Feasibility Analysis in the Sporadic DAG Task Model

Vincenzo Bonifaci*, Alberto Marchetti-Spaccamela†, Sebastian Stiller‡, and Andreas Wiese§
*Istituto di Analisi dei Sistemi ed Informatica, CNR, Email: vincenzo.bonifaci@iasi.cnr.it
†Università di Roma "La Sapienza", Email: alberto@dis.uniroma1.it
‡Technische Universität Berlin, Email: sebastian.stiller@tu-berlin.de
§Max-Planck-Institut für Informatik Saarbrücken, Email: awiese@mpi-inf.mpg.de

***Abstract*—Real-time systems increasingly contain processing units with multiple cores. To use this additional computational power in hard deadline environments, one needs schedulability tests for task models that represent the possibilities of parallel execution of jobs of a task. A standard model is to represent a (sporadically) recurrent task by a directed acyclic graph (DAG). The nodes of the DAG correspond to the jobs of the task. All such jobs are released simultaneously, have to be completed within some common relative deadline, and some pairs of jobs are linked by a precedence constraint, i.e., an arc of the DAG. This poses new challenges for analyzing whether a task system is feasible, in particular for the commonly used online algorithms Earliest Deadline First (EDF) and Deadline Monotonic (DM). While for ordinary sporadic tasks the required algorithmic techniques are well-understood, despite recent research [1], [3], [11], [13] much remains open in this model.**

In this work, we completely close the gap between the algorithmic understanding of feasibility analysis for the usual sporadic task model and the case where each sporadic task is a DAG. We show for DAG tasks that EDF has a tight speedup bound of $2 - 1/m$, where m is the number of processors, while DM has a speedup bound of at most $3 - 1/m$. Moreover, we present polynomial and pseudopolynomial time tests, of differing effectiveness, for determining whether a set of sporadic DAG tasks can be scheduled by EDF or DM to meet all deadlines on a specified number of processors. We remark that the effectiveness of some of our tests matches the best known algorithms for ordinary sporadic task sets, thus closing the gap.

I. Introduction

The sporadic task model is a well-known model to represent real-time systems based on a finite number of independent recurrent processes or tasks, each of which may generate an unbounded sequence of jobs. Determining how multiple recurrent tasks can be scheduled on a shared uni- or multiprocessor platform is one of the traditional subjects of study in real-time scheduling theory. Different formal models have been proposed for representing such recurrent tasks; these models differ from one another in the restrictions they place on the jobs that may be generated by a single task (see, for example, [5], [8], [9], [10], [14]).

It is well-known that the technological evolution of processor manufacturing is moving away from increasing clock frequencies to increasing the number of cores per processor; as an example we refer to the 2007 Intel Teraflops Research chip with as many as 80 cores. The presence of large core-counts offers new opportunities for executing more computation-intensive workloads in real time. Nowadays it is unclear how the resulting massively parallel multicore CPUs will be structured; in fact, it is not clear whether all the cores will be identical, or there will be different specialized cores to realize different functions, and/or whether some cores will be dedicated to certain functionalities. However, it is likely that in the near future an execution environment will allow for the possibility of having more expressive task models than the relatively simple recurrent task models considered thus far in the real-time scheduling literature. We refer to [6], [13], [14], [15] and to references therein for a thorough discussion of the models. We observe that an important characteristic of the more expressive models is to allow for partial parallelism within a task, as well as for precedence constraints between different parts of the task.

In this paper, we continue the study of a parallel task model, the *sporadic DAG model*, that was introduced in [3], [13] and that considers the preemptive scheduling of a recurrent task. The model generalizes the *fork-join* model that has been introduced in [6] and further generalized in [1], [11], [13]. In the fork-join model, the execution requirement of a task is an alternate sequence of parallel and sequential threads that are represented as sequential and parallel segments; parallel segments need to synchronize before starting execution of the next sequential segment.

In the sporadic DAG model a task is represented as a directed acyclic graph (DAG) $G = (V, E)$; the task repeatedly emits a *dag-job*, which is a set of precedence-constrained sequential jobs. More precisely, in [3] each vertex $v \in V$ of the DAG corresponds to a sequential job, and is characterized by a worst-case execution time (WCET) e_v. Each (directed) edge of the DAG represents a precedence constraint: if $(v, w) \in E$ is a (directed) edge in the DAG, then the job corresponding to vertex v must complete execution before the job corresponding to vertex w may begin execution. Any groups of jobs that are not constrained (directly or indirectly) by precedence constraints among each other may execute in parallel, whenever enough processors are available for them. This implies that jobs of subsequent dag-jobs of the same task can be scheduled in parallel.

When a dag-job is released by the task, it is assumed that all $|V|$ of the corresponding jobs become available for execution simultaneously, subject to the precedence constraints. All $|V|$ jobs that are released at some time-instant t must complete execution by time-instant $t + D$, where D is the (relative) deadline parameter of the task. A minimum interval of duration T must elapse between successive releases of two dag-jobs of

the same task. The duration T is called the period of the task. The above model generalizes the one presented in [13], where implicit deadlines (that is, $D = T$) and unit execution time of each node (that is, $e_v = 1$ for all v) were assumed.

In this paper we consider real-time workloads that can be modeled as a collection of independent sporadic DAGs and that are executed upon a platform comprised of m identical processors. We assume that the platform is fully *preemptive* and that it allows *global interprocessor migration*, although we assume that each job may execute on at most one processor at each instant of time. We study the behavior of two well known scheduling policies: *Earliest Deadline First (EDF)* and *Deadline Monotonic (DM)* [7], [10].

Feasibility, schedulability and speedup bounds. An important requirement of hard real-time systems is to guarantee prior to system run time that all deadlines will be met; such guarantees are given by *schedulability tests*. Since the period parameter T_i of the sporadic DAG task τ_i specifies the minimum, rather than exact, duration that can elapse between the release of successive dag-jobs, a task system may *generate* infinitely many different collections of dag-jobs. A task system Υ is said to be *feasible* on m speed-s processors if a valid schedule exists on m speed-s processors for every collection of dag-jobs that may be generated by the task system. Given a scheduling algorithm A, a task system is said to be *A-schedulable* on m speed-s processors if A meets all deadlines when scheduling any collection of dag-jobs that may be generated by the task system on m speed-s processors.

The problem of testing feasibility of a given DAG task system is highly intractable (NP-hard in the strong sense [16]) even when there is a single DAG task. It is therefore highly unlikely that we will be able to design efficient algorithms for solving the problem exactly, and our objective is therefore to design efficient algorithms that solve the problem *approximately*.

