (2013). 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. Moreover, as the time points that are selected to construct the representative vectors are independent from the execution times of tasks, we can also use the same strategy for *heterogeneous multiprocessor partitioned scheduling*. For heterogeneous multiprocessor systems, a task τ_i requires execution time $c_{i,m}$ on processor m. The corresponding *heterogeneous vector scheduling* problem is to consider systems, in which the values of the elements in a vector may be different when the vector is assigned on different processors. Bonifaci and Wiese (2012) have developed a PTAS for the heterogeneous vector scheduling problem when the number of dimensions and the number of heterogeneous processors are both constants. Therefore, our PTAS can also be extended to deal with the *heterogeneous multiprocessor partitioned scheduling* problem when the number of processors and $\frac{D_{\max}}{D_{\min}}$ are both constants.

6 Concluding remarks

This paper studies the resource augmentation factor for uniprocessor and multiprocessor partitioned scheduling. For the deadline-monotonic partitioning algorithm proposed in Baruah and Fisher (2005, 2006) , this paper provides tighter analysis for the upper bound on the resource augmentation factor. We show that the resource augmentation factor is at most $\frac{2e-1}{e} \approx 1.6322$ for uniprocessor systems, at most $\frac{3e-1}{e} - \frac{1}{M} \approx 2.6322 - \frac{1}{M}$ for constrained-deadline sporadic task sets on identical multiprocessor systems, and $3 - \frac{1}{M}$ and for arbitrary-deadline sporadic task sets on identical multiprocessor systems. In multiprocessor scheduling, for special cases in which the maximum density of the tasks is not large, we also provide tighter analysis. When the relative deadline is not much smaller than the minimum inter-arrival time, reducing the input instance to a task set with implicit deadlines can also have a bounded resource augmentation factor. Moreover, for some special cases in which the maximum relative deadline divided by the minimum relative deadline is a constant, we also provide a polynomial-time approximation scheme to trade the efficiency and the approximation for resource augmentation.

When the approximate demand bound function is adopted, we also present concrete input instances, in which the analytical lower bound of the resource augmentation factor for uniprocessor systems is 1.5. Similarly, for deadline-monotonic partitioning with the approximate demand bound function in identical multiprocessor systems with M processors, when the first-fit strategy is used for task assignment, the analytical lower bound for the resource augmentation factor is $2 - \frac{2}{M+1}$, while the

worst-fit strategy or the random-fit strategy has an analytical lower bound $\frac{2.5-\frac{1}{M}}{1+\frac{1.5}{M}-\frac{2}{M^2}}$. When M is sufficiently large, the corresponding lower bounds above are 2 and 2.5.

Our analysis for the upper bounds of the resource augmentation factors of the deadline-monotonic partitioning also holds for reasoning about the augmentation factor by assuming that there is a feasible identical multiprocessor global scheduling. This is because the statement in Lemma 1 also holds for global scheduling and the optimal global scheduling is superior to partitioned scheduling. The upper and lower bounds of the resource augmentation factors provided in this paper for the deadline-monotonic partitioning are also correct when compared to optimal global scheduling. However, by Theorem 1 in Fisher (2009), the resource augmentation of partitioned scheduling is at least $2-\frac{2}{M}$ against global scheduling. The instance provided in Theorem 9 is not representative any more.

For multiprocessor global scheduling, Bonifaci et al. (2008) have shown that the resource augmentation factor for global EDF is $2-\frac{1}{M}$, while the schedulability test also requires the verification of (2). To achieve polynomial-time tests, the approximation of the demand bound function is also required in Bonifaci et al. (2008). Another interesting research direction is to explore the resource augmentation factors for semi-partitioned scheduling algorithms. Specifically, for implicit-deadline tasks, in Guan et al. (2010), Guan et al. propose a fixed-priority scheduling algorithm, which has been shown to have a utilization bound $N(2^{\frac{1}{N}}-1)$ (the same as the Liu and Layland bound Liu and Layland 1973) for semi-partitioned scheduling. This, of course, also implies the resource augmentation factor of the algorithm in Guan et al. (2010) to be at most $\frac{1}{N(2^{\frac{1}{N}}-1)}$. However, we are not aware of any results for assuring the resource augmentation factors for task sets with constrained or arbitrary deadlines.

