Response Time Analysis for G-EDF and G-DM Scheduling of Sporadic DAG-Tasks with Arbitrary Deadline [*]

Andrea Parri
Scuola Superiore Sant'Anna
Pisa, Italy
andrea.parri@sssup.it

Alessandro Biondi
Scuola Superiore Sant'Anna
Pisa, Italy
alessandro.biondi@sssup.it

Mauro Marinoni
Scuola Superiore Sant'Anna
Pisa, Italy
m.marinoni@sssup.it

ABSTRACT

New programming models have been proposed to exploit the parallelism of modern computing architectures. Also in the real-time domain more detailed task models are under evaluation to provide a tighter analysis of parallel application with precedence and timing constraints.

This paper presents two schedulability tests based on Response Time Analysis for determining whether a set of sporadic DAG-tasks with arbitrary deadlines can be scheduled by G-EDF or G-DM on a platform consisting of m identical processor. The first test is a simple polynomial time test, while the second one is a pseudo-polynomial time test. Our tests exploit the combinatorial properties of the DAGs by considering the interference experienced by each vertex. We describe a set of simulations showing that our tests outperform the tests described in [7] in terms of schedulability ratio and running time. We also provide resource augmentation bounds for our polynomial time test when considering single-DAG systems.

1. INTRODUCTION

High-performance computing, cloud-based technologies, and real-time embedded systems are only some of the areas where execution parallelism is a crucial factor that can impact performance. In the embedded domain, the presence of stringent timing constraints poses new challenges in the development of predictable and efficient systems that must be compliant with the standards. In many web-based applications, the system can handle multiple requests at the same time, each one characterized by a deadline imposed by the requirement of guaranteeing to the clients a bounded response time.

Providing a schedulability analysis for these kind of systems requires to accurately model the intrinsic parallelism within the task and to consider the dependency constraints between different parts of the task. Several models have been proposed in the literature to represent recurrent real-time tasks (e.g., [18, 25, 5, 10, 17, 4]). In this paper, we study the *sporadic DAG model* introduced in [5] and [24]. This model describes every jobs of a recurrent sporadic task with a Directed Acyclic Graph (DAG). Each vertex in this graph represents a sequence of instructions to be sequentially executed, while each edge represents an execution precedence constraint: vertices that are not related by an edge can be executed in parallel; on the other hand, vertices which are related by an edge must respect the sequential order imposed by the DAG structure.

This work deals with the problem of performing a schedulability analysis of DAG-tasks with arbitrary deadline under both G-EDF and G-DM scheduling, proposing an approach based on Response Time Analysis (RTA). Following this approach, we estimate the response time of a task using the notion of interference; due to the complexity intrinsic in computing the exact interference, we provide upper-bounds characterized by a tractable complexity. However, contrary to previous works, our analysis explicitly considers the interference experienced by each component/vertex of the task, thus systematically exploiting its internal structure (i.e., the combinatorial properties of the corresponding DAG).

Related work. Most prior work considers a restricted class of task-sets than the general class of sporadic DAG-tasks with arbitrary deadline considered in this paper. We can broadly classify these results into three subcategories: (*i*) those based on resource augmentation, (*ii*) those based on capacity augmentation, and (*iii*) those based on RTA.

(*i*) Andersson and de Niz [1] proved a resource augmentation bound of 2 for G-EDF scheduling of synchronous (parallel) tasks (a subcategory of general DAGs) with constrained deadline; they also proposed a schedulability test "attaining" their bound. Nelissen et al. [19] proved a resource augmentation bound of 2 for a class of scheduling algorithms (including U-EDF, PD^2, DP-Wrap and LLREF) on synchronous tasks with constrained deadline. Baruah et al. [5] proved a resource augmentation bound of 2 for G-EDF scheduling of a single DAG-task with arbitrary deadline; they also proposed sufficient polynomial and pseudo-polynomial time schedulability tests attaining their bound. Bonifaci et al. [7] proved a resource augmentation bound of $2 - 1/m$ (resp., of $3 - 1/m$) for G-EDF (resp., for G-DM) scheduling of DAG-tasks with arbitrary deadline; they also proposed sufficient polynomial and pseudo-polynomial time schedulability tests attaining their bound. Baruah [3] proposed a sufficient pseudo-polynomial time test for G-EDF scheduling of DAG-tasks with constrained deadline, and showed that this test strictly dominates the schedulability test defined in [7]. Sai-

[*]This work has been partially supported by the FP7 JUNIPER project (FP7-ICT-2011.4.4), founded by the European Community under grant agreement n. 318763.

RTNS 2015, November 04-06, 2015, Lille, France
© 2015 ACM. ISBN 978-1-4503-3591-1/15/11…$15.00
DOI: http://dx.doi.org/10.1145/2834848.2834864

fullah et al. [23] proved a resource augmentation bound of 4 for G-EDF scheduling of decomposed DAG-tasks with implicit deadline and extended their result to non-preemptive G-EDF.

(*ii*) Lakshmanan et al. [12] proved a capacity augmentation bound of 3.42 for partitioned DM scheduling of fork-join tasks (a subcategory of synchronous tasks) with implicit deadline. Saifullah et al. [24] proved a capacity augmentation bound of 4 (resp., of 5) for G-EDF scheduling (resp., for partitioned DM scheduling) of synchronous tasks with implicit deadline. Kim et al. [11] proved a capacity augmentation bound of 3.73 for G-DM scheduling of synchronous tasks with implicit deadline. Li et al. [13] proved a capacity augmentation bound of $4 - 2/m$ for G-EDF scheduling of DAG-tasks with implicit deadline, and a resource augmentation bound of $2-1/m$ for G-EDF scheduling of DAG-tasks with arbitrary deadline. Li et al. [14] proved a capacity augmentation bound of 2.62 (resp., of 3.73) for G-EDF (resp., for G-DM) scheduling of DAG-tasks with implicit deadline.

(*iii*) Chwa et al. [8] proposed a RTA-based schedulability test for G-EDF scheduling of synchronous tasks with constrained deadline. Axer et al. [2] proposed a RTA-based schedulability analysis for fork-join tasks with arbitrary deadlines and applied it to the Romain framework. Qamhieh et al. [21] proposed a schedulability test for G-EDF scheduling of DAG-tasks with constrained deadline and a study of its sustainability. Maia et al. [15] proposed a RTA-based schedulability analysis for G-FP scheduling of synchronous tasks with constrained deadline. Nogueira et al. [20] proposed a scheduling approach for DAG-tasks combining resource reservation with work-stealing; this work addresses soft real-time guarantees. Qamhieh et al. [22] proposed a stretching algorithm for DAG-tasks with implicit deadline and proved a resource augmentation bound of 2.62 for G-EDF scheduling of stretched tasks.

Contributions of this paper. This work provides the following novel contributions:

1. it proposes a polynomial time schedulability test, based on Response Time Analysis, for G-EDF and G-DM scheduling of sporadic DAG-tasks with arbitrary deadline and arbitrary vertex utilization (i.e., vertices can have utilization greater than 1);[1]
2. it proposes a pseudo-polynomial time schedulability test, based on Response Time Analysis, for G-EDF and G-DM scheduling of sporadic DAG-tasks with arbitrary deadline and arbitrary vertex utilization;
3. it presents a set of simulation results to evaluate the performance of the proposed schedulability tests in comparison to existing works in the case of G-EDF;
4. it provides resource augmentation bounds for the polynomial time schedulability test in (1) in the case of single-DAG systems.

Paper structure. The remainder of this paper is organized as follows. Section 2 describes the system model. Section 3 introduces our schedulability tests, while Section 4 presents the simulation results. Finally, Section 5 provides the concluding remarks.

[1]We notice that the runtime efficiency of this test makes it a valid candidate solution to the problem of online admission control of parallel tasks with arbitrary utilization presented in [16].

2. SYSTEM MODEL

In the *sporadic DAG model* described in [5], a task τ_i ($i = 1, \dots, n$) is specified by a 3-tuple (G_i, D_i, T_i), where G_i is a vertex-weighted Directed Acyclic Graph (DAG), and D_i, T_i are natural numbers (i.e., positive integers).

- The DAG G_i is specified as $G_i = (V_i, E_i)$, where V_i is a set of vertices and E_i a set of directed edges between these vertices; it is required that these edges do not form any oriented cycle. Each $v \in V_i$ represents an abstract sequential operation (i.e., a type of "job"). Each vertex $v \in V_i$ is characterized by a weight $e_v \in \mathbb{N}$, that is the *worst-case execution time* (WCET) of all jobs of (type) v. The edges represent dependencies between jobs as described in the next paragraph. For $v \in V_i$, we define $\tau(v) := i$ and we set $G_v := G_{\tau(v)}$, $D_v := D_{\tau(v)}$, $T_v := T_{\tau(v)}$.