We say that a *scheduling algorithm A has a speedup bound s* if any task system that is feasible on m unit speed processors is A-schedulable on m speed-s processors. Furthermore, an *A-schedulability test has speedup bound s*, if the following holds: Any task system that is feasible on m unit speed processors is determined by the test to be A-schedulable on m speed-s processors. Note, that such a test also gives a positive answer for instances for which there is no schedule on m unit speed processors, but they are A-schedulable on m speed-s processors. In this sense, the value s, i.e., the *speedup bound* of the test, is a metric for quantifying the quality of the approximation of the test.

Previous results. It is known [16] that the preemptive scheduling of a given collection of precedence-constrained jobs (that is, a DAG) on a multiprocessor platform is NP-hard in the strong sense; this intractability result is easily seen to hold for the sporadic DAG model as well.

In the (sequential) sporadic task model there exist schedulability tests with speedup factor of $2-1/m$ when the scheduling algorithm is EDF and $3-1/m$ when the scheduling algorithm is Deadline Monotonic [2], [4].[1]

[1]Note that in [2], [4] it is assumed that subsequent jobs generated by the same task cannot be parallelized.

As already observed, there are several papers that considered models where the execution requirement of a task is an alternate sequence of parallel and sequential threads; namely, a task τ_i is a sequence of s_i segments where the j-th segment, $1 \leq j \leq s_j$, consists of $m_{i,j}$ parallel threads. In [13] the authors analyzed the case of implicit deadlines and all parallel threads with the same worst-case execution requirement and showed a global EDF-schedulability test with speedup 4 and a partitioned DM-schedulability test with speedup 5. The paper also extends the above results to the DAG model in the special case where each node of the DAG has unit execution time.

In [11] the same model consisting of a sequence of parallel and sequential threads is considered, with the assumption that relative deadlines of the tasks are not larger than the corresponding periods. The authors showed a speedup bound of 2 for a certain class of algorithms, which includes PD^2, U-EDF, LLREF and DP-Wrap. In [1], Andersson and de Niz considered a similar model and showed that EDF has a speedup bound of $2 - 1/m$. We remark that the schedulability test provided in [1] is not efficient and no bound on its running time is provided.

Most of the research described in [3] is concerned with the DAG model in the case of a single DAG and $D > T$ (that in the case of a single DAG is the more interesting case). First it is shown that the "synchronous arrival sequence", in which successive dag-jobs are released exactly the period T time-units apart, does not necessarily correspond to the worst-case behavior of a sporadic DAG task; hence, we cannot determine schedulability properties by simply studying this one behavior of the task. Furthermore, [3] also considers the Earliest Deadline First (EDF) scheduling [10], [5] of a sporadic DAG task on identical multiprocessors. It is shown that EDF has a speedup bound no larger than 2 for scheduling a sporadic DAG task. The paper also presents two different schedulability tests for determining whether EDF can schedule a given sporadic DAG task upon a specified identical multiprocessor to meet all deadlines. These tests have different run-time complexity — one has polynomial run-time while the other has run-time pseudopolynomial in the representation of the task — and effectiveness (as quantified, again, by the speedup bound metric).

This paper. The main limitation of [3] is that a single DAG task is considered. So its applicability is limited to systems that execute a single task. The major contribution of this paper is to consider the case of multiple tasks, where each task is specified by a different DAG. The main results of the paper is to show *a $2 - 1/m$ speedup bound for EDF*, thus improving and extending previous results; the bound matches the best bound for a sporadic task system [2], [4] and surprisingly shows that parallel threads and precedence constraints do not influence the effectiveness of EDF. Moreover, in addition to EDF, the analysis is extended to the Deadline Monotonic (DM) scheduling algorithm showing a $3-1/m$ *speedup bound*; also in this case, the speedup bound matches the best bound known for a sequential sporadic task system [2].

Our tests have pseudopolynomial time complexity; for this reason, we complement the pseudopolyomial tests with simple polynomial time sufficient conditions to test EDF and DM schedulability.

The remainder of this paper is organized as follows. In Section II, we formally define the notation and terminology used in describing our task model. In Section III-A we present the speedup bound for EDF. The speedup bound for DM is given in Section III-B. We present and analyze pseudopolynomial time EDF- and DM-schedulability tests in Section IV, that either guarantee that a DAG task system is EDF-schedulable (respectively, DM-schedulable) on m processors of speed $2 - 1/m$ (respectively, $3 - 1/m$) or prove that the system in infeasible on m processors of unit speed.

Finally, in Section V we present simple sufficient schedulability conditions that can easily be tested in polynomial time.

II. Model and definitions

In the *sporadic DAG* model, a task τ_i ($i = 1, \ldots, n$) 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 and T_i are 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 $v \in V_i$ denotes a sequential operation (a "job"). Each job $v \in V_i$ is characterized by a processing time $e_v \in \mathbb{N}$, also known as *worst-case execution time* or WCET. The edges represent dependencies between the jobs: if $(v_1, v_2) \in E_i$, then job v_1 must complete execution before job v_2 can begin execution.
- A *period* $T_i \in \mathbb{N}$. A *release* or arrival of a *dag-job* of the task at time-instant t means that all $|V_i|$ jobs $v \in V_i$ are released at time-instant t; 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 cannot be released prior to time-instant $t + T_i$. We say a job becomes *available* at time t if all its predecessor jobs have completed execution at time t and t is greater or equal than the release date of the job.
- A *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 $t + D_i$.

Throughout this paper we assume that the input consists of a *task system* $\mathcal{T} = (\tau_1, \tau_2, \ldots, \tau_n)$, a collection of n sporadic DAG tasks.

Remark 1. If $D_i > T_i$, then task τ_i may release a dag-job prior to the completion of its previously-released dag-jobs. We do *not* require that all jobs of a dag-job complete execution before jobs of the next dag-job can start executing.

Remark 2. We assume that each job requires an integer number of units of execution time (less or equal to its WCET). Note, however, that even though we assume the execution times to be integers, when analyzing algorithms with increased speed (e.g., as we will do for EDF with speed $2 - 1/m$ in Section III), a job could be completed at a non-integral point in time, even if it is never preempted. Therefore, in the analysis jobs may be started or preempted at fractional timepoints.

Some additional notation and terminology:

- A *chain* 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 in G_i, $1 \leq j < k$. The *length* of this chain is defined to be the sum of the WCETs of all its vertices: $\sum_{j=1}^{k} e_{v_j}$.
- We denote by $\mathrm{len}(G_i)$ the length of the longest chain in G_i. Note that $\mathrm{len}(G_i)$ can be computed in time linear in the number of vertices and the number of edges in the acyclic graph G_i, by first obtaining a topological order of the vertices of the graph and then running a straightforward dynamic program.
- We define $\mathrm{vol}(G_i) = \sum_{v \in V_i} e_v$. That is, $\mathrm{vol}(G_i)$ is the total WCET of each dag-job. It is clear that $\mathrm{vol}(G_i)$ can be computed in time linear in the number of vertices in G_i.
- We denote the length of a time interval I by $|I|$.

