Capacity Augmentation Bounds for Parallel DAG Tasks under G-EDF and G-RM

Jian-Jia Chen§ and Kunal Agrawal†

§Computer Science 12 at TU Dortmund
†Washington University in St. Louis, U.S.A.

Number: 845
July 2014
ABSTRACT

This paper considers global earliest-deadline-first (EDF) and global rate-monotonic scheduling for a general task model for parallel sporadic real-time tasks. In particular, each sporadic real-time task is characterized by the general directed acyclic graph (DAG). This paper provides the utilization-based analysis to test the schedulability of global EDF and global rate-monotonic scheduling. We show that if on unit-speed processors, a task set has total utilization of at most \( \frac{m}{2} \) and the critical path length of each task is smaller than its deadline, then global EDF can schedule that task set on \( m \) processors of speed \( \frac{3 + \sqrt{5}}{2} \approx 2.6181 \), defined as the capacity augmentation bound. Together with the lower bound on the speeding up, we close the gap for global EDF when \( m \) is sufficiently large. This is the best known capacity augmentation bound for parallel DAG tasks under any scheduling strategy. In addition, we also show that global rate monotonic scheduling has a capacity augmentation bound of \( 2 + \sqrt{3} \approx 3.7321 \) with a similar analysis procedure, the best known capacity augmentation bound for fixed priority scheduling of the general DAG tasks. For global EDF and global RM, we also present utilization-based schedulability analysis tests based on the utilization and the maximum critical path utilization.

ACKNOWLEDGMENTS

Part of the results of this paper will appear in Euromicro Conference on Real-Time Systems (ECRTS) 2014 in the paper “ANALYIS OF FEDERATED AND GLOBAL SCHEDULING FOR PARALLEL REAL-TIME TASKS”.

This research was supported in part by the priority program “Dependable Embedded Systems” (SPP 1500 - spp1500.itec.kit.edu), by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/).
## CONTENTS

1 INTRODUCTION  1
2 SYSTEM MODEL  4
3 CANONICAL FORM OF A DAG TASK  6
4 GLOBAL EDF  11
   4.1 Upper Bound on Capacity Augmentation of G-EDF  11
   4.2 Lower Bound on Capacity Augmentation of G-EDF  13
   4.3 Utilization-Based Schedulability Test  14
5 G-RM SCHEDULING  16
6 RELATED WORK  19
7 CONCLUSIONS  21
INTRODUCTION

In the last decade, multicore processors have become ubiquitous and there has been extensive work on how to exploit these parallel machines for real-time tasks. In the real-time systems community, there has been extensive research on scheduling task sets with inter-task parallelism — where each task in the task set is a sequential program. In this case, increasing the number of cores allows us to increase the number of tasks in the task set. However, since each task can only use one core at a time, the computational requirement of a single task is still limited by the capacity of a single core. Recently, there has been some interest in design and analysis of scheduling strategies for task sets with intra-task parallelism (in addition to inter-task parallelism), where individual tasks are parallel programs and can potentially utilize more than one core in parallel. These models enable tasks with higher execution demands and tighter deadlines, such as those used in autonomous vehicles [30], video surveillance, computer vision, radar tracking and real-time hybrid testing [28].

In this paper, we consider the general directed acyclic graph (DAG) model. We prove that both global EDF and global rate-monotonic schedulers provide strong performance guarantees, in the form of capacity augmentation bounds, for scheduling these parallel DAG tasks.

One can generally derive two kinds of performance bounds for real time schedulers. The traditional bound is called resource augmentation bound (also called processor speed-up factor). A scheduler $A$ provides a resource augmentation bound of $b \geq 1$ if it can successfully schedule any task set $T$ on $m$ processors of speed $b$ as long as the ideal scheduler can schedule $T$ on $m$ processors of speed $1$. A resource augmentation bound provides a good notion of how close a scheduler is to the optimal schedule, but has a drawback. Note that the ideal scheduler is only a hypothetical scheduler, meaning that it always finds a feasible schedule if one exists. Unfortunately, Fisher et al. [26] proved that optimal online multiprocessor scheduling of sporadic task systems is impossible. Since, often, we can not tell whether the ideal scheduler can schedule a given task set on unit-speed processors, a resource augmentation bound may not provide a schedulability test.

The other kind of bound that is commonly used is a utilization bound. A scheduler $A$ provides a utilization bound of $b$ if it can successfully schedule any task set which has total utilization at most $m/b$ on $m$ processors. A utilization bound provides more information than a resource augmentation bound does; any scheduler that guarantees a utilization bound of $b$ automatically guarantees a resource augmentation bound of $b$ as well. In addition, it acts as a very simple schedulability test in itself, since the total utilization of the task set can be calculated in linear time and compared to $m/b$. Finally, a utilization bound gives an indication of how much load a system can handle; allowing us to estimate how much over-provisioning may be necessary when designing a platform. Unfortunately, it is often impossible to prove a utilization bound for

---

1 A utilization bound is often stated in terms of $1/b$; we adopt this notation in order to be consistent.
parallel systems; often, we can construct pathological task sets with utilization arbitrarily close to 1, but which cannot be scheduled on m processors.

Li et al. [33] defined a concept of capacity augmentation bound which is similar to the utilization bound, but adds a new condition. A scheduler $A$ provides a capacity augmentation bound of $b$ if it can schedule any task set $T$ which satisfies the following two conditions: (1) the total utilization of $T$ is at most $m/b$, and (2) the worst-case critical-path length of each task $\Phi_i$ (execution time of the task on an infinite number of processors)$^2$ is at most $1/b$th fraction of its deadline. A capacity augmentation bound is quite similar to a utilization bound: It also provides more information than a resource augmentation bound does; any scheduler that guarantees a capacity augmentation bound of $b$ automatically guarantees a resource augmentation bound of $b$ as well. It also acts as a very simple schedulability test. Finally, it can also provide help while designing a platform by allowing one to estimate the load it is expected to handle.

There has been some recent research on proving both resource augmentation bounds and capacity augmentation bounds for various scheduling strategies for parallel tasks. This work falls into two categories. In decomposition-based strategies, the parallel task is decomposed into a set of sequential tasks and these sequential tasks are scheduled using existing strategies for scheduling sequential tasks on multiprocessors. In general, decomposition-based strategies require explicit knowledge of the structure of the DAG off-line in order to apply the decomposition. In non-decomposition-based strategies, the program can unfold dynamically since no off-line knowledge is required.