- The *period* $T_i \in \mathbb{N}$. A *release* of a *DAG-job* of the task at time-instant t means that a job of v is released at time-instant t for each $v \in V_i$; t is called the *release date* of both the DAG-job and the jobs that compose it. The period denotes the minimum amount of time that must elapse between the release of successive DAG-jobs: if a DAG-job is released at t, then the next DAG-job of the same task can not be released prior to time-instant $t + T_i$. Notice that we do *not* require $e_v \le T_v$ in this work. For a possible "realization" S of the sporadicity constraints, we denote the release date of the j-th job of v ($j \in \mathbb{N}$) by $r_v^{j,\,S}$. We identify S with the ordered set $(r_{v_1}^{1,\,S}, r_{v_2}^{1,\,S}, \dots, r_{v_1}^{2,\,S}, r_{v_2}^{2,\,S}, \dots)$ (all the vertices in the system, arbitrarily ordered), and we refer to S as the "sequence of release dates". The edges represent dependencies between jobs belonging to the same DAG-job: if $(v_1, v_2) \in E_i$, then each job of v_1 must complete execution before *the* job of v_2 *in the same DAG-job* can begin execution; we call this job of v_1 a *predecessor job* of the job of v_2. Any groups of jobs that are not constrained (directly or indirectly) by this kind of precedence constraint may execute in parallel, whenever enough processing resource is available for them (e.g., there is no "intra-vertex" precedence constraint). A job (that has been released and has not yet completed execution) is *enabled* at time-instant t if all its predecessor jobs have completed execution at time-instant t.

- The *deadline* $D_i \in \mathbb{N}$. If a DAG-job is released at time-instant t, then all $|V_i|$ jobs that were released at t must complete execution by time-instant $t + D_i$. We assume that each of these jobs is discarded from the system at time-instant $t + D_i + 1$.[2] For a possible sequence of release dates S, we denote the time-instant by which the j-th job of v completes execution ($j \in \mathbb{N}$) by $f_v^{j,\,S}$ ($\ge r_v^{j,\,S}$). The task τ_i is said to have a *constrained* deadline if $D_i \le T_i$. *In this paper, we consider tasks with unconstrained or arbitrary deadline*, that is we make no assumption on the relation existing between the deadline and the period of a task. Notice that a task with arbitrary deadline may release a DAG-job prior to the completion of its previously-released DAG-job.

[2]This is a "harmless" (since we will consider *hard* real-time schedulability problems) but technically useful assumption (see, e.g., the proof of Lemma 1).

In this paper, we consider a *task-set* $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ representing a collection of n sporadic DAG-tasks. The jobs of the task-set are executed on a *platform* Π consisting of $m \geq 1$ identical processors.

Some additional notation and terminology:

- A *path* in the sporadic DAG-task τ_i is a sequence of vertices $(v_1, v_2, \ldots, v_k)$ such that (v_j, v_{j+1}) is an edge of G_i, $1 \leq j < k$; the *length* of this path is defined to be the sum of the WCETs of all its vertices: $\sum_{j=1}^{k} e_{v_j}$.
- We say that the vertex v' is a *predecessor* of the vertex v if there exists a path $(v', \ldots, v)$ in G_v; we denote by P_v the set of *immediate predecessors* of v, i.e., $v' \in P_v$ if and only if $(v', v) \in E_{\tau(v)}$.
- We denote by $\mathrm{len}(G_i)$ the length of the longest path in G_i; $\mathrm{len}(G_i)$ can be computed in time linear in the number of vertices and the number of edges of the DAG G_i, by first obtaining a topological order of its vertices and then running a straightforward dynamic program.
- We define $\mathrm{vol}(G_i) := \sum_{v \in V_i} e_v$; $\mathrm{vol}(G_i)$ can be computed in time linear in the number of vertices of the DAG G_i.
- We define $u_i := \frac{\mathrm{vol}(G_i)}{T_i}$ and $U := \sum_{i=1}^{n} u_i$; u_i (resp., U) is called the *utilization* of the DAG-task τ_i (resp., the *total utilization* of the task-set).

In the following sections, it will be useful to consider indices over the set of all vertices in the task-set $\mathcal{T}$; for this reason, we introduce the notations

$$v \in \mathcal{T}, \qquad \sum_{v \in \mathcal{T}}$$

as abbreviations for, respectively,

$$v \in \bigcup_{i=1}^{n} V_i, \qquad \sum_{i=1}^{n} \sum_{v \in V_i}.$$

We denote by $|\mathcal{T}|$ the cardinality of the set $\bigcup_{i=1}^{n} V_i$. Moreover, we fix an ordering of the vertices $v \in \mathcal{T}$ and we adopt the vector notation

$$\mathbf{a} := (a_v)_{v \in \mathcal{T}} \in \mathbb{Z}^{|\mathcal{T}|},$$

where $a_v \in \mathbb{Z}$ for each $v \in \mathcal{T}$.

The task-set $\mathcal{T}$ is *$\mathcal{A}$-schedulable* on the platform Π with respect to a scheduling algorithm $\mathcal{A}$ if all its jobs meet their deadlines when scheduled according to $\mathcal{A}$, i.e.,

$$f_v^{j,\,S} - r_v^{j,\,S} \leq D_v \quad \forall v \in \mathcal{T}, \forall S, \forall j \in \mathbb{N}$$

or, equivalently, if

$$R_v := \sup_{S} \sup_{j \in \mathbb{N}} \left(f_v^{j,\,S} - r_v^{j,\,S} \right) \leq D_v \quad \forall v \in \mathcal{T}.$$

$R_v \geq e_v$ is called the *response time* of the vertex $v \in \mathcal{T}$.[3] The task-set $\mathcal{T}$ is *feasible* if it is $\mathcal{A}$-schedulable with respect to a scheduling algorithm $\mathcal{A}$.

In this paper, we adopt the *discrete-time* concept, i.e., any time value is an integer. This is based on the assumption that all events in the system happen only at clock ticks.

[3]For simplicity of notation, we omit to specify the dependence of R_v (and $f_v^{j,\,S}$) on the underlying system $(\mathcal{T}, \Pi, \mathcal{A})$.

3. INTERFERENCE

We begin by introducing a notion of "interference" for the sporadic DAG model w.r.t. a generic scheduling algorithm $\mathcal{A}$.

Definition 1. Let $(\mathcal{T}, \Pi, \mathcal{A})$ be a system.

(i) Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates (compatible with the sporadic DAG model) and let j be a natural number. We define the function $\mathcal{I}_v^{j,\,S}$,

$$\mathcal{I}_v^{j,\,S} \colon \mathbb{N} \to \mathbb{Z}$$
$$X_v \mapsto \mathcal{I}_v^{j,\,S}(X_v),$$

where $\mathcal{I}_v^{j,\,S}(X_v)$ equals the amount of time the j-th job of v from S is *either* not enabled *or* enabled but not executing in the time-interval $[r_v^{j,\,S}, r_v^{j,\,S} + X_v)$ of length X_v. More formally:

$$\mathcal{I}_v^{j,\,S}(X_v) := \sum_{t = r_v^{j,\,S}}^{r_v^{j,\,S} + X_v - 1} \chi_v^{j,\,S}(t),$$

where $\chi_v^{j,\,S}(t)$ equals 1 if the j-th job of v from S is *either* not enabled *or* enabled but not executing at time-instant t, and it equals 0 otherwise.[4]

(ii) We define the function $\boldsymbol{\mathcal{I}}$,

$$\boldsymbol{\mathcal{I}} \colon \mathbb{N}^{|\mathcal{T}|} \to \mathbb{Z}^{|\mathcal{T}|}$$
$$\mathbf{X} \mapsto \left(\mathcal{I}_v(\mathbf{X})\right)_{v \in \mathcal{T}},$$

according to

$$\mathcal{I}_v(\mathbf{X}) := \max_{S} \max_{j \in \mathbb{N}} \left\{ \mathcal{I}_v^{j,\,S}(X_v) \right\}.$$

$\boldsymbol{\mathcal{I}}$ is called the *interference* associated with $(\mathcal{T}, \Pi, \mathcal{A})$.[5]

It follows from the above definition that all jobs meet their deadlines if

$$e_v + \mathcal{I}_v(\mathbf{D}) \leq D_v \qquad \forall v \in \mathcal{T},$$

or equivalently (in vector notation) if

$$\mathbf{e} + \boldsymbol{\mathcal{I}}(\mathbf{D}) \leq \mathbf{D}.$$

Unfortunately, we are not able to efficiently compute the interference for all possible systems (this is true even when limiting to the case $\mathcal{A}$ = G-EDF or $\mathcal{A}$ = G-DM). Instead, we adopt an approach (to be described in this section) based on RTA and relying on an upper-bound on the interference. We can summarize our approach as follows.