III. Speedup bounds for collections of jobs

This section considers what we call a *normal collection* J of jobs. The job sequence generated by a DAG task system $\mathcal{T}$ is a normal collection of jobs. Arguing about job collections instead of DAG task systems makes our results slightly more general and—more importantly—cleaner and easier to present. We now define the normal collections model.

Assume we are given m identical processors. A job collection J is a set of jobs that are revealed online over time, i.e., a job $j \in J$ becomes known upon the release date of j. Each job $j \in J$ is characterized by a release date $r_j \in \mathbb{N}_0$, an absolute deadline $d_j \in \mathbb{N}$, an unknown execution time $e_j \in \mathbb{N}$, and a set of previous jobs J_j which are exactly the jobs which have to be finished before j can become available (the *predecessors* of j). Note that the actual execution time e_j of a job is discovered by the scheduler only after the job signals completion.

We call such a collection of jobs J a *normal* collection of jobs if we also have for every predecessor job j of job k that $r_j = r_k$ and $d_j = d_k$. Observe that every collection of jobs generated by a sporadic DAG task system is normal, since all jobs that constitute a certain dag-job have identical release date and deadline. A job j is *available at time* t if $t \geq r_j$ and all jobs in J_j have been completed, while j is not yet completed.

Given J, suppose that infinitely many (or, say, $|J|$) processors of unit speed were available. In this case it is easy to see that the following A_∞ scheduling algorithm is optimal: just allocate one processor to each job and schedule each job as early as possible. Denote by S_∞ the corresponding greedy schedule; it is easy to see that the following claims hold:

- S_∞ starts and ends processing jobs always at integral time points.
- S_∞ dominates all feasible schedules of J, in the sense that at any point in time and for any job it has processed at least as much of that job as any

feasible schedule of J upon a platform of m unit speed processors.

Below, we will analyze EDF and DM by comparing them to A_∞.

A. Analysis of EDF

The EDF scheduler, at any time, processes the m jobs with minimum deadline which are currently available (breaking ties arbitrarily).

Lemma 3. *Consider a normal collection J of jobs and let $\alpha \geq 1$. Then at least one of the following holds:*

(i) *all jobs in J are completed within their deadline under EDF on m processors of speed α, or*
(ii) *J is infeasible under A_∞, or*
(iii) *there is an interval I such that any feasible schedule for J must finish more than $(\alpha m - m + 1) \cdot |I|$ units of work within I.*

Proof: Suppose that both (i) and (ii) do not hold, that is, under EDF on m speed-α processors some job j fails its deadline d_j, and J is feasible if we are given a sufficiently large number of processors. Recall A_∞, the idealized greedy algorithm using infinitely many (or, say, $|J|$) processors of unit speed.

Without loss of generality, we can assume that there is no job j' in the instance with $d_{j'} > d_j$ (otherwise, since J is normal the removal of j' does neither affect EDF nor A_∞). Let t^* denote the latest point in time such that at any time $t \in [0, t^*]$ EDF with α speedup has processed at least as much of *every* job as A_∞ at time t. Such a time exists, since $t^* = 0$ satisfies this property. As (i) and (ii) are false, we also have $t^* < d_j$.

We claim that within $I = [t^*, d_j]$ EDF finishes more than $(\alpha m - m + 1) \cdot |I|$ units of work. This claim gives the lemma due to the following reasoning. If EDF finishes more than $(\alpha m - m + 1) \cdot |I|$ units of work, then the non failing algorithm A_∞ finishes at least the same amount of work during I (by construction of I). Hence *every* feasible schedule has to finish more than $(\alpha m - m + 1) \cdot |I|$ units of work during I, since it could not do more than A_∞ (and thereby more than EDF) before I.

We now prove the claim on the amount of work done by EDF in I. Denote by X the total length of the intervals within I where in the EDF schedule all m processors are busy. Define $Y = |I| - X$. We distinguish two cases. First assume that $\alpha \cdot Y \geq |I|$. Denote by $Y_1, ..., Y_k \subseteq I$ all subintervals of I where not all processors are busy. We define t' such that $\alpha \cdot |[t^*, t'] \cap \bigcup_i Y_i| = \lceil t^* \rceil - t^*$. During all points in time within $[t^*, t'] \cap \bigcup_i Y_i$ all jobs are available for EDF which are scheduled by A_∞ during $[t^*, \lceil t^* \rceil]$. Since during all these points in time EDF does not use all processors and runs the processors with speed α, by time t' it has processed at least as much of every job as A_∞ by time $\lceil t^* \rceil$.

Next, define timepoints t_i, $i = 0, ..., d_j - \lceil t^* \rceil$ such that $\alpha \cdot |[t^*, t_i] \cap \bigcup_i Y_i| = \lceil t^* \rceil - t^* + i$ for each i. We prove by induction that up to time t_i EDF has processed as much of every job as A_∞ by time $\lceil t^* \rceil + i$. The case $i = 0$ was proven above. Now suppose that the claim is true for some value i. Then at each timepoint during $[t_i, t_{i+1}) \cap \bigcup_i Y_i$ all jobs are available for EDF that A_∞ works on during $[\lceil t^* \rceil + i, \lceil t^* \rceil + i + 1)$. Since during all these timepoints EDF does not use all processors and runs the processors with speed α, by time t_{i+1} it has processed at least as much of every job as A_∞ by time $\lceil t^* \rceil + i + 1$. By induction the claim is true for $i^* = d_j - \lceil t^* \rceil$ and hence at time $\lceil t^* \rceil + i^* = d_j$ EDF has finished as much of every job as A_∞. This yields a contradiction since we assumed that A_∞ is feasible and EDF is not.

Now assume that $\alpha \cdot Y < |I|$. Hence, in the interval I EDF finishes at least

$$\begin{aligned} \alpha m \cdot X + \alpha \cdot Y &= \alpha m \cdot (|I| - Y) + \alpha \cdot Y \\ &= \alpha m \cdot |I| - \alpha m Y + \alpha \cdot Y \\ &> \alpha m \cdot |I| - m \cdot |I| + |I| \\ &= (\alpha m - m + 1) \cdot |I| \end{aligned}$$

units of work, and by construction of I, any feasible schedule has to finish during the interval I all work that EDF finishes during I. ∎

The above lemma implies the following theorem if we choose $\alpha = 2 - 1/m$.