For decomposed strategy, most prior work considers synchronous tasks (subcategory of general DAGs) with implicit deadlines. Lakshmanan et al. [31] proved a capacity augmentation bound of $3.42$ for partitioned fixed-priority scheduling for a restricted category of synchronous tasks$^3$ by decomposing tasks and scheduling the decomposed tasks using a deadline monotonic scheduling strategy. Saifullah et al. [42] provide a different decomposition strategy for general parallel synchronous tasks and prove a capacity augmentation bound of $4$ when the decomposed tasks are scheduled using global EDF and $5$ when they are scheduled using partitioned DM. Kim et al. [30] provide a different decomposition strategy for these synchronous tasks and prove a capacity augmentation bound of $3.73$ using global deadline monotonic strategy. In the respective papers, these results are stated as resource augmentation bounds, but they are in fact the stronger capacity augmentation bounds. Nelisson et al. [39] proved a resource augmentation bound of $2$ for general synchronous tasks.

For non-decomposition strategies, researchers have studied primarily global earliest deadline first (G-EDF) and global rate-monotonic (G-RM). Andersson and Niz [5] show that global EDF provides resource augmentation bound of $2$ for synchronous tasks with constrained deadlines. Both Li et al. [33] and Bonifaci et al. [16] concurrently showed that global EDF provides a resource augmentation bound of $2$ for general DAG tasks with arbitrary deadlines. In their paper, Bonifaci et al. also proved that G-RM provides a resource augmentation bound of $3$ for parallel DAG tasks with arbitrary deadlines. In addition,

$^2$ critical-path length of a sequential task is equal to its execution time
$^3$ Fork-join task model in their terminology
Agrawal et al also provide a capacity augmentation bound of 4 for global EDF for task sets with implicit deadlines.

In summary, the best known capacity augmentation bound for implicit deadlines tasks are 4 for DAG tasks using global EDF, and 3.73 for parallel synchronous tasks using decomposition combined with global DM. The contributions of this paper are as follows:

1. We improve the capacity augmentation bound of global EDF to \( \frac{3+\sqrt{5}}{2} \approx 2.6181 \) for DAGs. When the number of processors, \( m \), is large, there is a matching lower bound for global EDF due to [33]; therefore, this result closes the gap for large \( m \). In addition, this is the best known capacity augmentation bound for any scheduler for parallel DAG tasks.

2. We show that global RM has a capacity augmentation bound of \( 2 + \sqrt{3} \approx 3.7321 \). This is the best known capacity augmentation bound for any fixed-priority scheduler for DAG tasks. Even if we restrict ourselves to synchronous tasks, this is the best bound for global fixed priority scheduling without decomposition.

3. For global EDF and global RM, we also present their resource augmentation factor as a function of the utilization and the maximum critical path utilization. Moreover, we also present utilization-based schedulability analysis tests based on the utilization and the maximum critical path utilization.

The paper is organized as follows. Section 2 defines the DAG model for parallel tasks and provides some definitions. Section 3 presents a canonical form to give an upper bound of the work of a DAG that should be done in a specified interval length. Section 4 proves that global EDF provides the capacity augmentation bound of 2.6181. Section 5 shows that global RM provides the capacity augmentation bound of 3.7321. Section 7 concludes this paper.
We now present the details of the DAG task model for parallel tasks and some additional definitions.

**Task Model** This paper considers a given set $T$ of independent sporadic real-time tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$. A task $\tau_i$ represents an infinite sequence of arrivals and executions of task instances (or also called jobs). We consider the sporadic task model [38, 11] where, for a task $\tau_i$, the minimum inter-arrival time or period $T_i$ represents the time between consecutive arrivals of task instances, and the relative deadline $D_i$ represents the temporal constraint for executing the job. If a task instance of $\tau_i$ arrives at time $t$, the execution of this instance must be finished no later than the absolute deadline $t + D_i$ and the release of the next instance of task $\tau_i$ must be no earlier than $t$ plus the minimum inter-arrival time, i.e., $t + T_i$. In this paper, we consider *implicit deadline tasks* where each task $\tau_i$’s relative deadline $D_i$ is equal to its minimum inter-arrival time $T_i$; that is, $T_i = D_i$.

Each task $\tau_i \in T$ is a parallel task; we consider a general model for deterministic parallel tasks, namely the DAG model. Each task is characterized by its execution pattern, defined by a directed acyclic graph (DAG). Each node (subtask) in the DAG represents a sequence of instructions (a thread) and each edge represents dependency between nodes. A node (subtask) is ready to be executed when all its predecessors have been executed. Throughout this paper, as it is not necessary to build the analysis based on specific structures of the execution pattern, only two parameters related to the execution pattern of task $\tau_i$ are defined:

- *total execution time (or work) $C_i$* of task $\tau_i$: This is the summation of the worst-case execution times of all the subtasks of task $\tau_i$.
- *critical-path length $\Phi_i$* of task $\tau_i$: This is the length of the critical path in the given DAG, in which each node is characterized by the worst-case execution time of the corresponding subtask of task $\tau_i$; critical path length is the worst case execution time of the task on an infinite number of processors.

Given a DAG, obtaining work $C_i$ and the critical-path length $\Phi_i$ [43, pages 661-666] can both be done in linear time.

For notational brevity, the *utilization* $\frac{C_i}{T_i}$ of task $\tau_i$ is denoted by $u_i$. The total utilization of the task set is $U_{\sum} = \sum_{\tau_i \in T} u_i$. Moreover, let the *critical path utilization* of task $\tau_i$, denoted as $\Delta_i$, be $\frac{\Phi_i}{T_i}$. Also, let $\Delta_{\max}$ is the maximum critical path utilization of task set $T$, i.e., $\Delta_{\max} = \max_{\tau_i \in T} \Delta_i$. Finally, we also define $V_i$ as $\Delta_{\max} \cdot T_i$.

**Processor Model and Global Scheduling** This paper considers scheduling a task set on a uniform multiprocessor or multicore consisting of $m$
identical processors or cores. Specifically, we only consider global policies in which an instance of a subtask/task can be migrated among the m processors. In global scheduling, there is a global queue for the subjobs (subtask instances) that are ready to be executed.