1. We start by considering a "parameter" $\mathbf{Y} \in \mathbb{N}^{|\mathcal{T}|}$ such that
$$\mathbf{Y} \geq \min\{\mathbf{D} + \mathbf{1}, \mathbf{R}\} \tag{1}$$
(i.e., $Y_v \geq \min\{D_v + 1, R_v\}$ for each $v \in \mathcal{T}$).
2. For each $\mathbf{Y}$ as in step 1, we define a function $\overline{\boldsymbol{\mathcal{I}}}(-, \mathbf{Y})$,
$$\overline{\boldsymbol{\mathcal{I}}}(-, \mathbf{Y}) \colon \mathbb{N}^{|\mathcal{T}|} \to \mathbb{Z}^{|\mathcal{T}|}$$
$$\mathbf{X} \mapsto \left(\overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y})\right)_{v \in \mathcal{T}},$$
representing a (point-wise) upper-bound on the interference, i.e., $\overline{\boldsymbol{\mathcal{I}}}(\mathbf{X}, \mathbf{Y}) \geq \boldsymbol{\mathcal{I}}(\mathbf{X})$ for each $\mathbf{X} \in \mathbb{N}^{|\mathcal{T}|}$.

[4]Remark that the time is discrete (c.f., Section 2).

[5]For simplicity of notation, we omit to specify the dependence of $\boldsymbol{\mathcal{I}}$ (and $\boldsymbol{\mathcal{I}}^{j,\,S}$) on the underlying system $(\mathcal{T}, \Pi, \mathcal{A})$.

3. We prove that any vector $\mathbf{X} \in \mathbb{N}^{|\mathcal{T}|}$ satisfying the inequality

$$\mathbf{e} + \overline{\boldsymbol{\mathcal{I}}}(\mathbf{X}, \mathbf{Y}) \le \mathbf{X} \qquad (2)$$

must also satisfy $\mathbf{X} \ge \mathbf{R}$ (c.f. Theorem 1).

4. Step 3 allows us to define a schedulability test by first "finding" a parameter $\mathbf{Y}$ and a vector $\mathbf{X}$ satisfying (2), and by then simply checking if $\mathbf{X} \le \mathbf{D}$.

Intuitively, we consider Y_v as the (known) upper-bound to the response time of v, and we consider X_v as the "unknown" (to be defined using (2)) representing the same upper-bound.

From now on, we assume $\mathcal{A}$ = G-EDF or $\mathcal{A}$ = G-DM unless stated otherwise.

3.1 Upper-bounds

In this subsection, we define the upper-bound $\overline{\boldsymbol{\mathcal{I}}}(-, \mathbf{Y})$ (Definition 5), and we show how it can be used to estimate the response times (c.f. Theorem 1). The main idea underlying this result is to estimate the interference "by walking through a so called $(j,\ S)$-critical path", as defined next.

Definition 2. Let $\mathcal{A}$ be arbitrary. Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates and let j be a natural number. We recursively define the path $(v_K, v_{K-1}, \ldots, v_1)$, called the $(j,\ S)$-*critical path of* v, as follows:

(i) first, set $v_1 := v$;

(ii) for $k > 1$, let v_k denote the first vertex $u \in P_{v_{k-1}}$ (w.r.t. the ordering on $\mathcal{T}$) such that $f_u^{j,\,S} \ge f_{u'}^{j,\,S}$ for each $u' \in P_{v_{k-1}}$, if such a vertex exists (i.e., if $P_{v_{k-1}} \neq \emptyset$); otherwise, set $K := k-1$ and exit the recursion.

Intuitively, the $(j,\ S)$-critical path of v is built, starting with v, by going "backward" to the immediate predecessor of the current vertex whose j-th job from S completed last; an example of $(j,\ S)$-critical path is illustrated in Figure 1.

The notion of $(j,\ S)$-critial path allows us to "split" the interference (estimate) into the sum of two distinct terms; this result is formally established by Proposition 1. We first introduce the following fundamental notation.

Definition 3. Let $\mathcal{A}$ be an arbitrary scheduling algorithm. Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates and let j be a natural number. We define the function $\mathcal{W}_v^{j,\,S} : \mathbb{N} \to \mathbb{Z}$, $X_v \mapsto \mathcal{W}_v^{j,\,S}(X_v)$, where $\mathcal{W}_v^{j,\,S}(X_v)$ equals the amount of time the j-th job of v from S is enabled but not executing in the time-interval $[r_v^{j,\,S}, r_v^{j,\,S} + X_v)$ of length X_v. More formally:

$$\mathcal{W}_v^{j,\,S}(X_v) := \sum_{t=r_v^{j,\,S}}^{r_v^{j,\,S}+X_v-1} \zeta_v^{j,\,S}(t),$$

where $\zeta_v^{j,\,S}(t)$ equals 1 if the j-th job of v from S is enabled but not executing at time-instant t, and it equals 0 otherwise. ($\mathcal{W}_v^{j,S}(X_v) \le \mathcal{I}_v^{j,\,S}(X_v)$ for each $X_v \in \mathbb{N}$.)

Proposition 1. *Let $\mathcal{A}$ be an arbitrary scheduling algorithm. Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates and let j be a natural number. Moreover, let $(v_K, v_{K-1}, \ldots, v_1)$ denote the $(j,\ S)$-critical path of v and let $X_v \in \mathbb{N}$ be arbitrary. The following inequality holds:*

$$\mathcal{I}_v^{j,\,S}(X_v) \le \sum_{k=2}^{K} e_{v_k} + \sum_{k=1}^{K} \mathcal{W}_{v_k}^{j,\,S}(X_v). \qquad (3)$$

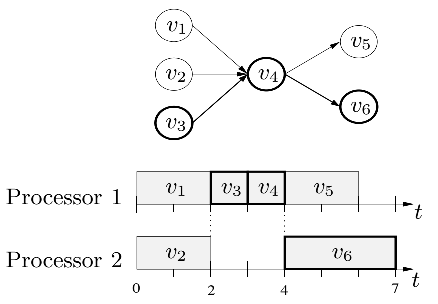


Figure 1: Example illustrating the $(j,\ S)$-critical path of vertex v_6 on a platform with $m = 2$ processors. Vertices have WCETs $e_1 = 2, e_2 = 2, e_3 = 1, e_4 = 1, e_5 = 2, e_6 = 3$; the jobs shown in the schedule all belong to the j-th DAG-job of the task from the given sequence S of release dates, and are labeled with the symbol of the corresponding vertex. The $(j,\ S)$-critical path of v_6 is (v_3, v_4, v_6); notice that this path does *not* coincide with a "critical path reaching" v_6 (i.e., with either (v_1, v_4, v_6) or (v_2, v_4, v_6)).

Proof. The proof is by induction on $K \in \mathbb{N}$. The base case of the induction ($K = 1$) holds trivially, since in this case vertex v has no predecessors and $\mathcal{I}_v^{j,\,S}(X_v) = \mathcal{W}_v^{j,\,S}(X_v)$.

We now consider the induction step. By the induction hypothesis (applied for v_2), we know that

$$\mathcal{I}_{v_2}^{j,\,S}(X_v) \le \sum_{k=3}^{K} e_{v_k} + \sum_{k=2}^{K} \mathcal{W}_{v_k}^{j,\,S}(X_v). \qquad (4)$$

Since $f_{v_2}^{j,\,S} \ge f_u^{j,\,S}$ for each $u \in P_{v_1}(= P_v)$, by construction, it follows from (4) that we can upper-bound the amount of time in which the j-th job of v from S is not enabled in the considered time-interval with the value

$$\sum_{k=2}^{K} e_{v_k} + \sum_{k=2}^{K} \mathcal{W}_{v_k}^{j,\,S}(X_v) \; \left(\ge e_{v_2} + \mathcal{I}_{v_2}^{j,\,S}(X_v)\right).$$

Finally, we upper-bound $I_v^{j,\,S}(X_v)$ by adding $W_v^{j,\,S}(X_v) = W_{v_1}^{j,\,S}(X_v)$ to the above value, thus obtaining (3). $\square$

Our next step towards the identification of a "safe" upper-bound on the interference will be to upper-bound the term $\sum_{k=1}^{K} \mathcal{W}_{v_k}^{j,\,S}(X_v)$ in equation (3). We provide these upper-bounds in Lemmas 1 and 2 for the case of G-EDF and G-DM scheduling, respectively. Some preliminary notations:

Definition 4. Let $v, v' \in \mathcal{T}$ be arbitrary.