Theorem 4. *Any normal collection of jobs that is feasible on m processors of unit speed is EDF-schedulable on m processors of speed $2 - 1/m$.*

Proof: Since we assumed the instance to be feasible, it is in particular feasible on a sufficiently high number of processors of unit speed. Also, the instance admits a valid schedule which finishes in any interval I at most $m \cdot |I|$ units of work. Note that if $\alpha = 2 - 1/m$ then $(\alpha m - m + 1) \cdot |I| = (2m - 1 - m + 1) \cdot |I| = m|I|$. Hence, Lemma 3 implies that EDF finishes all jobs by their respective deadline. ∎

Since every collection of jobs generated by a sporadic DAG task system is normal, we obtain the following corollary.

Corollary 5. *Any DAG task system that is feasible on m processors of unit speed is EDF-schedulable on m processors of speed $2 - 1/m$.*

The above bound is tight: examples are known, even without precedence constraints, of feasible collections of jobs that are not EDF-schedulable unless the speedup is at least $2 - 1/m$ [12].

B. Analysis of DM

The relative deadline of a job j is the difference $(d_j - r_j)$ between its deadline and release date. At any time, the DM scheduler processes the m jobs with minimum relative deadline which are currently available (breaking ties arbitrarily).

Lemma 6. *Consider a normal collection J of jobs and let $\alpha \geq 1$. Then at least one of the following holds:*

(i) *all jobs in J are completed within their deadline under DM on m processors of speed α, or*
(ii) *J is infeasible under A_∞, or speed, or*
(iii) *there is an interval I such that any feasible schedule for J must finish more than $(\alpha m - m + 1) \cdot |I|/2$ units of work within I.*

Proof: Suppose that both (i) and (ii) do not hold, that is, under DM on m speed-α processors some job j fails its deadline d_j, and J is feasible if we are given a sufficiently large number of processors. We again will consider the idealized greedy algorithm A_∞.

Without loss of generality, we can assume that there is no job j' in the instance with $d_{j'} > 2d_j - r_j$ where r_j is the release date of job j. In fact assume that in J there is a job j' that has deadline later than $2d_j - r_j$. If the relative deadline of job j' is at most $d_j - r_j$ then the job is released after d_j and we can ignore it; if the relative deadline of job j' is greater than $d_j - r_j$ then the execution of job j is not interrupted by job j' and hence by removing j' from J we obtain a smaller collection J' that violates the claim. Let t^* denote the latest point in time such that at any time $t \in [0, t^*]$, DM has processed at least as much of *every* job as A_∞ at time t. Such a time exists, since $t^* = 0$ satisfies this property. Also, it must hold that $t^* < d_j$.

Let $\hat{t} = \min(t^*, r_j)$, $I = [\hat{t}, 2d_j - r_j]$ and $\hat{I} = [\hat{t}, d_j]$. Observe that the definition of DM implies that during $\hat{I}$ DM executes only jobs that have their deadline in I.

We claim that, within $\hat{I}$, DM finishes more than $(\alpha m - m + 1) \cdot |\hat{I}|$ units of work, hence A_∞ finishes at least the same amount of work during I (by construction of I and $\hat{I}$) and, hence, *every* feasible schedule has to finish more than $(\alpha m - m + 1) \cdot |\hat{I}|$ units of work during I.

Analogously to the case of EDF, we can show by contradiction that, within $\hat{I}$, DM finishes more than $(\alpha m - m + 1) \cdot |\hat{I}|$ units of work. Again, we denote by X the total length of the intervals within $\hat{I}$ where in the DM schedule all m processors are busy. Define $Y = |\hat{I}| - X$. As in the proof of EDF we distinguish two cases. First, if $\alpha \cdot Y \geq |\hat{I}|$, by the same argument as in the proof for EDF it is possible to show that DM has finished as much of every job as A_∞. This yields a contradiction since we assumed that A_∞ is feasible and DM is not.

If $\alpha \cdot Y < |\hat{I}|$, as in the proof of EDF it follows that during $\hat{I}$ DM finishes at least

$$\alpha m \cdot X + \alpha \cdot Y \quad > \quad (\alpha m - m + 1) \cdot |\hat{I}|$$

units of work, and by construction of I, any feasible schedule has to finish during the interval I all work that DM finishes during $\hat{I}$. Since $|\hat{I}| \geq |I|/2$, the lemma follows. ∎

The above lemma implies the following theorem if we choose $\alpha = 3 - 1/m$.

Theorem 7. *Any normal collection of jobs that is feasible on m processors of unit speed is DM-schedulable on m processors of speed $3 - 1/m$.*

Proof: Since we assumed the instance to be feasible, it is in particular feasible on a sufficiently high number of processors of unit speed. Also, the instance admits a valid schedule which finishes in any interval I at most $m \cdot |I|$ units of work. Note that if $\alpha = 3 - 1/m$ then $(\alpha m - m + 1) \cdot |I|/2 = (3m - 1 - m + 1) \cdot |I|/2 = m|I|$. Hence, Lemma 6 implies that DM finishes all jobs by their respective deadline. ∎

Corollary 8. *Any DAG task system that is feasible on m processors of unit speed is DM-schedulable on m processors of speed $3 - 1/m$.*

IV. Pseudopolynomial time tests with bounded speedup

In the following we present a pseudopolynomial time test for both EDF- and DM-feasibility that is based on a characterization of the work that a feasible instance requires.

Recall the definition of A_∞ from Section III. Suppose we are given a set $\mathcal{T}$ of sporadic DAG tasks. Lemma 3 implies that, in order to assert that EDF feasibly schedules any job sequence J of $\mathcal{T}$ when given speed α, it suffices to ensure that for any such job sequence J,

- A_∞ is feasible for J, and
- there is no interval I during which any feasible schedule for J must finish more than $(\alpha m - m + 1) \cdot |I|$ units of work.

On the other hand, if any of the two conditions fail (with $\alpha \geq 2 - \frac{1}{m}$) then the system is infeasible on machines with unit speed. Using Lemma 6 allows a similar reasoning for DM.

Remark 9. Observe that both conditions are monotone in the execution times of the job sequence. That is, if they are satisfied by a job sequence with some execution times, they are also satisfied by a similar job sequence with decreased execution times. This allows us to focus on the WCETs of the tasks when verifying the conditions.

Condition 1. It is easy to check whether A_∞ is feasible for every job sequence of $\mathcal{T}$: this is the case if and only if $\text{len}(G_i) \leq D_i$ for all $i = 1, \ldots, n$. This condition can be verified by n comparisons, that is, in linear time.