For future research, we would like to tighten the analysis, and consider uniform multiprocessor systems. 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. Moreover, the first-fit strategy seems to be a better preference, but how to analyze its behavior tightly also remains as an open problem. Last, but not least, designing semi-partitioned scheduling algorithms with good resource augmentation bounds for constrained- or arbitrary-deadline task sets is of course also an interesting research direction.

Acknowledgements This work was partially supported by Baden Württemberg MWK Juniorprofessoren-Programms. We also thank the anonymous reviewers for their valuable feedback in helping improve this paper.

Appendix

Proof of Inequality $>^*$ in (33) If $\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1<0$, then we know that

$$\inf_{0\le k<1}\frac{\beta}{1+k+\beta}=\inf_{0\le k<1}\frac{\beta}{2+\beta}>\frac{1}{1+2-\frac{2}{|\mathbf{M}_1|}}\ge\frac{1}{3-\frac{1}{|\mathbf{M}_1|}}$$

For the rest, we consider $\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1\geq 0$. The infimum for

$$\max_{0\leq k<1, x\geq \frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{\beta}{1+k+\beta}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}$$

happens when k is equal to $\beta-1+\frac{1}{1+x}$. Therefore,

$$\begin{aligned}
&\max_{0\leq k<1, x\geq \frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1\geq 0}\left\{\frac{\beta}{1+k+\beta}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}\\
&\quad\geq \inf_{x\geq \frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1\geq 0}\left\{\frac{\beta}{2\beta+\frac{1}{1+x}}\right\} \geq \inf_{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}\geq 1}\frac{\beta}{2\beta+\frac{1}{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}}}\\
&\quad= \inf_{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}\geq 1}\frac{1}{3-\frac{\frac{1}{|\mathbf{M}_1|}}{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}}} \geq \frac{1}{3-\frac{1}{|\mathbf{M}_1|}}.
\end{aligned}$$

When k equals to $\frac{(1+\frac{\beta}{|\mathbf{M}_1|})(2x+1)-(x^2+x)(1+\beta)}{(1+x)^2}$, the infimum for

$$\max_{0\leq k<1, x<\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{(1+x)(1+k+\beta)}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}$$

happens. Therefore, we have

$$\begin{aligned}
&\max_{0\leq k<1, 0\leq x<\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{(1+x)(1+k+\beta)}, \frac{\frac{k}{1+k+\beta}+x}{1+2x}\right\}\\
&\quad\geq \inf_{0\leq x<\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}-1}\left\{\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{1+\beta+\frac{(1+\frac{\beta}{|\mathbf{M}_1|})(2x+1)}{1+x}}\right\}\\
&\quad= \inf_{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}\geq 1}\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{1+\beta+2(1+\frac{\beta}{|\mathbf{M}_1|})-\frac{1+\frac{\beta}{|\mathbf{M}_1|}}{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}}}\\
&\quad= \inf_{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}\geq 1}\frac{1}{2+\frac{|\mathbf{M}_1|}{\beta+|\mathbf{M}_1|}} = \inf_{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}\geq 1}\frac{1}{3-\frac{\frac{1}{|\mathbf{M}_1|}}{\frac{1}{\beta}+\frac{1}{|\mathbf{M}_1|}}} \geq \frac{1}{3-\frac{1}{|\mathbf{M}_1|}}.
\end{aligned}$$

□

References

Albers K, Slomka F (2004) An event stream driven approximation for the analysis of real-time systems. In: ECRTS, pp 187–195

Albers K, Slomka F (2005) Efficient feasibility analysis for real-time systems with EDF scheduling. In: DATE, pp 492–497