(i) Define the function $\overline{\mathcal{W}}_{v',v}^{\text{G-EDF}} : \mathbb{N}^{|\mathcal{T}|} \times \mathbb{N}^{|\mathcal{T}|} \to \mathbb{Z}$ as follows:

$$\overline{\mathcal{W}}_{v',v}^{\text{G-EDF}}(\boldsymbol{X}, \boldsymbol{Y}) := \left(\left\lceil \frac{Y_{v'} + \min\{D_v - D_{v'}, X_v\}}{T_{v'}} \right\rceil_0 - \gamma_{v',v} \right) e_{v'},$$

where

$$\lceil a \rceil_0 := \begin{cases} \lceil a \rceil & \text{if } a \ge 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$\gamma_{v',v} := \begin{cases} 1 & \text{if } v \text{ precedes } v' \\ 0 & \text{otherwise.} \end{cases}$$

(ii) Define the function $\overline{\mathcal{W}}^{\text{G-DM}}_{v',v}: \mathbb{N}^{|\mathcal{T}|} \times \mathbb{N}^{|\mathcal{T}|} \to \mathbb{Z}$ as follows:

$$\overline{\mathcal{W}}^{\text{G-DM}}_{v',v}(\boldsymbol{X},\boldsymbol{Y}) := \begin{cases} \left(\left\lceil \frac{Y_{v'}+X_v}{T_{v'}} \right\rceil - \gamma_{v',v}\right) e_{v'}, & \text{if } D_{v'} \le D_v \\ 0 & \text{otherwise,} \end{cases}$$

where $\gamma_{v',v}$ is as defined above.

Lemma 1 (G-EDF)**.** *Consider the case $\mathcal{A}$ = G-EDF. Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates and let j be a natural number. Let $(v_K, v_{K-1}, \ldots, v_1)$ denote the (j, S)-critical path of v. Finally, let $\mathbf{Y} \ge \min\{\mathbf{D}+\mathbf{1}, \mathbf{R}\}$ and let $\mathbf{X} \in \mathbb{N}^{|\mathcal{T}|}$ be arbitrary. The following inequality holds:*

$$m \cdot \left(\sum_{k=1}^{K} \mathcal{W}^{j,\,S}_{v_k}(X_v)\right) \le \sum_{v' \in \mathcal{T}} \overline{\mathcal{W}}^{\text{G-EDF}}_{v',v}(\boldsymbol{X},\boldsymbol{Y}) - \sum_{k=1}^{K} e_{v_k}. \quad (5)$$

Proof. Consider a time-instant t, $r^{j,\,S}_v \le t \le r^{j,\,S}_v + X_v - 1$, when the j-th job of v_k from S is enabled but not executing (i.e., $\zeta^{j,\,S}_{v_k}(t) = 1$). Since our scheduling algorithm is work-conserving, this job can be delayed from execution only if all processors are busy with "higher priority" jobs that preempt[6] the job of v_k; in other words, for each t such that $\zeta^{j,\,S}_{v_k}(t) = 1$, the processors must perform m units of work to execute some jobs preempting the job of v_k. Moreover, due to precedence constraints, the equality $\zeta^{j,\,S}_{v_k}(t) = 1$ implies $\zeta^{j,\,S}_{v_{k'}}(t) = 0$ for each $k' \ne k$ ($1 \le k' \le K$). As a consequence, we can upper-bound the left-hand term in (5) by determining an upper-bound on the total work performed by the processors to execute the jobs which preempt (any of) the j-th jobs of $v_1, \ldots, v_K$ from S in the considered time-interval.

With this in mind, let us now consider an arbitrary vertex $v' \in \mathcal{T}$. Notice that:

(i) every job of v' released before or at time-instant $r^{j,\,S}_v - Y_{v'}$ can not preempt the jobs of $v_1, \ldots, v_K$ in the time-interval $[r^{j,\,S}_v, r^{j,\,S}_v + X_v)$, since this job must have completed execution by time-instant $r^{j,\,S}_v$ (if $Y_{v'} \ge R_{v'}$) or it must have been discarded from the system by that time-instant (if $Y_{v'} \ge D_{v'} + 1$), by definition of $Y_{v'}$;

(ii) every job of v' released after time-instant $r^{j,\,S}_v + D_v - D_{v'}$ has lower priority than the jobs of $v_1, \ldots, v_K$, and thus can not preempt them in the considered time-interval; moreover, every job of v' released after (or at) time-instant $r^{j,\,S}_v + X_v$ can not preempt the jobs of $v_1, \ldots, v_K$ in the given time-interval.

Thus there are at most

$$\left\lceil \frac{Y_{v'} + \min\{D_v - D_{v'}, X_v\}}{T_{v'}} \right\rceil_0 \quad (6)$$

jobs of v' which can preempt the j-th jobs of $v_1, \ldots, v_K$ from S in the time-interval $[r^{j,\,S}_v, r^{j,\,S}_v + X_v)$.

Notice that if v precedes v' ($\gamma_{v',v} = 1$) or if v' belongs to the (j, S)-critical path of v, then we have $D_v = D_{v'}$ (v and v' belong to the same DAG, in this case) and expression (6) reduces to $\lceil Y_{v'}/T_{v'} \rceil \ge 1$. This number can be further reduced to $\lceil Y_{v'}/T_{v'} \rceil - 1$, because the job of v' released at time-instant $r^{j,\,S}_v$ (that has been certainly counted in (6)) can not preempt the jobs of $v_1, \ldots, v_K$ in the case under consideration, due to precedence constraints.

[6]More formally, we say that the job J' *preempts* the job J at time-instant t if J is enabled but not executing at time-instant t while J' is executing at that time-instant; notice that a job J' could be "interfering" with a job J without preempting it (e.g., J' could be preempting a predecessor job of J).

In conclusion, inequality (5) can be obtained by remarking that each of the "preempting jobs" of v' can contribute to the total work with at most $e_{v'}$ time-units and, finally, by summing over all vertices $v' \in \mathcal{T}$. □

Lemma 2 (G-DM)**.** *Consider the case $\mathcal{A}$ = G-DM. Let $v \in \mathcal{T}$ be a vertex, S a sequence of release dates and let j be a natural number. Let $(v_K, v_{K-1}, \ldots, v_1)$ denote the (j, S)-critical path of v. Finally, let $\mathbf{Y} \ge \min\{\mathbf{D}+\mathbf{1}, \mathbf{R}\}$ and let $\mathbf{X} \in \mathbb{N}^{|\mathcal{T}|}$ be arbitrary. The following inequality holds:*

$$m \cdot \left(\sum_{k=1}^{K} \mathcal{W}^{j,\,S}_{v_k}(X_v)\right) \le \sum_{v' \in \mathcal{T}} \overline{\mathcal{W}}^{\text{G-DM}}_{v',v}(\boldsymbol{X},\boldsymbol{Y}) - \sum_{k=1}^{K} e_{v_k}.$$

Proof (sketch). The proof follows the same argument in the proof of Lemma 1 with the exceptions that no preemption "from v' to v" is possible if $D_{v'} > D_v$ ($\overline{\mathcal{W}}^{\text{G-DM}}_{v',v}(\boldsymbol{X},\boldsymbol{Y}) = 0$) and that the first part of item (ii) above does not apply. □

As emerged from the proof of Lemmas 1 and 2, the function $\overline{\mathcal{W}}^{\mathcal{A}}_{v,v'}$ could be interpreted as an "interfering workload" (e.g., [17, 15]) of the vertex v' on the vertex v. We stress that the assumption $\boldsymbol{Y} \ge \min\{\boldsymbol{D}+\boldsymbol{1}, \boldsymbol{R}\}$ is essential to the proofs of the lemmas reported above.

We now define the anticipated interference upper-bound.