Condition 2. For the remainder of this section we can focus on verifying the second condition. For a sequence of jobs J and an interval I, we denote by $\text{work}^J(I)$ the amount of work done by A_∞ during I on the jobs in J whose deadlines are in I. The motivation for this quantity is that any feasible schedule with unit speed machines has to finish at least $\text{work}^J(I)$ units of work during I.

Definition 10. Given a sporadic DAG task system $\mathcal{T}$, let $\text{gen}(\mathcal{T})$ be the set of job sequences that may be generated by $\mathcal{T}$, and define

$$\text{work}_{\mathcal{T}}(t) = \sup_{J \in \text{gen}(\mathcal{T})} \sup_{t_0 \geq 0} \text{work}^J([t_0, t_0 + t]).$$

$$\lambda_{\mathcal{T}} = \sup_{t \in \mathbb{N}} \frac{\text{work}_{\mathcal{T}}(t)}{t}.$$

Intuitively, the quantity $\lambda_{\mathcal{T}}$ denotes the maximum "workload density" which an interval can have. In particular, if $\lambda_{\mathcal{T}} > m$ then the system is infeasible since there is a job sequence for $\mathcal{T}$ and an interval I during which more than $m \cdot |I|$ units of work have to be finished by any schedule.

We want to compute the maximum workload density to test the second condition. We cannot afford to compute it with perfect precision. However, computing the workload density up

to an ϵ-error is sufficient, because of the next lemma. It shows that with a certain speedup EDF and DM are still feasible, if the workload density is a bit higher than m. So, if we can at least distinguish whether it is greater than m, or less-or-equal to a bit more than m, then we can either say EDF and DM with speedup are feasible, or no feasible unit speed schedule exists.

Lemma 11. *Let $\mathcal{T}$ be a sporadic DAG task system. Let $\epsilon \geq 0$ and suppose that $\mathrm{work}_{\mathcal{T}}(t) \leq (1+\epsilon)mt$ for any $t \in \mathbb{N}$ and that $\mathcal{T}$ is feasible on a sufficiently large number of unit-speed processors. Then $\mathcal{T}$ is EDF-schedulable on m processors of speed $2 - 1/m + \epsilon$ and DM-schedulable on m processors of speed $3 - 1/m + \epsilon$.*

Proof: We give the proof for EDF; the one for DM follows by exactly the same arguments and is therefore omitted. Suppose that EDF fails on some job sequence $J \in \mathrm{gen}(\mathcal{T})$ when running at speed $2 - 1/m + \epsilon$. Then by Lemma 3 there is an interval I in which any feasible schedule must finish more than

$$(\alpha m - m + 1) \cdot |I| = (2m - 1 + \epsilon m - m + 1)|I| = (1+\epsilon)m|I|$$

units of work. This contradicts that $\mathrm{work}_{\mathcal{T}}(|I|) \leq (1+\epsilon)m|I|$. ∎

Therefore, in order to approximately test the feasibility of $\mathcal{T}$ it suffices to estimate $\lambda_{\mathcal{T}}$. We summarize this in the following lemma.

Lemma 12. *Let $\epsilon \geq 0$ and $\hat{\lambda}_{\mathcal{T}}$ be such that $\lambda_{\mathcal{T}}/(1+\epsilon) \leq \hat{\lambda}_{\mathcal{T}} \leq \lambda_{\mathcal{T}}$. Assume that $\mathcal{T}$ is feasible on a sufficiently large number of unit-speed processors. Then*

(i) *if $\hat{\lambda}_{\mathcal{T}} > m$, $\mathcal{T}$ is infeasible on m unit speed processors;*

(ii) *if $\hat{\lambda}_{\mathcal{T}} \leq m$, $\mathcal{T}$ is EDF-schedulable on m speed-$(2 - 1/m + \epsilon)$ processors and DM-schedulable on m speed-$(3 - 1/m + \epsilon)$ processors.*

Proof: In case (i), we have $\lambda_{\mathcal{T}} \geq \hat{\lambda}_{\mathcal{T}} > m$. Thus, there is a job collection $J \in \mathrm{gen}(\mathcal{T})$ and an interval I such that $\mathrm{work}^J(I) > m|I|$, hence $\mathcal{T}$ is not feasible on m unit speed machines.

In case (ii), we have $\lambda_{\mathcal{T}} \leq (1+\epsilon)\hat{\lambda}_{\mathcal{T}} \leq (1+\epsilon)m$. Thus, Lemma 11 yields the claim. ∎

Given a DAG task set $\mathcal{T}$, a *$(1+\epsilon)$-approximation algorithm for $\lambda_{\mathcal{T}}$* is an algorithm computing a value $\hat{\lambda}_{\mathcal{T}}$ which fulfills

$$\lambda_{\mathcal{T}}/(1+\epsilon) \leq \hat{\lambda}_{\mathcal{T}} \leq \lambda_{\mathcal{T}}.$$

In other words, it computes a value not larger than the true maximum work density, but also not much smaller than it. Note that these are exactly the conditions required in Lemma 12. Thus, we can reformulate the lemma:

Corollary 13. *Let $\epsilon \geq 0$. A $(1+\epsilon)$-approximation algorithm for $\lambda_{\mathcal{T}}$ yields an EDF-schedulability test for $\mathcal{T}$ with speedup $2 - 1/m + \epsilon$ and a DM-schedulability test for $\mathcal{T}$ with speedup $3 - 1/m + \epsilon$.*

Approximation of $\lambda_{\mathcal{T}}$. We will now construct such $(1+\epsilon)$-approximation algorithm for $\lambda_{\mathcal{T}}$, for any given $\epsilon > 0$. By Corollary 13, this allows to test the second condition for speedup factors arbitrarily close to $2 + 1/m$ for EDF and arbitrarily close to $3 + 1/m$ for DM. The running-time of the $(1+\epsilon)$-approximation algorithm depends on ϵ. Thus, by increasing the running time of the test, we decrease the required speedup factor.

Recall that $\lambda_{\mathcal{T}}$ represents the maximum relative load of an interval (over all possible job sequences). Given an interval, its total load is the sum of the loads caused by the tasks $\tau_1, \ldots, \tau_n$. Since the tasks are independent of each other, we can equivalently write

$$\lambda_{\mathcal{T}} = \sup_{t \in \mathbb{N}} \frac{\sum_{i=1}^{n} \mathrm{work}_i(t)}{t}$$

where $\mathrm{work}_i(t)$ is the maximum amount of work that may be done by A_∞ on jobs of task τ_i that are due in an interval of length t (i.e., the maximum load caused by task τ_i during an interval of length t). This maximum is achieved when the deadline of some dag-job of τ_i coincides with the rightmost endpoint of the interval, and the other dag-jobs of τ_i are released as closely as possible. That is, if the interval is (without loss of generality) $[t_0, t_0 + t]$, then there is

- one dag-job with release date $t_0 + t - D_i$ and deadline $t_0 + t$,
- one dag-job with release date $t_0 + t - D_i - T_i$ and deadline $t_0 + t - T_i$,
- one dag-job with release date $t_0 + t - D_i - 2T_i$ and deadline $t_0 + t - 2T_i$,
- $\ldots$
- in general, one dag-job with release date $t_0 + t - D_i - kT_i$, up to a k such that $t_0 + t - (k+1)T_i \leq t_0$ (more dag-jobs would not contribute to the amount of work done by A_∞ during $[t_0, t_0 + t]$).