This paper will explore two global scheduling policies, global earliest-deadline-first (global EDF, or G-EDF) and global rate monotonic (global RM, or G-RM), to decide the priority orders of the subjobs in the global queue. In G-EDF, a subjob has higher priority than another if its absolute deadline of the job that contains it is earlier. In G-RM, a subjob of a task $\tau_i$ has higher priority than another subjob of a task $\tau_j$ if $T_i \leq T_j$. For the simplicity of presentation, it is assumed that the tasks are sorted by increasing deadline so that $T_i \leq T_j$ if $i \leq j$.

**Utilization-based schedulability test**  As mentioned in Section 1 we analyze algorithms in terms of their capacity augmentation bound. The formal definition is presented here:

**Definition 2.0.1** Given a task set $\mathbf{T}$ with total utilization of $U_\Sigma$, a scheduling algorithm $A$ with capacity augmentation bound $b$ can always schedule this task set on $m$ processors of speed $b$ as long as $T$ satisfies the following conditions on speed $t$ processors.

\begin{align}
\text{Utilization does not exceed total cores, } \sum_{\tau_i \in \mathbf{T}} u_i & \leq m \quad (1) \\
\text{For each task } \tau_i \in \mathbf{T}, \text{ the critical path } \Phi_i & \leq D_i \quad (2)
\end{align}

Since no scheduler can schedule a task set $\mathbf{T}$ on $m$ unit speed processors unless Conditions (1) and (2) are met, capacity augmentation bound automatically leads to a resource augmentation bound. This definition can be equivalently stated (without reference to the speedup factor) as follows: Condition (1) says that the total utilization $U_\Sigma$ is at most $m/b$ and Condition (2) says that the critical-path length of each task is at most $1/b$ times its relative deadline, that is, $\Delta_{\max} \leq b$. Therefore, in order to check if a task set is schedulable we only need to know the sum of the task utilizations, and the maximum critical path utilization. Note that a scheduler with a smaller $b$ is better than another with a larger $b$, since $b = 1$ means that $A$ is an optimal scheduler.
In this section, we will represent each task \( \tau_i \) using its canonical form DAG. Note each task can have an arbitrarily complex DAG structure which is difficult to analyze and may not even be known to the scheduler before runtime. However, given the known task set parameters (work, critical path length, utilization, critical-path utilization, etc.) we represent each task using a canonical DAG which allows us to upper bound the demand of the task in any given interval length \( t \). These results will play important building blocks when we analyze the capacity augmentation bounds for global EDF in Section 4 and global RM in Section 5.

We first classify each task \( \tau_i \) as a light or heavy task. A task is a light task if its utilization \( u_i = C_i / T_i \leq \Delta_{\text{max}} \). Otherwise, if \( \tau_i \)'s utilization \( u_i > \Delta_{\text{max}} \), then we say that \( \tau_i \) is heavy.

For analytical purposes, instead of considering the complex DAG structure of individual tasks \( \tau_i \), we consider a canonical form \( \tau_i^* \) of task \( \tau_i \). The canonical form of a task is also represented by a DAG, but it is a much simpler DAG. In particular, each (node) subtask of task \( \tau_i^* \) has execution time \( \epsilon \), which is positive and arbitrarily small. (For presentation clarity, we assume that \( V_i \epsilon \) and \( C_i \epsilon \) are both integers.) Light and heavy tasks have different canonical forms described below.

- The canonical form \( \tau_i^* \) of a light task \( \tau_i \) is simply a chain of \( C_i / \epsilon \) nodes, each with execution time \( \epsilon \). Note that task \( \tau_i^* \) is a sequential task.

- The canonical form \( \tau_i^* \) of a heavy task \( \tau_i \) is a little more complex. It starts with a chain of \( V_i / \epsilon - 1 \) nodes each with execution time \( \epsilon \). Therefore, the total work of this chain is \( V_i - \epsilon \). The last node of the chain forks all the remaining nodes. That is, all the remaining \( (C_i - V_i + \epsilon) / \epsilon \) nodes have an edge from the last node of this chain, but no other edges. Therefore, all these forked subtasks can execute entirely in parallel.

Due to the assumption that \( V_i \epsilon \) and \( C_i \epsilon \) are both integers for each task \( \tau_i \), we know that the above construction of the canonical form \( \tau_i^* \) results in a feasible DAG structure. Figure 1 provides an example for such a transformation for a heavy task. It is important to note that the canonical form \( \tau_i^* \) does not depend on the internal DAG structure of \( \tau_i \) at all. On the other hand, it depends only on the task parameters of task \( \tau_i \) and the maximum critical path utilization \( \Delta_{\text{max}} \) of the task set, since \( V_i = \Delta_{\text{max}} \cdot T_i \).

As an additional analysis tool, we define a hypothetical scheduling \( A_{\infty} \) which must schedule a task set \( T \) on an infinite number of processors, that is, \( m = \infty \). Since the system has an infinite number of processors, the prioritization of the subjobs becomes unnecessary and \( A \) can obtain an optimal schedule by simply assigning a subjob to a processor as soon as that subjob becomes ready for execution. Using this schedule, all the tasks respond in their critical-path length, that is, a job of task \( \tau_i \) finishes exactly \( \Phi_i \) time units after it is released.
Therefore, if $\Phi_i \leq T_i$ for all tasks $\tau_i$ in $T$, the above schedule always meets the deadlines of the tasks. We denote this schedule as $S_{\infty}$. Similarly, $S_{\infty, \alpha}$ is the resulting schedule when $A_{\infty}$ schedules tasks on processors of speed $\alpha \geq 1$. Note that $S_{\infty, \alpha}$ finishes a job of task $\tau_i$ exactly $\Phi_i/\alpha$ time units after it is released.