Definition 5. We define the function $\overline{\boldsymbol{\mathcal{I}}}$,

$$\overline{\boldsymbol{\mathcal{I}}} \colon \mathbb{N}^{|\mathcal{T}|} \times \mathbb{N}^{|\mathcal{T}|} \to \mathbb{Z}^{|\mathcal{T}|}$$
$$(\mathbf{X}, \mathbf{Y}) \mapsto \left(\overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y})\right)_{v \in \mathcal{T}},$$

according to

$$\overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y}) := \ell^+_v - e_v + \left\lfloor \frac{1}{m} \left(\sum_{v' \in \mathcal{T}} \overline{\mathcal{W}}^{\mathcal{A}}_{v',v}(\boldsymbol{X}, \boldsymbol{Y}) - \ell^+_v \right) \right\rfloor,$$

where

$$\ell^+_v := \max\left\{ \sum_{j=1}^{k} e_{v_j} \;\middle|\; \begin{array}{c} (v_1, \ldots, v_k) \text{ a path in } G_v \\ \text{s.t. } v_k = v \end{array} \right\}$$

is the *length of the critical path reaching v*.

Notice that the summation in the definition of $\overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y})$ extends, in particular, over the vertex v and over all the vertices in v's DAG; that is, we are accounting for a "self-interference" (e.g., [9]) on v and on v's DAG.

We are finally ready for the main result in this section.

Theorem 1. *Let $\mathbf{Y} \ge \min\{\mathbf{D}+\mathbf{1}, \mathbf{R}\}$, and let $\mathbf{X} \in \mathbb{N}^{|\mathcal{T}|}$ be arbitrary. For every $v \in \mathcal{T}$,*

$$e_v + \overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y}) \le X_v \implies X_v \ge R_v.$$

In particular, the inequality

$$\mathbf{e} + \overline{\boldsymbol{\mathcal{I}}}(\mathbf{X}, \mathbf{Y}) \le \mathbf{X} \quad (7)$$

implies $\mathbf{X} \ge \mathbf{R}$.

Proof. We will prove the contrapositive: for every $v \in \mathcal{T}$,

$$X_v < R_v \implies \overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y}) > X_v - e_v.$$

Let us suppose $X_v < R_v$ for some vertex $v \in \mathcal{T}$, and let us fix a sequence S of release dates and a natural number j such that $f^{j,\,S}_v - r^{j,\,S}_v > X_v$ (S and j exist, by definition of R_v). This means that the j-th job of v from S has not yet completed execution at time-instant $r_v^{j,\,S} + X_v$; in particular, we deduce

$$\mathcal{I}_v^{j,\,S}(X_v) > X_v - e_v. \tag{8}$$

Let $(v_K, v_{K-1}, \ldots, v_1)$ denote the $(j,\, S)$-critical path of v. We have[7]

$$\begin{aligned}
\overline{\mathcal{I}}_v(\mathbf{X}, \mathbf{Y}) &\geq \sum_{k=2}^{K} e_{v_k} + \left\lfloor \frac{1}{m} \left(\sum_{v' \in \mathcal{T}} \overline{\mathcal{W}}^{\mathcal{A}}_{v',v}(\boldsymbol{X}, \boldsymbol{Y}) - \sum_{k=1}^{K} e_{v_k} \right) \right\rfloor \\
&\geq \sum_{k=2}^{K} e_{v_k} + \sum_{k=1}^{K} \mathcal{W}_{v_k}^{j,\,S}(X_v) \quad \text{(by Lemmas 1 and 2)} \\
&\geq \mathcal{I}_v^{j,\,S}(X_v) \quad \text{(by Proposition 1)} \\
&> X_v - e_v \quad \text{(by (8))},
\end{aligned}$$

as desired. □

Remark 1. We remark that the argument presented in the proof of Theorem 1 could be applied *without modification* to *any* work-conserving scheduling algorithm $\mathcal{A}$, once a "suitable" expression for the interfering workload is available, that is, once a function $\overline{\mathcal{W}}^{\mathcal{A}}_{v',v}$ satisfying analogous to Lemmas 1 and 2 has been determined. □

Remark 2. The upper-bounds established by Lemmas 1 and 2 when $\mathcal{A}$ = G-EDF and $\mathcal{A}$ = G-DM, respectively, can be tightened if additional assumptions are imposed on the DAG model (e.g., if $e_v \leq T_v$ for all $v \in \mathcal{T}$). In this paper we adhere to the (more general) model described in Section 2, thus leaving this analysis for future work. □

3.2 Schedulability tests

Theorem 1 allows us to define a simple schedulability test by first setting the parameter $\mathbf{Y} := \mathbf{D} + \mathbf{1}$ and then testing a "tentative" vector $\mathbf{X} \leq \mathbf{D}$ against inequality (7).

Definition 6 (RTA-P)**.** We denote by RTA-P the schedulability test defined according to the following steps: given an input system $(\mathcal{T}, \Pi, \mathcal{A})$,

1. return TRUE, if $\mathbf{e} + \overline{\boldsymbol{\mathcal{I}}}(\mathbf{D}, \mathbf{D}+\mathbf{1}) \leq \mathbf{D}$;
2. return FALSE.

The running time of RTA-P is polynomial in the input size. We refer to Section 3.3 for an analysis of the resource augmentation of RTA-P for single-DAG systems.

In the rest of this subsection, we describe an iterative procedure to construct both the parameter $\mathbf{Y}$ and the test vector $\mathbf{X} \leq \mathbf{D}$. Definition 7 provides the basic notation.

Definition 7. Let $\mathbf{Y} \geq \min\{\mathbf{D}+\mathbf{1}, \mathbf{R}\}$. We recursively define the sequence $(\mathbf{X}[\mu])_{\mu \in \mathbb{N}}$ in $\mathbb{N}^{|\mathcal{T}|}$ by

$$\mathbf{X}[\mu] := \begin{cases} \mathbf{e} & \text{if } \mu = 1, \\ \min\left\{\mathbf{D}+\mathbf{1}, \mathbf{e} + \overline{\boldsymbol{\mathcal{I}}}(\mathbf{X}[\mu-1], \mathbf{Y})\right\} & \text{if } \mu > 1. \end{cases} \tag{9}$$

It follows directly from Definition 7 that the equality $\mathbf{X}[\widetilde{\mu}+1] = \mathbf{X}[\widetilde{\mu}]$ ($\widetilde{\mu} \in \mathbb{N}$) implies $\mathbf{X}[\mu] = \mathbf{X}[\widetilde{\mu}]$ for each $\mu \geq \widetilde{\mu}$. Proposition 2 establishes that such a $\widetilde{\mu}$ does indeed exist.

[7]For the first inequality remark that the WCETs are integers, so these can be carried in (and factorized within) the "floor" sign.

Proposition 2. *The sequence* $(\mathbf{X}[\mu])_{\mu \in \mathbb{N}}$ *defined by (9)* has a fixed-point*, i.e., there exists a natural number* $\widetilde{\mu}$ *such that*

$$\mathbf{X}[\widetilde{\mu}+1] = \mathbf{X}[\widetilde{\mu}],$$

and such a $\mathbf{X}[\widetilde{\mu}]$ *can be computed in time pseudo-polynomial in the sizes of the system* $(\mathcal{T}, \Pi, \mathcal{A})$ *and of the parameter* $\mathbf{Y}$*.*

Moreover, denoted by $\widetilde{\mathbf{X}}(\mathbf{Y})$ *the (unique) fixed-point of* $(\mathbf{X}[\mu])_{\mu \in \mathbb{N}}$*, the following property holds: for every* $v \in \mathcal{T}$*,*

$$\pi^v\left(\widetilde{\mathbf{X}}(\mathbf{Y})\right) \leq D_v \implies \pi^v\left(\widetilde{\mathbf{X}}(\mathbf{Y})\right) \geq R_v, \tag{10}$$

where $\pi^v\colon \mathbb{N}^{|\mathcal{T}|} \to \mathbb{N}$ *denotes the* projection *onto the* v*-th component.*

Proof. The existence of a fixed-point follows immediately from the (componentwise) monotonicity and from the boundedness of $(\mathbf{X}[\mu])_{\mu \in \mathbb{N}}$ together with the discrete-time assumption.

We now prove that the fixed-point can be computed in pseudo-polynomial time. Notice that equation (9) is separable w.r.t. its components: that is, $\pi^v(\mathbf{X}[\mu])$ is independent from (the value of) $\pi^{v'}(\mathbf{X}[\mu'])$ for $v' \neq v$ and $\mu' < \mu$, by definition of $\overline{\boldsymbol{\mathcal{I}}}$. In particular, every components of $\mathbf{X}[\mu]$ must stabilize (i.e., equal a constant value) after at most $\max\{2, D_{\max} - e_{\min} + 2\}$ iterations of equation (9), where $D_{\max} := \max\{D_i \mid 1 \leq i \leq n\}$ and $e_{\min} := \min\{e_v \mid v \in \mathcal{T}\}$. To conclude, it is thus enough to remark that *each* of these iterations can be run in polynomial time in the sizes of the system $(\mathcal{T}, \Pi, \mathcal{A})$ and of the parameter $\mathbf{Y}$.