As a consequence, $\mathrm{work}_i(t)$ is piecewise linear as a function of t, with a number of pieces that is proportional to $|V_i| \cdot t/T_i$, as each dag-job is responsible for at most $|V_i|$ pieces.

For our purposes it suffices to approximately compute $\sup_{t \in \mathbb{N}} \mathrm{work}_i(t)/t$ for each task τ_i. In the next lemma, we first prove some (rough) upper and lower bounds for the quantity $\mathrm{work}_i(t)$.

Lemma 14. *For any task $\tau_i = (G_i, D_i, T_i)$,*

$$\mathrm{work}_i(t) \geq \max\left(\left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor, 0\right) \cdot \mathrm{vol}(G_i), \quad (1)$$

$$\mathrm{work}_i(t) \leq \left\lceil \frac{t}{T_i} \right\rceil \cdot \mathrm{vol}(G_i). \quad (2)$$

Proof: (1): there can be as many as $\lfloor (t + T_i - D_i)/T_i \rfloor$ releases of τ_i-dag-jobs in an interval of length t whose release date and deadline fall within the interval; each of them contributes $\mathrm{vol}(G_i)$ to the work function.

(2): there cannot be more than $\lceil t/T_i \rceil$ releases of τ_i-dag-jobs in an interval of length t whose deadline falls within the interval. These dag-jobs are the only ones that contribute an amount of work larger than 0. ∎

The number of linear pieces of the function $\mathrm{work}_i(t)$ can be very large; so it is not clear how to handle this function efficiently. Therefore, we approximate $\mathrm{work}_i(t)$ by a function $\hat{\mathrm{w}}_i(t)$ defined as follows:

$$\hat{\mathrm{w}}_i(t) = \begin{cases} \mathrm{work}_i(t) & \text{if } t \le T_i/\epsilon + (1+1/\epsilon)D_i \\ \frac{t-D_i}{T_i}\,\mathrm{vol}(G_i) & \text{if } t > T_i/\epsilon + (1+1/\epsilon)D_i. \end{cases}$$

Lemma 15. *The piecewise linear function $\hat{\mathrm{w}}_i$ has $O(\frac{1}{\epsilon}\cdot|V_i|\cdot(1+\frac{D_i}{T_i}))$ many linear pieces.*

Proof: Define $T_i^* = T_i/\epsilon + (1+1/\epsilon)D_i$. During the interval (T_i^*, ∞) the function $\hat{\mathrm{w}}_i(t)$ is linear by definition (i.e., has only one linear piece). For the interval $[0, T_i^*]$ observe that $\hat{\mathrm{w}}_i(t)$ is piecewise linear and continuous and it can change its slope only when a dag-job has finished processing. The number of dag-jobs released during $[0, T_i^*]$ is bounded by $|V_i|\cdot\lceil T_i^*/T_i\rceil = O(\frac{1}{\epsilon}\cdot|V_i|\cdot(1+\frac{D_i}{T_i}))$, which implies the claim. ∎

Summing over all tasks, we get:

Proposition 16. *The function $\hat{\mathrm{w}}$ is piecewise linear and has $O(\frac{1}{\epsilon}\cdot\sum_{i=1}^n|V_i|\cdot\max_{i=1}^n(1+\frac{D_i}{T_i}))$ many linear pieces.*

We will now use the function $\hat{\mathrm{w}}(t) = \sum_{i=1}^n \hat{\mathrm{w}}_i(t)$ instead of the term $\sum_{i=1}^n \mathrm{work}_i(t)$ to (approximately) compute $\lambda_{\mathcal{T}}$. The next lemma shows that $\hat{\mathrm{w}}_i(t)$ approximates $\mathrm{work}_i(t)$ sufficiently well, implying that also $\hat{\mathrm{w}}(t)$ is close to $\mathrm{work}(t)$.

Lemma 17. *For all $i = 1, \ldots, n$ and all $t \in \mathbb{N}$,*

$$\frac{1}{1+\epsilon}\mathrm{work}_i(t) \le \hat{\mathrm{w}}_i(t) \le \mathrm{work}_i(t).$$

Proof: First observe that $\mathrm{work}_i(t) \ge \hat{\mathrm{w}}_i(t)$, since for all $t > T_i/\epsilon + (1+1/\epsilon)D_i$, by (1),

$$\frac{\mathrm{work}_i(t)}{\mathrm{vol}(G_i)} \ge \left\lfloor \frac{t+T_i-D_i}{T_i} \right\rfloor \ge \frac{t+T_i-D_i}{T_i} - 1 = \frac{t-D_i}{T_i} = \frac{\hat{\mathrm{w}}_i(t)}{\mathrm{vol}(G_i)}.$$

Moreover, using (2),

$$\frac{\mathrm{work}_i(t)}{\hat{\mathrm{w}}_i(t)} \le \frac{\lceil t/T_i\rceil}{\frac{t-D_i}{T_i}} \le \frac{t/T_i+1}{t/T_i - D_i/T_i} = \frac{t+T_i}{t-D_i} \le \frac{(D_i+T_i)/\epsilon + D_i + T_i}{(D_i+T_i)/\epsilon + D_i - D_i} = 1+\epsilon.$$

∎

Corollary 18. *For all $t \in \mathbb{N}$, $\frac{1}{1+\epsilon}\mathrm{work}(t) \le \hat{\mathrm{w}}(t) \le \mathrm{work}(t)$.*

Finally, we show that for piecewise linear functions with few pieces, we can compute $\sup_{t\in\mathbb{N}} f(t)/t$ efficiently which together with the above preparation allows us to estimate $\mathcal{T}$ and eventually to infer EDF- or DM-schedulability.