We now define some notations based on $S_{\infty, \alpha}$. We say that maximum load $\text{work}_i(t, \alpha)$ for task $i$ as the maximum amount of work that $S_{\infty, \alpha}$ may do on the subjobs of $\tau_i$ in any interval of length $t$. Let $q_i(t, \alpha)$ be the total work finished by $S_{\infty, \alpha}$ between the arrival time $r_i$ of task instance $\tau_i$ and time $r_i + t$. That is, from $r_i + t$ to $r_i + T_i$, i.e., interval length $T_i - t$, the remaining $C_i - q_i(t, \alpha)$ workload (execution time) has to be finished. Clearly, both $q_i(t, \alpha)$ and $\text{work}_i(t, \alpha)$ depend on the structure of the DAG. We can derive $\text{work}_i(t, \alpha)$ as follows:

$$\text{work}_i(t, \alpha) =
\begin{cases}
    C_i - q_i(T_i - t, \alpha) & t \leq T_i \\
    \left[\frac{1}{T_i}\right] C_i + \text{work}_i(t - \left[\frac{1}{T_i}\right] T_i) & t > T_i.
\end{cases}
$$

(3)

For the canonical form $\tau^*_i$, let $q^*_i(t, \alpha)$ be defined with the same definition of $q_i(t, \alpha)$ for task $\tau_i$. As the canonical form in task $\tau^*_i$ is well-defined, we can derive $q^*_i(t, \alpha)$ directly. Note that $\epsilon$ can be arbitrarily small, and, hence, its impact is ignored when calculating $q^*_i(t, \alpha)$. Suppose that $V'_i$ is $\min[C_i, V_i]$. For both heavy and light tasks, the first $V'_i$ unit of work is sequential. Therefore, it is clear that $q^*_i(t, \alpha)$ is $\alpha \cdot t$ when $t < V'_i/\alpha$. Moreover, if task $\tau_i$ is a heavy task, we also have $q^*_i(V'_i/\alpha, \alpha) = C_i$.

We can now define the canonical workload $\text{work}^*_i(t, \alpha)$ as the maximum workload of the canonical task $\tau^*_i$ in any time interval length in schedule $S_{\infty, \alpha}$. For a light task $\tau_i$, where $C_i/T_i \leq \Delta_i$, and $\tau^*_i$ is a chain, it is easy to see that the canonical workload is:

$$\text{work}^*_i(t, \alpha) =
\begin{cases}
    0 & t < T_i - \frac{C_i}{\alpha} \\
    \alpha(t - (T_i - \frac{C_i}{\alpha})) & T_i - \frac{C_i}{\alpha} \leq t \leq T_i \\
    \left[\frac{1}{T_i}\right] C_i + \text{work}^*_i(t - \left[\frac{1}{T_i}\right] T_i) & t > T_i.
\end{cases}
$$

(4)
Similarly, for heavy tasks, where \( C_i/T_i \geq \Delta_i \), when \( \epsilon \) is arbitrarily small, we have

\[
\text{work}^*_i(t, \alpha) =
\begin{cases} 
0, & t < T_i - \frac{V_i}{\alpha} \\
N - V_i + \alpha(t - (T_i - \frac{V_i}{\alpha})), & T_i - \frac{V_i}{\alpha} < t \leq T_i \\
\left\lfloor \frac{t}{T_i} \right\rfloor C_i + \text{work}^*_i(t - \left\lfloor \frac{t}{T_i} \right\rfloor T_i), & t > T_i.
\end{cases}
\]

\[(5)\]

Figure 2 provides an illustrative example for demonstrating \( q^*_i(t, \alpha), q_i(t, \alpha), \text{work}^*_i(t, \alpha), \) and \( \text{work}_i(t, \alpha) \) of the heavy task \( \tau_i \) defined in Figure 1 when \( T_i = 20, \alpha = 1 \), and \( \alpha = 2 \). It can be observed that \( \text{work}^*_i(t, \alpha) \geq \text{work}_i(t, \alpha) \) in this example. In fact, the following lemma proves that \( \text{work}^*_i(t, \alpha) \geq \text{work}_i(t, \alpha) \) for any \( t > 0 \) and \( \alpha \geq 1 \).
Lemma 1 For any $t > 0$ and $\alpha \geq 1$,
\[
    \text{work}^*_i(t, \alpha) \geq \text{work}_i(t, \alpha).
\]

Proof. Suppose that $V'_i$ is $\min\{C_i, V_i\}$. For both heavy and light tasks, the first $V'_i$ unit of work is sequential. Therefore, it is clear that $q'_i(t, \alpha)$ is $\alpha \cdot t$ when $t < \frac{V'_i}{\alpha}$. According to the definition, we have $q_i(t, \alpha) \geq \alpha \cdot t \geq q'_i(t, \alpha)$ when $t < \frac{V'_i}{\alpha}$.

In addition, $q'_i(V'_i, \alpha) = C_i$, since at $t = V'_i/\alpha$, $S_{\infty, \alpha}$ finishes the job since $\Phi_i = V'_i$ for all $\tau_i$. Moreover, since the critical path length $\Phi_i \leq V_i$, for all $i$, we have $q_i(V_i, \alpha) = C_i$, since $S_{\infty, \alpha}$ finishes a job of task $\tau_i$ exactly $\Phi_i/\alpha$ time units after it is released. Since $\Phi_i \leq V'_i$, we can conclude that $q'_i(t) \leq q_i(t)$ for any $0 \leq t < T_i$. Combining this fact with the definition of work$(t, \alpha)$ (Equation (3)), we complete the proof.

Moreover, the following lemma provides an upper bound of the density (workload that has to be finished divided by the interval length) for both heavy and light tasks.

Lemma 2 For any task $\tau_i$ (heavy or light), we have
\[
    \frac{\text{work}_i(t, \alpha)}{t} \leq \frac{\text{work}^*_i(t, \alpha)}{t} \leq \frac{u_i}{1 - \frac{\Delta_{\max}}{\alpha}} \quad (6)
\]
for any $t > 0$ and $\alpha > 1$.

Proof. The first inequality in Inequality (6) comes from Lemma 1. We prove this lemma by showing that the second inequality in Inequality (6) holds for light tasks and heavy tasks. We first note that the right hand side is non-negative; that is, $1 - \frac{\Delta_{\max}}{\alpha} > 0$, since $\alpha > 1$, and $0 < \Delta_{\max} \leq 1$. There are two cases:

Case 1: $0 < t \leq T_i$ — We further consider light and heavy tasks separately as follows:

- If $\tau_i$ is a light task, where the function $\text{work}^*_i(t, \alpha)$ is defined in Equation (4): Therefore, for any $0 < t \leq T_i$, we have
  \[
  \text{work}^*_i(t, \alpha) - \frac{C_i}{T_i} \cdot t \leq \alpha(t - T_i + \frac{C_i}{\alpha}) - \frac{C_i}{T_i} \cdot t \\
  = (t - T_i)(\alpha - \frac{C_i}{T_i}) \\
  \leq 1 \cdot (t - T_i)(\alpha - 1) \\
  \leq 0,
  \]
  where we get the step $\leq 1$ by relying on three assumptions: (a) $t \leq T_i$; (b) since $\tau_i$ is a light task, we have $\frac{C_i}{T_i} \leq \Delta_{\max} \leq 1$; and (c) $\alpha > 1$.