For the last part of the proposition notice that, if $\mathbf{X}[\widetilde{\mu}]$ is the fixed-point of $(\mathbf{X}[\mu])_{\mu \in \mathbb{N}}$ and $\pi^v(\mathbf{X}[\widetilde{\mu}]) \leq D_v$, then

$$\begin{aligned}
(D_v \geq)\ \pi^v(\mathbf{X}[\widetilde{\mu}]) &= \pi^v(\mathbf{X}[\widetilde{\mu}+1]) \\
&:= \min\left\{D_v + 1, e_v + \overline{\mathcal{I}}_v(\mathbf{X}[\widetilde{\mu}], \mathbf{Y})\right\} \\
&= e_v + \overline{\mathcal{I}}_v(\mathbf{X}[\widetilde{\mu}], \mathbf{Y});
\end{aligned}$$

thus, by Theorem 1, we have $\pi^v(\mathbf{X}[\widetilde{\mu}]) \geq R_v$, as desired. □

As emerged from the proof, a fixed-point $\widetilde{\mathbf{X}}(\mathbf{Y}) \leq \mathbf{D}$ satisfies inequality (7); thus we have defined a procedure that, given a parameter $\mathbf{Y}$, returns the "ideal" test vector $\widetilde{\mathbf{X}}(\mathbf{Y})$. It is now a matter of selecting the parameters.

Definition 8 (RTA)**.** Let $\xi \in \mathbb{N}$ be fixed. We denote by RTA(ξ) the schedulability test defined according to the following steps: given an input system $(\mathcal{T}, \Pi, \mathcal{A})$,

1. set $\nu := 1$ and $\mathbf{Y}[1] := \mathbf{D} + \mathbf{1}$;
2. iteratively compute $\mathbf{X}[\mu]$ according to equation (9) until a fixed-point $\widetilde{\mathbf{X}}(\mathbf{Y}[\nu])$ as in Proposition 2 is reached; return TRUE, if $\widetilde{\mathbf{X}}(\mathbf{Y}[\nu]) \leq \mathbf{D}$;
3. update $\nu := \nu + 1$ and
$$\mathbf{Y}[\nu] := \min\left\{\mathbf{Y}[\nu-1], \widetilde{\mathbf{X}}(\mathbf{Y}[\nu-1])\right\};$$
return FALSE, if $\nu > \xi$ or $\mathbf{Y}[\nu] = \mathbf{Y}[\nu-1]$; go to step 2.

Informally, RTA(ξ) starts with a "safe" parameter $\mathbf{Y}$ and then constructs the test vector $\widetilde{\mathbf{X}}(\mathbf{Y})$ as in Proposition 2; if $\widetilde{\mathbf{X}}(\mathbf{Y})$ can not guarantee the schedulability, RTA(ξ) uses this vector to construct a "better" parameter and iterates.

Theorem 2. *RTA(ξ) is well-defined (*that is*, it is a schedulability test); its running time is pseudo-polynomial in the input size.*

The proof uses the following lemma.

Lemma 3. *For every (computed) $\nu \in \mathbb{N}$, the following inequalities are satisfied: (i) $\mathbf{Y}[\nu] \leq \mathbf{D}+\mathbf{1}$, and (ii) $\mathbf{Y}[\nu] \geq \min\{\mathbf{D}+\mathbf{1}, \mathbf{R}\}$.*

Proof. (i) This follows immediately (e.g., by induction on ν) from the definition of $\mathbf{Y}[\nu]$ in steps 1 and 3.

(ii) The proof is by induction on ν. The base case of the induction ($\nu = 1$) follows trivially from step 1. The inductive step follows by combining the definition of $\mathbf{Y}[\nu]$ (in step 3) together with equation (10) of Proposition 2. □

Proof (of Theorem 2). We start showing that RTA(ξ) runs in pseudo-polynomial time. Each of the three steps of RTA(ξ) can be executed at most ξ times. By Proposition 2, the running times of every steps are pseudo-polynomial in the sizes of $(\mathcal{T}, \Pi, \mathcal{A})$ and of $\mathbf{Y}[\nu]$ ($\nu \in \mathbb{N}$). Thus, Lemma 3 concludes this part of the proof.

The well-definiteness of RTA(ξ) follows immediately from Lemma 3(ii) together with equation (10) of Proposition 2 (c.f. the "return" statement in step 2). □

Remark 3. RTA(ξ) dominates RTA(ξ') for every $\xi' < \xi$. It is possible to remove the halting condition $\nu > \xi$ in Definition 8 *without* compromising the termination of the test; this is due to the monotonicity and boundedness of $(\mathbf{Y}[\nu])_{\nu\in\mathbb{N}}$ (and the remaining halting condition in step 3). Notice, however, that the proof of pseudo-polynomiality presented above does not extend to the resulting schedulability test; we refer to Section 4 for considerations related to our simulations. □

Remark 4. In Definition 8, we have presented a concise (formally correct) description of RTA(ξ). Of course, an actual implementation of this test should exploit the "parallelism" implicit in this description and, in particular, in equation (9). For example, due to separability (see the proof of Proposition 2), there is no need to iterate this equation on components which have already stabilised. For reason of space, we will not discuss these (and other) optimizations further in the present paper. □

3.3 Augmentation bounds

In this subsection, we restrict the domain of each schedulability test to be the set $\mathcal{X}$ of systems of the form

$$x := (\{\tau\},\ \Pi,\ \textit{G-EDF}),$$

*where $\tau := (G, D, T)$ is a sporadic DAG-task with $D > T$ and the number of processors m in Π is arbitrary (*not *fixed).*

We recall the following definition to fix the notation.

Definition 9. The *(integer) resource augmentation* $\sigma(B) \in \mathbb{N}\cup\{\infty\}$ of a schedulability test B is the infimum of the $\sigma \in \mathbb{N}$ which satisfies the following property: for every $x \in \mathcal{X}$,

$$B \text{ returns FALSE on } x \implies x^{(\sigma)} \text{ is not feasible},$$

where $x^{(\sigma)}$ denotes the system obtained after multiplying x's WCETs by a factor of σ.[8]

Proposition 3. σ*(RTA-P)* ≤ 3.

[8]Resource augmentation (or "speedup") in the presence of fractional-time has been considered in the literature (e.g., [5, 7]); Definition 9 maintains the discrete-time assumption, consistently with the model described in Section 2.

Proof. We will prove the contrapositive: for every $x \in \mathcal{X}$,

$$x^{(3)} \text{ is feasible} \implies \text{RTA-P returns TRUE on } x.$$

Let us consider a (generic) system x such that $x^{(3)}$ is feasible; in particular, we have

$$\mathrm{len}(G) \leq \tfrac{1}{3}D$$
$$\mathrm{vol}(G) \leq \tfrac{1}{3}mT.$$

Then, for $v \in \mathcal{T}$, we compute

$$\begin{aligned} e_v + &\overline{\mathcal{I}}_v(\mathbf{D}, \mathbf{D}+\mathbf{1}) \\ &\leq \mathrm{len}(G) + \frac{1}{m}\left(\left\lceil\frac{D+1}{T}\right\rceil \cdot \mathrm{vol}(G) - \mathrm{len}(G)\right) \\ &< \tfrac{1}{3}D + \frac{1}{m}\left(2\frac{D+1}{T}\cdot\tfrac{1}{3}mT - \tfrac{1}{3}D\right) \\ &= D\left(1 - \tfrac{1}{3m}\right) + \tfrac{2}{3} < D+1; \end{aligned} \tag{11}$$

since the left-hand term in equation (11) is an integer number, we conclude $e_v + \overline{\mathcal{I}}_v(\mathbf{D}; \mathbf{D}+\mathbf{1}) \leq D$ and thus, due to the arbitrariness of v, RTA-P returns TRUE on x. □

Remark 5. It is possible to modify the proof of Proposition 3 and to "include" the case $\mathcal{A}$ = G-DM by replacing the first inequality in (11) with

$$\begin{aligned} e_v + &\overline{\mathcal{I}}_v(\mathbf{D}, \mathbf{D}+\mathbf{1}) \\ &\leq \mathrm{len}(G) + \frac{1}{m}\left(\left\lceil\frac{2D+1}{T}\right\rceil \cdot \mathrm{vol}(G) - \mathrm{len}(G)\right). \end{aligned}$$

By reasoning as in the proof of Proposition 3, we could then obtain σ(RTA-P) ≤ 5. □

4. SIMULATION RESULTS

In this section, we present a set of simulation results evaluating the performance of RTA-P and RTA with respect to the schedulability tests described in [7], in terms of both schedulability ratio and running time. *To the best of the authors' knowledge, these are the only currently available tests for G-EDF or G-DM scheduling of (multiple) sporadic DAG-tasks with arbitrary deadline.* In this paper, we restrict the comparison to these kind of tests, since our analysis were specifically targeted to the (general) model described in Section 2 (see also Remarks 1 and 2).