Lemma 19. *Let $f : \mathbb{N} \to \mathbb{N}$ be a piecewise linear function with K linear pieces and assume we can compute $\lim_{t\to\infty} f(t)/t$. Then the value $\sup_{t\in\mathbb{N}} f(t)/t$ can be found by evaluating f in $O(K)$ points.*

Proof: Let $[a, b]$ be a piece of f, that is, a maximal interval in which f is linear. Then $f(t)/t$ is monotone in $[a, b]$, so that $\max(f(a)/a, f(b)/b) \ge f(t)/t$ for all $t \in [a, b]$. Therefore, to compute $\sup_{t\in\mathbb{N}} f(t)/t$ it suffices to compute the value of f in $K+1$ points (one of these "points" is $t = \infty$). ∎

We can now conclude:

Theorem 20. *Let $\epsilon > 0$. There is a pseudopolynomial time EDF-schedulability test with speedup $2 - 1/m + \epsilon$, and a pseudopolynomial time DM-schedulability test with speedup $3 - 1/m + \epsilon$.*

Proof: After combining Corollary 13, Corollary 16, Corollary 18 and Lemma 19, it only remains to show that each $\hat{\mathrm{w}}_i(t)$ can be evaluated in pseudopolynomial time for any t. This is clear from the definition of $\hat{\mathrm{w}}_i$ when $t > T_i/\epsilon + (1+1/\epsilon)D_i$. When $t \le T_i/\epsilon + (1+1/\epsilon)D_i$, notice that there can be $O(1+D_i/T_i)$ dag-jobs that contribute only partially (less than $\mathrm{vol}(G_i)$) to $\hat{\mathrm{w}}_i(t)$. For each of them, the exact amount of contributed work can be computed in polynomial time. ∎

V. Simple sufficient conditions for schedulability

We complement the results of the previous sections with two sufficient conditions for EDF- and DM-schedulability, respectively, that can be easily checked in polynomial time.

Given a sporadic DAG task system, w.l.o.g. we assume that the DAG-tasks τ_i are ordered according to nondecreasing D_i (breaking ties arbitrarily).

A. EDF-schedulability

Theorem 21. *Assume a sporadic DAG task system satisfies the following conditions:*

i) $\mathrm{len}(G_k) \le D_k/3$, $k = 1, 2, \ldots, n$,
ii) *for each k, $k = 1, 2, \ldots, n$,*

$$\sum_{i:T_i\le D_k} \mathrm{vol}(G_i)/T_i + \sum_{i:T_i>D_k} \mathrm{vol}(G_i)/D_k \le (m+1/2)/3.$$

Then the system is EDF-schedulable on m unit-speed processors.

Proof: Suppose by contradiction that EDF fails to meet some deadline while scheduling some sequence of dag-jobs released by a sporadic task τ_k. Let j be the first job of task τ_k that misses its deadline d_j. W.l.o.g. we assume that there are no jobs with a deadline later than d_j. Consider the interval $I = [r_j, d_j)$. Denote by X the total amount of time during I where all processors are busy. Let $Y = (d_j - r_j) - X = D_k - X$, i.e., Y denotes the total amount of time in I during which not all processors are busy.

We first observe that $Y \le D_k/3$. This follows from the observation that whenever a processor is idle, EDF must be executing a job belonging the longest chain of the last dag-job released by τ_k and hence $Y \le \mathrm{len}(G_k)$, which is assumed to be at most $D_k/3$ (condition (i)).

Condition $Y \le D_k/3$ implies that $X \ge 2D_k/3$. Now since the total amount of execution occurring over the interval I is greater or equal to $(mX+Y)$, we conclude that the total work done by EDF during I is greater or equal to $(2m+1)D_k/3$.

Now recall (2) and observe that the total amount of work due in I is bounded above by

$$\begin{aligned}
&\sum_{i:T_i \le D_k} \left\lceil \frac{D_k}{T_i} \right\rceil \mathrm{vol}(G_i) + \sum_{i:T_i > D_k} \mathrm{vol}(G_i) \\
&\le 2D_k \left(\sum_{i:T_i \le D_k} \mathrm{vol}(G_i)/T_i + \sum_{i:T_i > D_k} \mathrm{vol}(G_i)/D_k \right) \\
&\le \frac{2m+1}{3} D_k
\end{aligned}$$

where we have used condition (ii) and the fact that $\lceil x \rceil \le 2x$ when $x \ge 1$. This contradicts the assumption that EDF fails and completes the proof of the theorem. ■

B. DM-schedulability

Theorem 22. *Assume a sporadic DAG task system satisfies the following conditions:*

(i) $\mathrm{len}(G_k) \le D_k/5,\ k = 1, 2, \ldots, n,$
(ii) *for each* $k,\ k = 1, 2, \ldots, n,$

$$\sum_{i:T_i \le 2D_k} \mathrm{vol}(G_i)/T_i + \\ + \sum_{i:T_i > 2D_k} \mathrm{vol}(G_i)/4D_k \le (m + 1/4)/5.$$

Then the system is DM-schedulable on m unit-speed processors.

Proof: Suppose by contradiction that DM fails to meet some deadline while scheduling some sequence of dag-jobs released by a sporadic task τ_k. Let j be the first job of task τ_k that misses its deadline d_j in the minimal instance S that violates the theorem, i.e. S is the instance with the smallest number of jobs that violates the theorem. W.l.o.g. we assume that there are no jobs with a deadline later than $2d_j - r_j$.

Consider the intervals $\hat{I} = [r_j, d_j)$ and $I = [r_j, 2d_j - r_j)$; the crucial observation in this case is that, during $\hat{I}$, DM processes jobs that have their deadline in I.

Denote by X the total amount of time during $\hat{I}$ when all processors are busy according to the DM schedule. Let $Y = (d_j - r_j) - X = D_k - X$, i.e., Y denotes the total amount of time in $\hat{I}$ during which not all processors are busy.

We first observe that $Y \le D_k/5$. This follows from the observation that whenever a processor is idle, DM must be executing a job belonging to the longest chain of the last job released by τ_k and hence $Y \le \mathrm{len}(G_k)$, which is assumed to be at most $D_k/5$.

Condition $Y \le D_k/5$ implies that $X \ge 4D_k/5$. Now since the total amount of execution occurring over the interval $\hat{I}$ is greater or equal to $(mX + Y)$, we conclude that the total work done by DM during $\hat{I}$ is greater or equal to $(4m + 1)D_k/5$.

Now recall (2) and observe that the total amount of work due in I is bounded above by

$$\begin{aligned}
&\sum_{i:T_i \le 2D_k} \left\lceil \frac{2D_k}{T_i} \right\rceil \mathrm{vol}(G_i) + \sum_{i:T_i > 2D_k} \mathrm{vol}(G_i) \\
&\le 4D_k \left(\sum_{i:T_i \le 2D_k} \mathrm{vol}(G_i)/T_i + \sum_{i:T_i > 2D_k} \mathrm{vol}(G_i)/4D_k \right) \\
&\le \frac{4m+1}{5} D_k
\end{aligned}$$

where we have used the fact that $\lceil 2x \rceil \le 4x$ when $x \ge 1/2$. This contradicts the assumption that DM fails and completes the proof of the theorem. ■

When the DAG task system satisfies $D_k \le T_k$ for all tasks τ_k, the following theorem provides a slightly better guarantee.