As we observed above, the RHS of Inequality (6) is non-negative. Therefore, the said inequality holds for any light task $\tau_i$ for any $0 < t \leq T_i$.

- If $\tau_i$ is a heavy task, where the function $\text{work}^*_i(t, \alpha)$ is defined in Equation (5). Inequality (6) holds naturally when $0 < t < T_i - \frac{V_i}{\alpha}$.
since the left hand side is 0 and right hand side is non-negative. For any $t$ such that $T_i - \frac{V_i}{\alpha} \leq t \leq T_i$, we have

$$\frac{\text{work}_i^*(t, \alpha)}{t} = \frac{C_i + \alpha t - \alpha T_i}{t} = \alpha + T_i \left( \frac{u_i - \alpha}{t} \right).$$

Therefore, $\frac{\text{work}_i^*(t, \alpha)}{t}$ is maximized either (a) when $t = T_i - \frac{V_i}{\alpha}$ if $u_i - \alpha \geq 0$ or (b) $t = T_i$ if $u_i - \alpha < 0$. If (b) is true, Inequality (6) is obvious since we know that $\text{work}_i^*(T_i, \alpha) = u_i \leq \frac{u_i}{1 - \frac{\Delta_{\text{max}}}{\alpha}}$. If (a) is true, then we have

$$\frac{\text{work}_i^*(T_i - \frac{V_i}{\alpha}, \alpha)}{T_i - \frac{V_i}{\alpha}} = \frac{C_i - V_i}{T_i - \frac{V_i}{\alpha}} = \frac{C_i - T_i \Delta_{\text{max}}}{T_i - \frac{T_i \Delta_{\text{max}}}{\alpha}} = \frac{u_i - \Delta_{\text{max}}}{1 - \frac{\Delta_{\text{max}}}{\alpha}} \leq \frac{u_i}{1 - \frac{\Delta_{\text{max}}}{\alpha}}.$$

Therefore, Inequality (6) holds for any heavy task $\tau_i$ for any $0 < t \leq T_i$.

**Case 2:** $t > T_i$ — Suppose that $t$ is $kT_i + t'$, where $k = \left\lfloor \frac{t}{T_i} \right\rfloor$ and $0 < t' \leq T_i$. By Equation (4) and Equation (5) and the results known for $0 < t' \leq T_i$, we have

$$\frac{\text{work}_i^*(t, \alpha)}{t} = \frac{kC_i + \text{work}_i^*(t', \alpha)}{kT_i + t'} \leq \frac{ku_i T_i + \frac{u_i}{1 - \frac{\Delta_{\text{max}}}{\alpha}} t'}{kT_i + t'} \leq \frac{(kT_i + t') \cdot \frac{u_i}{1 - \frac{\Delta_{\text{max}}}{\alpha}}}{kT_i + t'} \leq \frac{u_i}{1 - \frac{\Delta_{\text{max}}}{\alpha}}.$$

We will use the result of this lemma in Sections 4 and 5 to derive bounds on global EDF and global RM scheduling.
GLOBAL EDF

In this section, we will use the results from Section 3 to prove the capacity augmentation bound of $\frac{3 - 1}{m} + \frac{\sqrt{5 - 2/m + 1/m^2}}{2} \approx (3 + \sqrt{5})/2$ for global EDF scheduling of parallel DAG tasks. For large $m$, this is tight, since Li et al. [33] showed that global EDF can not provide a capacity augmentation bound of less than $(3 + \sqrt{5})/2$ when $m$ is large. In addition, we also show a lower bound of $\frac{3 - 2/m + \sqrt{5 - 12/m + 4/m^2}}{2}$ when $m \geq 3$.

4.1 Upper Bound on Capacity Augmentation of G-EDF

Our analysis builds on the analysis used to prove the resource augmentation bounds by Bonifaci et al. [16]. We first review the particular lemma from the paper that we will use to achieve our bound.

Lemma 3 If

$$\forall t > 0, \quad (\alpha \cdot m - m + 1) \cdot t \geq \sum_{i=1}^{n} \text{work}_i(t, \alpha),$$

the task set is schedulable by G-EDF on a platform with speed $\alpha$.

Proof. This is based on a reformulation of Lemma 3 and Definition 10 in [16]. □

Theorem 1 The capacity augmentation bound for G-EDF is $\frac{3 - 1}{m} + \frac{\sqrt{5 - 2/m + 1/m^2}}{2}$.

Proof. According to Lemma 2, for any $\alpha > 1$, it is clear that

$$\sup_{t > 0} \sum_{\tau_i} \frac{\text{work}_i(t, \alpha)}{t} \leq \sum_{\tau_i} \sup_{t > 0} \frac{\text{work}_i^*(t, \alpha)}{t} \leq \frac{\sum_{\tau_i} u_i}{1 - \Delta_{\text{max}}/\alpha} \leq \frac{m}{1 - \frac{1}{\alpha}},$$

where sup is the supremum of a set of numbers.

Therefore, if

$$\frac{m}{1 - \frac{1}{\alpha}} \leq (\alpha \cdot m - m + 1),$$

we can also conclude that the schedulability test for G-EDF in Lemma 3 holds. Therefore, if $(\alpha \cdot m - m + 1) \cdot (1 - \frac{1}{\alpha}) \geq m$, then the task set is schedulable. In order to calculate $\alpha$, we can solve the equivalent quadratic equation

$$m\alpha^2 - (3m - 1)\alpha + (m - 1) = 0.$$
which solves to $\alpha = \frac{3 - \frac{1}{m} + \sqrt{5 - \frac{2}{m} + \frac{1}{m^2}}}{2}$. \qed

We now state a slightly more general corollary relating the resource augmentation bound to the total utilization.