The notation used to refer to the schedulability tests is reported in Table 1. We remark that BON(δ) estimates the maximum "workload density" by approximating it up to an ε-error, $\varepsilon := 2^{-\delta}$; for values of ε tending to zero, this test performs more accurately but it is also computationally more demanding.[9] All tests have been implemented as *sequential* algorithms and executed on an 8-core Intel Xeon running at 3.5GHz.

In each simulation, task-sets have been generated starting from a given "configuration",

$$(n, U, T_{\min}, T_{\max}, \alpha_{\min}, \alpha_{\max}, N_{\min}, N_{\max}, p_{\text{edge}}) \in \mathbb{N}^9,$$

according to the following steps:

[9]The (worst-case) running time of BON(δ) grows linearly with $1/\varepsilon := 2^{\delta}$; this explains the different scales of the parameters ξ and δ chosen in the simulations (remark that the running time of RTA(ξ) grows linearly with ξ).

RTA(ξ)	The test described in Definition 8.
RTA-P	The test described in Definition 6.
BON(δ)	The test proposed in [7], Section IV.
BON-P	The test proposed in [7], Section V.

Table 1: Schedulability tests in the simulations.

1. the utilization u_i of the DAG-task τ_i ($1 \leq i \leq n$) is randomly generated by using the UUniSort algorithm on n and U (see [6]);
2. the period T_i is uniformly chosen in $[T_{\min}, T_{\max}]$; the deadline D_i is uniformly chosen in $[\alpha_{\min}T_i, \alpha_{\max}T_i]$;
3. the number of vertices $|V_i|$ of the DAG G_i is uniformly chosen in the range $[N_{\min}, N_{\max}]$; WCETs are randomly generated by using UUniSort on $|V_i|$ and u_iT_i;
4. for any two vertices v, v' of G_i such that $v < v'$, we add the edge (v, v') with percent probability p_{edge}.

For each configuration, we generated $P := 10000$ task-sets. We evaluated more than 800 different configurations, *including* the case of implicit deadlines ($\alpha_{\min} = \alpha_{\max} = 1$); our results were consistent across all of them; due to space constraints, we report results for some representative configurations only. Moreover, we limit our report to the case of G-EDF scheduling; similar considerations can be made, according to our results, for the case of G-DM scheduling.

4.1 Simulation I

In this simulation, we aim at studying the behavior of RTA(ξ) and BON(δ), in terms of schedulability ratio and running time, for different values of ξ and δ respectively. This simulation is mainly intended to "justify" the selection of these parameters that we will make in order to perform the actual comparison between RTA(ξ) and BON(δ) (c.f. Simulation II).

The results for this simulation are all reported in Figures 1-4 and in Tables 2-5, at page 9 in this paper.

(i) Figures 1 and 3 report the schedulability ratio of the tests as a function of the total utilization U of the task-sets for a platform composed of 16 processors. The values of the configurations (except for the value of the total utilization, that is determined by the abscissa) are written in the figure captions. Tables 2 and 4 report the corresponding minimum, maximum and average running times of the tests.

(ii) Figures 2 and 4 report the schedulability ratio of the tests as a function of the number of processors m composing the platform; again, the values of the configuration are written in the figure captions. Tables 3 and 5 report the corresponding minimum, maximum and average running times of the tests.

We make the following observations.

1. RTA(64) and RTA(16) are indistinguishable with respect to the schedulability ratio (Figures 1 and 2). As expected from the (worst-case) analysis, the maximum running time of RTA(ξ) grows linearly with ξ (Tables 2 and 3); the fact that RTA(64) and RTA(16) present identical average running times (up to a millisecond precision) suggests that the halting condition $\boldsymbol{Y}[\nu] = \boldsymbol{Y}[\nu - 1]$ from Definition 8(3) is satisfied with $\nu \leq 16$ for "almost every" task-sets.
2. BON(10) and BON(6) are indistinguishable with respect to the schedulability ratio (Figures 3 and 4). We remark that, according to our results, BON(6) and BON(δ), with δ = 8, 10, 12, 14, have been always found to be indistinguishable with respect to the schedulability ratio on all tested configurations. The maximum and average running time of BON(δ) grows exponentially with δ (Tables 4 and 5).
3. BON(δ) presents a *schedulability threshold* of $m/U \sim 2 - 1/m + 2^{-\delta}$ (that is, no task-set with m/U less than about $2 - 1/m + 2^{-\delta}$ is deemed schedulable by BON(δ)) and a *schedulability saturation* (that is, the schedulability ratio for BON(δ) reaches its maximum value at $m/U \sim 2 - 1/m + 2^{-\delta}$ and then remains constant or even decreases).

4.2 Simulation II

In this simulation, we aim at comparing the behavior of RTA(16), BON(6), RTA-P and BON-P in terms of schedulability ratio and running time. The values $\xi := 16$ and $\delta := 6$ for RTA(ξ) and BON(δ), respectively, have been selected ("following" the results presented in Simulation I) in the attempt to "maximize" the schedulability ratios of these tests on the given configurations.

The results for this simulation are all reported in Figures 5-6 and in Tables 6-7, at page 10 in this paper.

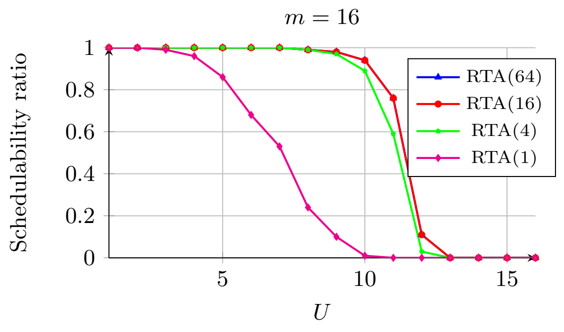
(i) Figure 5 reports the schedulability ratio of the tests as a function of the total utilization U of the task-sets for a platform composed of 16 processors. The values of the configurations are written in the figure caption. Table 6 reports the corresponding minimum, maximum and average running times of the pseudo-polynomial time tests.

(ii) Figure 6 reports the schedulability ratio of the tests as a function of the number of processors m composing the platform; the values of the configuration are written in the figure caption. Table 7 reports the corresponding minimum, maximum and average running times of the pseudo-polynomial time tests.

We make the following observations.

1. RTA(16) outperforms BON(6) in terms of both schedulability ratio (Figures 5-6) and running time (Tables 6-7).[10] The number of task-sets which are deemed schedulable by RTA(16), in Figure 5, is about 275% the corresponding number for BON(6).
2. RTA-P outperforms BON-P in terms of schedulability ratio (Figures 5-6). RTA-P and BON are incomparable w.r.t. the dominance quasi-order, but the number of task-sets which are deemed schedulable by RTA-P, in Figure 5, is about the corresponding number for BON(6).
3. Figures 5-6 and Figures 1-2 suggest that RTA(1) $\equiv$ RTA-P, as boolean functions. Our results did *not* break this equivalence but we were unable to formally derive it.

[10]In fact, according to our results, RTA(16) is already superior to BON(4) in terms of both maximum and average running times.