Theorem 23. *Assume a sporadic DAG task system satisfies the following conditions:*

(i) $\mathrm{len}(G_k) \le D_k/4,\ k = 1, 2, \ldots, n,$
(ii) *for each* $k,\ k = 1, 2, \ldots, n,$

$$\sum_{i:T_i \le 2D_k} \mathrm{vol}(G_i)/T_i + \\ + \sum_{i:T_i > 2D_k} \mathrm{vol}(G_i)/D_k \le (m + 1/3)/4,$$

(iii) $D_k \le T_k,\ k = 1, 2, \ldots, n.$

Then the system is DM-schedulable on m unit-speed processors.

Proof: The proof is similar to the proof above. Suppose by contradiction that DM fails to meet some deadline while scheduling some sequence of dag-jobs released by a sporadic task τ_k. Let j be the first job of task τ_k that misses its deadline d_j. W.l.o.g. we assume that there are no jobs with a release later than d_j. Consider the intervals $\hat{I} = [r_j, d_j)$; in this case the crucial observation is that, during $\hat{I}$, DM processes jobs that have their deadline in $\hat{I}$ or are released in $\hat{I}$. Since the task system satisfies (iii) it follows that for each DAG task there is at most one job that is released in $\hat{I}$ that is not due in $\hat{I}$.

As in the previous proof we denote by X the total amount of time during $\hat{I}$ where all processors are busy according to a DM schedule. Let $Y = (d_j - r_j) - X = D_k - X$, i.e., Y denotes the total amount of time in $\hat{I}$ during which not all processors are busy. reasoning similarly to the previous proof we observe that $Y \le D_k/4$ and $X \ge 3D_k/4$; Therefore, the total work done by DM during $\hat{I}$ is greater or equal to $(3m + 1)D_k/4$.

Now recall (2). Since the total amount of work done in $\hat{I}$ by DM is bounded by the total work due in $\hat{I}$ or released in $\hat{I}$, we have

$$\begin{aligned}
&\sum_{i:T_i \le D_k} \left(\left\lceil \frac{D_k}{T_i} \right\rceil + 1 \right) \mathrm{vol}(G_i) + \sum_{i:T_i > D_k} \mathrm{vol}(G_i) \\
&\le 3D_k \left(\sum_{i:T_i \le D_k} \mathrm{vol}(G_i)/T_i + \sum_{i:T_i > D_k} \mathrm{vol}(G_i)/3D_k \right) \\
&\le \frac{3m+1}{4} D_k,
\end{aligned}$$

where we have used the fact that $\lceil x \rceil \leq 2x$ when $x \geq 1$. This contradicts the assumption that DM fails and completes the proof of the theorem. ■

VI. Conclusions and Future Works

In this paper we have closed the gap between feasibility analysis for the sequential sporadic task model and that of its parallel generalization, in which each sporadic task is modeled as a DAG. We have shown that, even for DAG tasks, global EDF has a tight speedup bound of $2 - 1/m$, where m is the number of processors, while DM has a speedup bound of at most $3 - 1/m$. We have also presented polynomial and pseudopolynomial time tests for determining whether a set of sporadic DAG tasks can be scheduled by EDF or DM to meet all deadlines on a specified number of processors. It is remarkable that the speedup bound of the pseudopolynomial time test matches that of the best EDF-schedulability test known for ordinary (sequential) sporadic task sets, see [2], [4]. This suggests that better speedup bounds can only be achieved by algorithms with a higher degree of sophistication than global EDF. Another interesting direction for future work is to provide speedup bounds for sufficient schedulability tests based on simpler conditions, such as the polynomial time schedulability tests that we proposed in Section V.

Acknowledgment

Research of the second author has been partially supported by the INRIA International Partnership AMICI.

References

[1] B. Andersson and D. de Niz. Analyzing Global-EDF for multiprocessor scheduling of parallel tasks. In *Proceedings of the 16th Int. Conf. on Principles of Distributed Systems*, pages 16–30. Springer, 2012.

[2] S. K. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller. Improved multiprocessor global schedulability analysis. *Real-Time Systems*, 46(1):3–24, 2010.

[3] S. K. Baruah, V. Bonifaci, A. Marchetti-Spaccamela, L. Stougie, and A. Wiese. A generalized parallel task model for recurrent real-time processes. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 63–72. IEEE, Los Alamitos, CA, 2012.

[4] V. Bonifaci, A. Marchetti-Spaccamela, and S. Stiller. A constant-approximate feasibility test for multiprocessor real-time scheduling. *Algorithmica*, 62(3–4):1034–1049, 2012.

[5] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proceedings of the Int. Federation for Information Processing Congress*, pages 807–813. North-Holland, Amsterdam, 1974.

[6] K. Lakshmanan, S. Kato, and R. Rajkumar. Scheduling parallel real-time tasks on multi-core processors. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 259–268. IEEE, Los Alamitos, CA, 2010.

[7] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Eval.*, 2(4):237–250, 1982.

[8] C. L. Liu. Scheduling algorithms for hard real-time programming of a single processor. *JPL Space Programs Summary*, 37–60(II):31–37, 1969.

[9] C. L. Liu. Scheduling algorithms for multiprocessors in a hard real-time environment. *JPL Space Programs Summary*, 37–60(II):28–31, 1969.

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

[11] G. Nelissen, V. Berten, J. Goossens, and D. Milojevic. Techniques optimizing the number of processors to schedule multi-threaded tasks. In *Proceedings of the Euromicro Conf. on Real-Time Systems*, pages 321–330, 2012.

[12] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.

[13] A. Saifullah, K. Agrawal, C. Lu, and C. D. Gill. Multi-core real-time scheduling for generalized parallel task models. In *Proceedings of the IEEE Real-Time Systems Symposium*, pages 217–226. IEEE, Los Alamitos, CA, 2011.

[14] M. Stigge, P. Ekberg, N. Guan, and W. Yi. The digraph real-time task model. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 71–80. IEEE, Los Alamitos, CA, 2011.

[15] M. Stigge, P. Ekberg, N. Guan, and W. Yi. On the tractability of digraph-based task models. In *Proceedings of the Euromicro Conference on Real-Time Systems*, pages 162–171. IEEE, Los Alamitos, CA, 2011.

[16] J. D. Ullman. NP-complete scheduling problems. *Journal of Computer and Systems Sciences*, 10(3):384–393, 1975.