**Corollary 1** The resource augmentation bound for G-EDF is

$$\Delta_{\text{max}} \leq \frac{U_{\Sigma} + 1 - \frac{1}{m} + \sqrt{(\Delta_{\text{max}} + \frac{U_{\Sigma}}{m} + 1 - \frac{1}{m})^2 - 4(1 - \frac{1}{m})\Delta_{\text{max}}}}{2}.\quad(8)$$

**Proof.** The proof is the same as in the proof of Theorem 1 without taking the inequality $\leq$ in (7). Therefore, if

$$\frac{U_{\Sigma}}{1 - \Delta_{\text{max}}} \leq (\alpha \cdot m - m + 1),$$

$$\Rightarrow m \alpha^2 - (m \Delta_{\text{max}} + U_{\Sigma} + m - 1)\alpha + (m - 1)\Delta_{\text{max}} \geq 0,$$  \quad(8)

we can also conclude that the schedulability test for G-EDF in Lemma 3 holds. By solving the above inequality, it can be proved that the inequality holds when

$$\alpha \geq \frac{\Delta_{\text{max}} + \frac{U_{\Sigma}}{m} + 1 - \frac{1}{m} + \sqrt{(\Delta_{\text{max}} + \frac{U_{\Sigma}}{m} + 1 - \frac{1}{m})^2 - 4(1 - \frac{1}{m})\Delta_{\text{max}}}}{2}.\quad\Box$$

Figure 3 illustrates the resource augmentation bound of G-EDF provided in Corollary 1 when $m$ is sufficiently large, i.e., $m = \infty$, by varying $\frac{U_{\Sigma}}{m}$ and $\Delta_{\text{max}}$. The previously known resource augmentation bound for EDF in such a case is $2^{16}$. As can be seen from the figure, this corollary provides a tighter resource augmentation bound for small values of $\frac{U_{\Sigma}}{m}$ and $\Delta_{\text{max}}$, but looser bound for larger values.
4.2 LOWER BOUND ON CAPACITY AUGMENTATION OF G-EDF

As mentioned above, Li et al.’s lower bound [33] demonstrates the tightness of the above bound for large m. We now provide the lower bound of the capacity augmentation bound for small m.

Consider a task set T with two tasks, \( \tau_1 \) and \( \tau_2 \). Task \( \tau_1 \) starts with sequential execution for \( 1 - \epsilon \) amount of time and then forks \( \frac{m-\epsilon}{\epsilon} + 1 \) subtasks with execution time \( \epsilon \). Here, \( \epsilon \) is assumed to be a positive number that is very small and \( \frac{1}{\epsilon m} \) is assumed to be a positive integer. Therefore, the total work of task \( \tau_1 \) is \( C_1 = m - 1 \) and its critical-path length \( \Phi_i = 1 \). The minimum inter-arrival time of task \( \tau_1 \) is 1.

Task \( \tau_2 \) is simply a sequential task with work/execution time of \( 1 - \frac{1}{\alpha} \) and minimum inter-arrival time also \( 1 - \frac{1}{\alpha} \), where \( \alpha > 1 \) will be defined later.

Clearly, the total utilization is \( m \) and the critical-path length of each task is at most the relative deadline (minimum inter-arrival time).

**Lemma 4** When \( \alpha < \frac{3 - \frac{2}{m} - \delta + \sqrt{5 - \frac{14}{m} + \frac{1}{\alpha^2}}}{2} \) and \( \delta = 2\epsilon - g(\epsilon) \) and \( m \geq 3 \),

\[
1 - \frac{2\epsilon}{\alpha} + m - 2 \frac{2}{m\alpha} > 1 - \frac{1}{\alpha}.
\]

**Proof.** By solving

\[
1 - \frac{2\epsilon}{\alpha} + m - 2 \frac{2}{m\alpha} = 1 - \frac{1}{\alpha},
\]

we know that the equality holds when \( \alpha \) is equal to \( \frac{3 - \frac{2}{m} - 2\epsilon + \sqrt{5 - \frac{14}{m} + \frac{1}{\alpha^2}} - g(\epsilon)}{2} \), in which \( g(\epsilon) \) is a function of \( \frac{1}{\epsilon} \) and approaches to 0 when \( \epsilon \) approaches 0.

Therefore, when \( \alpha < \frac{3 - \frac{2}{m} - 2\epsilon + \sqrt{5 - \frac{14}{m} + \frac{1}{\alpha^2}} - g(\epsilon)}{2} \), it is clear that \( \frac{1 - \frac{\epsilon}{\alpha}}{\alpha} + m - 2 \frac{2}{m\alpha} > 1 - \frac{1}{\alpha} \). Now, by setting \( \delta \) to \( 2\epsilon \), we reach the conclusion. \( \Box \)

**Theorem 2** The capacity augmentation bound for G-EDF is at least \( \frac{3 - \frac{2}{m} + \sqrt{5 - \frac{14}{m} + \frac{1}{\alpha^2}}}{2} \), when \( \epsilon \to + 0 \).

**Proof.** Consider the system with two tasks \( \tau_1 \) and \( \tau_2 \) defined at the beginning of Section 4.2. Suppose that the arrival time of task \( \tau_1 \) is at time 0, and the arrival time of task \( \tau_2 \) is at time \( 1 - \frac{1}{\alpha} + \frac{\epsilon}{\alpha} \). By definition, the first jobs of \( \tau_1 \) and \( \tau_2 \) have absolute deadlines at 1 and \( 1 + \frac{\epsilon}{\alpha} \). Therefore, G-EDF scheduling will execute the sequential execution of task \( \tau_1 \), and then execute the sub-jobs of task \( \tau_1 \), and then execute \( \tau_2 \).

The finishing time of task \( \tau_1 \) by running at speed \( \alpha \) is not earlier than

\[
1 - \frac{\epsilon}{\alpha} + m - 2 \frac{2\epsilon}{m\alpha} = 1 - \frac{\epsilon}{\alpha} + m - 2 \frac{2}{m\alpha}.
\]

Therefore, the finishing time of task \( \tau_2 \) is not earlier than

\[
1 - \frac{\epsilon}{\alpha} + m - 2 \frac{2}{m\alpha} + 1 - \frac{1}{\alpha}.
\]
If $\frac{1-\epsilon}{\alpha} + \frac{m-2}{m\alpha} + \frac{1-\frac{1}{\alpha}}{\alpha} > 1 + \frac{\epsilon}{\alpha}$, then we know that task $\tau_2$ misses its deadline. By Lemma 4, we reach the conclusion.