Baruah S (2011) The partitioned EDF scheduling of sporadic task systems. In: Real-time systems symposium (RTSS), pp 116–125

Baruah S (2013) Partitioning sporadic task systems upon memory-constrained multiprocessors. ACM Trans Embed Comput Syst 12(3):78

Baruah SK, Fisher N (2005) The partitioned multiprocessor scheduling of sporadic task systems. In: RTSS, pp 321–329

Baruah SK, Fisher N (2006) The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. IEEE Trans Comput 55(7):918–923

Baruah SK, Mok AK, Rosier LE (1990) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: IEEE real-time systems symposium, pp 182–190

Bonifaci V, Wiese A (2012) Scheduling unrelated machines of few different types. In: First interdisciplinary workshop on algorithmic challenges in real-time systems

Bonifaci V, Marchetti-Spaccamela A, Stiller S (2008) A constant-approximate feasibility test for multiprocessor real-time scheduling. In: ESA, pp 210–221

Bonifaci V, Chan H-L, Marchetti-Spaccamela A, Megow N (2010) Algorithms and complexity for periodic real-time scheduling. In: SODA, pp 1350–1359

Chakraborty S, Künzli S, Thiele L (2002) Approximate schedulability analysis. In: IEEE real-time systems symposium, pp 159–168

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

Davis RI, Burns A (2011) A survey of hard real-time scheduling for multiprocessor systems. ACM Comput Surv 43(4):35

Eisenbrand F, Rothvoß T (2010) EDF-schedulability of synchronous periodic task systems is coNP-hard. In: SODA, pp 1029–1034

Fisher NW (2009) How hard is partitioning for the sporadic task model? In: Proceedings of the 2009 international conference on parallel processing workshops, ICPPW'09, pp 2–5

Garey MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, New York

Graham RL (1969) Bounds on multiprocessing timing anomalies. SIAM J Appl Math 17(2):416–429

Guan N, Stigge M, Yi W, Yu G (2010) Fixed-priority multiprocessor scheduling with Liu and Layland's utilization bound. In: IEEE real-time and embedded technology and applications symposium, pp 165–174

Hochbaum DS, Shmoys DB (1987) Using dual approximation algorithms for scheduling problems theoretical and practical results. J ACM 34(1):144–162

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. J ACM 20(1):46–61

Mok AK (1983) Fundamental design problems of distributed systems for the hard-real-time environment. Technical report Cambridge, MA, USA

Jian-Jia Chen is a Juniorprofessor at Department of Informatics in Karlsruhe Institute of Technology (KIT) in Germany. He received his Ph.D. degree from Department of Computer Science and Information Engineering, National Taiwan University, Taiwan in 2006. He received his B.S. degree from the Department of Chemistry at National Taiwan University 2001. After finishing the compulsory civil service in Dec. 2007, between Jan. 2008 and April 2010, he was a postdoc researcher at Computer Engineering and Networks Laboratory (TIK) in ETH Zurich, Switzerland. He joined KIT in May 2010. His research interests include real-time systems, embedded systems, energy-efficient scheduling, power-aware designs, temperature-aware scheduling, and distributed computing. He received Best Paper Awards from ACM Symposium on Applied Computing (SAC) in 2009 and IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA) in 2005.

Samarjit Chakraborty is a Full Professor of Real-Time and Embedded Systems at the Technical University of Munich in Germany. He also leads a research program on embedded systems design for electric vehicles at the TUM-CREATE Centre for Electromobility in Singapore, where he serves as a Scientific Advisor. Prior to joining TU Munich in 2008, he was an Assistant Professor of Computer Science at the National University of Singapore from 2003–2008. He obtained his Ph.D. in Electrical and Computer Engineering from ETH Zurich in 2003. His research interests include distributed embedded systems, hardware/software co-design, embedded control systems, and sensor network-based information processing for healthcare, smart-buildings and electromobility.