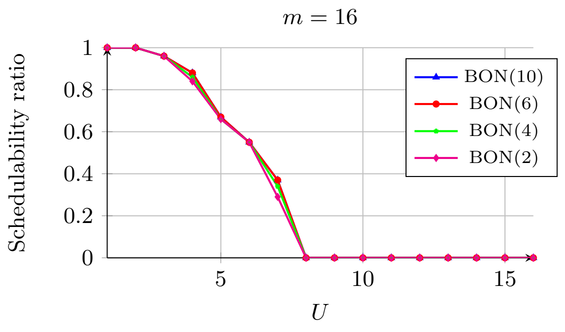


Figure 2: $n = 20$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

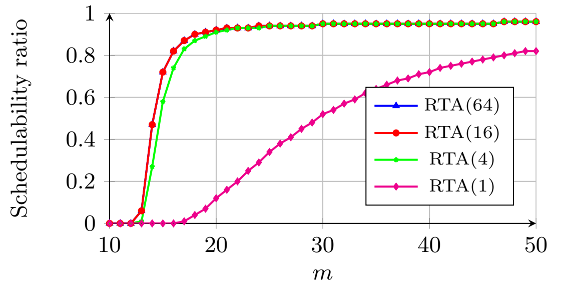
Figure 4: $n = 20$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

	Min (s)	Max (s)	Avg (s)
RTA(64)	0.001	0.397	0.014
RTA(16)	0.001	0.225	0.014
RTA(4)	0.001	0.050	0.009
RTA(1)	0.001	0.014	0.005

Table 2: Running times, in seconds, for the simulations of Figure 2.

	Min (s)	Max (s)	Avg (s)
BON(10)	0.000	17.855	3.357
BON(6)	0.000	1.160	0.214
BON(4)	0.000	0.292	0.051
BON(2)	0.000	0.142	0.012

Table 4: Running times, in seconds, for the simulations of Figure 4.



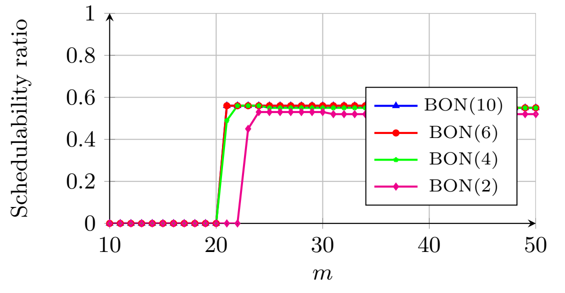


Figure 3: $n = 20$, $U = 10$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

Figure 5: $n = 20$, $U = 10$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

	Min (s)	Max (s)	Avg (s)
RTA(64)	0.001	0.487	0.016
RTA(16)	0.001	0.236	0.016
RTA(4)	0.001	0.054	0.011
RTA(1)	0.001	0.013	0.007

Table 3: Running times, in seconds, for the simulations of Figure 3.

	Min (s)	Max (s)	Avg (s)
BON(10)	0.000	19.495	4.684
BON(6)	0.000	1.284	0.292
BON(4)	0.000	0.316	0.053
BON(2)	0.000	0.069	0.009

Table 5: Running times, in seconds, for the simulations of Figure 5.

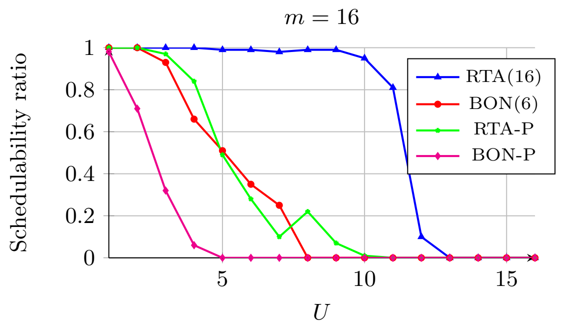


Figure 6: $n = 20$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

	Min (s)	Max (s)	Avg (s)
RTA(16)	0.001	0.191	0.010
BON(6)	0.000	1.312	0.134

Table 6: Running times, in seconds, for the simulations of Figure 6.

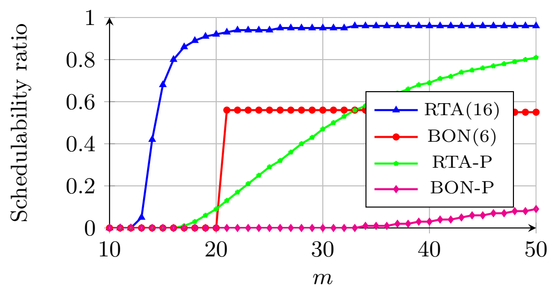


Figure 7: $n = 20$, $U = 10$, $T_{\min} = 100$, $T_{\max} = 1000$, $\alpha_{\min} = 1$, $\alpha_{\max} = 5$, $N_{\min} = 5$, $N_{\max} = 20$, $p_{\text{edge}} = 25$ ($P = 10000$).

	Min (s)	Max (s)	Avg (s)
RTA(16)	0.001	0.208	0.012
BON(6)	0.000	1.306	0.330

Table 7: Running times, in seconds, for the simulations of Figure 7.

5. CONCLUSIONS

We have studied the schedulability problem for sporadic DAG-tasks with arbitrary deadline on multiprocessor platforms proposing an approach based on Response Time Analysis. In particular, a novel upper-bound on the interference, specifically targeted to the considered task model, has been defined for the case of G-EDF and G-DM and two schedulability tests have been derived. The first test is a simple polynomial time test, while the second one is a pseudo-polynomial time test. Simulation results have shown that our tests outperform the tests described in [7] in terms of both schedulability ratio and running time. We also derived resource augmentation bounds for our polynomial time test on single-DAG systems.

6. REFERENCES

[1] B. Andersson and D. de Niz. Analyzing global-EDF for multiprocessor scheduling of parallel tasks. *OPODIS 2012*.

[2] P. Axer, S. Quinton, B. Döbel, and H. Härtig. Response-time analysis of parallel fork-join workloads with real-time constraints. *ECRTS 2013*.

[3] S. Baruah. Improved multiprocessor global schedulability analysis of sporadic DAG task systems. *ECRTS 2014*.

[4] S. Baruah, V. Bonifaci, and A. Marchetti-Spaccamela. The global EDF scheduling of systems of conditional sporadic DAG tasks. *ECRTS 2015*.

[5] S. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, L. Stougie, and A. Wiese. A generalized parallel task model for recurrent real-time processes. *RTSS 2012*.

[6] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *RTS 2005*.

[7] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese. Feasibility analysis in the sporadic DAG task model. *ECRTS 2013*.

[8] H. S. Chwa, J. Lee, K.-M. Phan, A. Easwaran, and I. Shin. Global EDF schedulability analysis for synchronous parallel tasks on multicore platforms. *ECRTS 2013*.

[9] J. Fonseca, V. Nélis, G. Nelissen, and L. M. Pinho. Analysis of self-interference within dag tasks. *RTSOPS 2015*.

[10] J. Fonseca, V. Nelis, G. Raravi, and L. M. Pinho. A multi-DAG model for real-time parallel applications with conditional execution. *SAC 2015*.

[11] J. Kim, H. Kim, K. Lakshmanan, and R. Rajkumar. Parallel scheduling cyber-physical systems: analysis and case study on a self-driving car. *ICCPS 2013*.

[12] K. Lakshmanan, S. Kato, and R. Rajkumar. Scheduling parallel real-time tasks on multi-core processors. *RTSS 2010*.

[13] J. Li, K. Agrawal, C. Lu, and C. Gill. Analysis of global EDF for parallel tasks. *ECRTS 2013*.

[14] J. Li, J.-J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. *ECRTS 2014*.

[15] C. Maia, M. Bertogna, L. Nogueira, and L. M. Pinho. Response-time analysis of synchronous parallel tasks in multiprocessor systems. *RTNS 2014*.

[16] C. Maia, L. Nogueira, and L. M. Pinho. Online admission of parallel real-time tasks. *RTSOPS 2015*.

[17] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. Buttazzo. Response-time analysis of conditional DAG tasks in multiprocessor systems. *ECRTS 2015*.

[18] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *TSE 1997*.

[19] G. Nelissen, V. Berten, J. Goossens, and D. Milojevic. Techniques optimizing the number of processors to schedule multi-threaded tasks. *ECRTS 2012*.

[20] L. Nogueira and L. M. Pinho. Server-based scheduling of parallel realtime tasks. *EMSOFT 2012*.

[21] M. Qamhieh, F. Fauberteau, L. George, and S. Midonnet. Global EDF scheduling of directed acyclic graphs on multiprocessor systems. *RTNS 2013*.

[22] M. Qamhieh, L. George, and S. Midonnet. A stretching algorithm for parallel real-time DAG tasks on multiprocessor systems. *RTNS 2014*.

[23] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu, and C. Gill. Parallel real-time scheduling of DAGs. *TPDS 2014*.

[24] A. Saifullah, J. Li, K. Agrawal, C. Lu, and C. Gill. Multi-core real-time scheduling for generalized parallel task models. *RTSS 2011*.

[25] M. Stigge, P. Ekberg, and W. Yi. The fork-join real-time task model. *ACM SIGBED Review 2013*.