Figure 4 illustrates the upper bound of G-EDF provided in Theorem 1 and the lower bound in Theorem 2 with respect to the capacity augmentation bound. It can be easily seen that the upper and lower bounds are getting closer when $m$ is larger. When $m$ is 100, the gap between the upper and the lower bounds is roughly about 0.01622.

4.3 Utilization-Based Schedulability Test

We conclude this section by extending the above analysis to provide a utilization-based schedulability test based on $\Delta_{max}$ and $U_{\sum}$ in the following corollary.

**Corollary 2** If

$$U_{\sum} \leq \frac{m}{1-\frac{\Delta_{max}}{\alpha} + 1 - \frac{1}{\alpha}}$$

then the task set can be feasibly scheduled by using G-EDF.

**Proof.** This is proved by performing the schedulability test at the platform with speed $\alpha$. In that platform with speed $\alpha$, the critical path utilization among the tasks is at most $\frac{\Delta_{max}}{\alpha}$ and the total utilization is $\frac{U_{\sum}}{\alpha}$. The following proof analyzes the speed-up factor $\alpha$ that can guarantee the schedulability at a platform with speed $\alpha$ by assuming that $\frac{\Delta_{max}}{\alpha}$ is given as an input $Y$. 
Similar to the proof of Theorem 1 without taking the condition $\Delta_{\text{max}} \leq 1$ in the inequality $\leq 1$ in (7), we know that if

$$\frac{m}{1 - \frac{\Delta_{\text{max}}}{\alpha}} \leq (\alpha \cdot m - m + 1),$$

$$\Rightarrow \alpha \geq \frac{1}{1 - \frac{1}{m}} + 1 - \frac{1}{m},$$

we can also conclude that the schedulability test for G-EDF in Lemma 3 holds at the platform with speed $\alpha$. This implies that when the total utilization at speed $\alpha$ is no more than $\frac{m}{\frac{m}{\pi} + \frac{1}{\pi}}$, we can guarantee the schedulability of this task set at a platform with speed $\alpha$.

By the above argument, if the platform with speed $\alpha$ is the platform that we would like to test the schedulability of G-EDF, we already reach the conclusion.
This section presents the proof that G-RM provides a capacity augmentation bound of $2 - \frac{1}{2m} + \sqrt{3 - \frac{1}{m} + \frac{1}{4m^2}} \approx 2 + \sqrt{3}$ for large $m$. The structure of the proof is very similar to the analysis in Section 4.

Again, we use a lemma from [16], restated below.

**Lemma 5** If

$$\forall t > 0, \quad 0.5 \cdot (\alpha \cdot m - m + 1) \cdot t \geq \sum_{i=1}^{n} \text{work}_i(t, \alpha),$$

the task set is schedulable by G-RM on a platform with speed $\alpha$.

**Proof.** This is based on a reformulation of Lemma 6 and Definition 10 in [16]. Note that that analysis in [16] is for deadline-monotonic scheduling, by giving a subjob of a task higher priority if its relative deadline is shorter. As we consider task sets with implicit deadlines, deadline-monotonic scheduling is the same as rate-monotonic scheduling.

By using Lemma 5, with similar proofs in Theorem 1 and Corollary 1 we have the following results.

**Theorem 3** The capacity augmentation bound for G-RM is $4 - \frac{1}{m} + \sqrt{12 - \frac{4}{m} + \frac{1}{m^2}}$.  

**Proof.** In the similar manner as the proof of Theorem 1, for any $\alpha > 1$, we know that

$$\sup_{t > 0} \frac{\sum_{i} \text{work}_i(t, \alpha)}{t} \leq \frac{m}{1 - \frac{1}{\alpha}}.$$ (11)

Therefore, if

$$\frac{m}{1 - \frac{1}{\alpha}} \leq 0.5(\alpha \cdot m - m + 1),$$

we can also conclude that the schedulability test for G-RM in Lemma 5 holds. By solving the inequality above, we know that $\frac{m}{1 - \frac{1}{\alpha}} \leq 0.5(\alpha \cdot m - m + 1)$ holds when $\alpha \geq \frac{4 - \frac{1}{m} + \sqrt{12 - \frac{4}{m} + \frac{1}{m^2}}}{2}$. □

The result in Theorem 3 is the best known result for the capacity augmentation bound for global fixed-priority scheduling for general DAG tasks with arbitrary structures. Interestingly, Kim et al. [30] get the same bound of $2 + \sqrt{3}$ for global fixed-priority scheduling of parallel synchronous tasks (a subset of DAG tasks).

The strategy used in [30] is quite different. In particular, in their algorithm, the tasks undergo a stretch transformation which generates a set of sequential subtask (each with its release time and deadline) for each parallel task $\tau_i$ in the
original task set. These subtask are then scheduled using a global DM scheduling algorithm [13]. Note that even though the parallel tasks in the original task set are implicit deadline tasks, the transformed sequential tasks are only constrained deadline tasks — hence the need for deadline monotonic scheduling instead of rate monotonic scheduling.

The following, slightly more general, corollary relates the resource augmentation bound to the utilization.

**Corollary 3** The resource augmentation bound for G-RM is

$$\Delta_{\text{max}} + \frac{2U_{\Sigma}}{m} + 1 - \frac{1}{m} + \frac{\sqrt{\left(\Delta_{\text{max}} + \frac{2U_{\Sigma}}{m} + 1 - \frac{1}{m}\right)^2 - 4\left(1 - \frac{1}{m}\right)\Delta_{\text{max}}}}{2}.$$

**Proof.** The proof is the same as in the proof of Corollary 1. Therefore, if

$$\frac{U_{\Sigma}}{1 - \frac{\Delta_{\text{max}}}{\alpha}} \leq 0.5(\alpha \cdot m - m + 1),$$

$$\Rightarrow m\alpha^2 - (m\Delta_{\text{max}} + 2U_{\Sigma} + m - 1)\alpha + (m - 1)\Delta_{\text{max}} \geq 0,$$

we can also conclude that the schedulability test for G-EDF in Lemma 3 holds. By solving the above inequality, it can be proved that the inequality holds when

$$\alpha \geq \frac{\Delta_{\text{max}} + \frac{2U_{\Sigma}}{m} + 1 - \frac{1}{m} + \frac{\sqrt{\left(\Delta_{\text{max}} + \frac{2U_{\Sigma}}{m} + 1 - \frac{1}{m}\right)^2 - 4\left(1 - \frac{1}{m}\right)\Delta_{\text{max}}}}{2}}{2}.$$

Figure 6 illustrates the resource augmentation bound of G-RM provided in Corollary 3 when $m$ is sufficiently large, i.e., $m = \infty$, by varying $\frac{U_{\Sigma}}{m}$ and $\Delta_{\text{max}}$. The previously known resource augmentation bound for global RM is 3 [16]. As can be seen from the figure, this corollary provides a tighter resource augmentation bound for small values of $\frac{U_{\Sigma}}{m}$ and $\Delta_{\text{max}}$, but looser bound for larger values.

**Corollary 4** If

$$U_{\Sigma} \leq \frac{m}{1 - \Delta_{\text{max}} + 1 - \frac{1}{m}},$$

then the task set can be feasibly scheduled by using G-RM.

**Proof.** The proof is identical to the proof of Corollary 2 by following Theorem 3 instead of Theorem 1. □
Figure 6: The resource augmentation bound of G-RM when $m$ is sufficiently large.

Figure 7: The utilization bound schedulability test for G-RM with respect to given $\Delta_{\text{max}}$. 
In this section, we review closely related work on real-time scheduling, concentrating primarily on scheduling task sets with parallel tasks.

Real-time multiprocessor scheduling considers scheduling sequential tasks on computers with multiple processors or cores and has been studied extensively (see [22, 12] for a survey). In addition, platforms such as LitmusRT [19, 17] have been designed to support these task sets. Here, we review a few relevant theoretical results. Researchers have proven both resource augmentation bounds, utilization bounds and capacity augmentation bounds. The best known utilization bound for global EDF for sequential tasks on a multiprocessor is 2 (traditionally stated as $1/b = 50\%$)[9]; therefore, global EDF trivially provides a resource and capacity augmentation bound of 2 as well. Partitioned EDF and versions partitioned static priority schedulers also provide a utilization bound of 2 [35, 6]. Global RM provides a capacity augmentation bound of 3 [4] to implicit deadline tasks.

For parallel real-time tasks, most early work considered intra-task parallelism of limited task models such as malleable tasks [32, 21, 29] and moldable tasks [37]. Kato et al. [29] studied the Gang EDF scheduling of moldable parallel task systems.

Researchers have since considered more realistic task models that represent programs that are typically generated by commonly used general purpose parallel programming languages such as Cilk family [1, 14], OpenMP [2], and Intel’s Thread Building Blocks [41]. These languages and libraries generally support primitives such as parallel-for loops and fork/join or spawn/sync in order to expose parallelism within the programs. Using these constructs in various combinations generates tasks whose structure can be represented with different types of DAGs.

Tasks with one particular structure, namely parallel synchronous tasks, have been studied more than others in the real-time community. These tasks are generated if only we use only parallel-for loops to generate parallelism. Laksmanan et al. [31] proved a (capacity) augmentation bound of 3,42 for a restricted synchronous task model which is generated when we restrict each parallel-for loop in a task to have the same number of iterations. General synchronous tasks (with no restriction on the number of iterations in the parallel-for loops), have also been studied [42, 30, 39, 5]. (More details on these results were presented in Section 1) Chwa et al. [20] provide a response time analysis.

If we do not restrict the primitives used to parallel-for loops, we get a more general task model — most easily represented by a general directed acyclic graph. A resource augmentation bound of $2 - \frac{1}{m}$ for G-EDF was proved for a single DAG with arbitrary deadlines [10] and for multiple DAGs [16, 33]. A capacity augmentation bound of $4 - \frac{1}{m}$ was proved in [33] for tasks with for implicit deadlines. Liu and Anderson [34] provide a response time analysis for G-EDF.
There has been significant work on scheduling parallel systems in the non-real time context \cite{40, 24, 23, 7, 8, 25}. In this context, the goal is generally to maximize throughput; tasks have no deadlines or periods. Various provably good scheduling strategies, such as list scheduling \cite{27, 18} and work-stealing \cite{15} have been designed. In addition, many platforms have been built based on these results: examples include parallel languages and runtime systems, such as the Cilk family \cite{1, 14}, OpenMP \cite{2}, and Intel’s Thread Building Blocks \cite{41}. While multiple tasks on a single platform have been considered in the context of fairness in resource allocation \cite{3}, none of this work considers real-time constraints.
In this paper, we consider parallel tasks in the DAG model and prove that global EDF provides a capacity augmentation bound of \( \frac{3 - 1/m + \sqrt{5} - 2/m + 1/m^2}{2} \leq (3 + \sqrt{5})/2 \approx 2.618 \) to parallel tasks with implicit deadlines and global RM provides the capacity augmentation bound of \( 2 - \frac{1}{2m} + \sqrt{3 - \frac{1}{m} + \frac{1}{4m^2}} \leq 2 + \sqrt{3} \approx 3.73 \) to these tasks. These are the best known bounds for these schedulers for DAG tasks. In addition, for a large enough number of processors \( m \), the global EDF bound of \((3 + \sqrt{5})/2\) is tight, since there exists a matching lower bound. Moreover, the global EDF bound of \((3 + \sqrt{5})/2\) is the best known capacity augmentation bound for any scheduler for parallel tasks. For global EDF and global RM, we also present utilization-based schedulability analysis tests based on the utilization and the maximum critical path utilization.

There are several directions of future work. Global RM capacity augmentation bound is not known to be tight. The current lower bound of the capacity augmentation bound of G-RM is \( 1/0.37482 \approx 2.668 \), inherited from the sequential sporadic real-time tasks without DAG structures [36]. Therefore, it is worth investigating a matching lower bound or lowering the upper bound. In addition, it would be interesting to investigate if the lower bound of 2.618 for global EDF is a general lower bound for any parallel scheduler or if it is possible to design schedulers that beat this bound. Finally, all the known capacity augmentation bound results are restricted to implicit deadline tasks; it would be interesting to generalize capacity augmentation bounds for constrained and arbitrary deadline tasks.
BIBLIOGRAPHY


