Exact Response-Time Bounds of Periodic DAG Tasks under Server-Based Global Scheduling

Shareef Ahmed and James H. Anderson
Department of Computer Science, University of North Carolina at Chapel Hill
{shareef, anderson}@cs.unc.edu

***Abstract*—Artificial-intelligence (AI) techniques are revolutionizing modern safety-critical real-time systems by enabling autonomous features never seen before. However, AI-based workloads are typically expressed as processing graphs that are subject to complex tradeoffs involving parallelism and dataflow dependencies. Due to such complexities, exact analysis of graph-based tasks is challenging under most (if not all) schedulers. This paper presents a periodic server-based scheduling policy for periodic graph-based task systems and provides an exact response-time analysis under this policy. This analysis entails pseudo-polynomial time complexity for pseudo-harmonic periodic graph-based tasks, which are commonly used in practice.**

I. Introduction

Recent advances in artificial-intelligence (AI) techniques have fueled increased demand for autonomous features in safety-critical real-time systems. AI-based workloads that realize such features often have complicated dataflow dependencies that are modeled as processing graphs, where nodes represent sequential computation and edges represent data dependencies. The temporal correctness of a graph-based task system is often validated via response-time analysis; recent work on this topic includes [4], [20], [26], [28], [30], [34].

Unfortunately, most existing work on response-time bounds for graph-based tasks on multiprocessors does not provide exact response-time bounds. In fact, existing response-time analysis for graph-based tasks incurs pessimism by over-approximating intra- and inter-task interference. The lack of exact response-time bounds hinders the quantification of such pessimism inherent in efficiently computable existing response-time bounds. However, determining exact response-time bounds for graph-based tasks on multiprocessors is a challenging problem that requires accounting for **(i)** intra-task interference, **(ii)** inter-task interference, **(iii)** inter-instance dependencies, and **(iv)** instance-level parallelism.

In this paper, we show that exact response-time bounds can be obtained for directed-acyclic-graph-(DAG)-based periodic task systems under a server-based global scheduler via a simulation-based strategy that accounts for all of (i)–(iv). According to a recent survey, more than 80% of industrial real-time systems have periodic activities [3]. Our work was inspired by prior seminal work on simulation-based strategies for globally scheduled independent, periodic (non-DAG) tasks [1], [9], [12], [21]. The method we present enables the computation of exact response-time bounds in pseudo-polynomial time for pseudo-harmonic DAG-based tasks where every period divides the maximum period. Pseudo-harmonic tasks (which generalize harmonic tasks) are common in practice [16].

Work was supported by NSF grants CPS 1837337, CPS 2038855, and CPS 2038960, ARO grant W911NF-20-1-0237, and ONR grant N00014-20-1-2698.

Why server-based scheduling? A simulation-based approach involves upper bounding both the time to reach schedule repetition and the period of schedule repetition. This is difficult for DAG tasks under schedulers without servers. Such difficulties are mainly due to variations in node activation times due to precedence constraints. Per-node reservation servers provide an upper bound on the processor capacity allocated to a node over a hyperperiod, a property that we exploit to ensure schedule repetition of DAG tasks. Moreover, server-based scheduling is useful for other purposes, *e.g.*, to deal with inaccurate *worst-case execution time* (WCET) estimates.

It's not so easy. Our considered task model is subject to all of (i)–(iv), so per-node reservation servers have both inter-instance dependencies and instance-level parallelism. While schedule repetition of independent non-DAG tasks is well-studied, to our knowledge, no work has considered task models with instance-level parallelism. Such parallelism complicates the schedule analysis as a node can have multiple ready instances at the same time. Translating the schedule repetition from servers to DAGs poses added challenges due to unused server budgets, *e.g.*, a schedule at the DAG level may not start repeating when the sever-level schedule starts to repeat.

Related work. Most prior work on DAG-based task systems has focused on task models that obviate inter-instance interference (*e.g.*, hard real-time systems with constrained deadlines) [13], [19], [23]–[25], [34], inter-instance dependencies (*e.g.*, systems with unrestricted parallelism) [11], [14], [17], [22], [30], [32], or inter-task interference (*e.g.*, systems with a single DAG) [5], [6]. Work that has considered all of the above mostly precludes instance-level parallelism by forcing dependencies between consecutive DAG instances [18], [31], [33]. Recent work by Amert *et al.* [4] provides non-exact response-time bounds for task models allowing both inter-instance parallelism and dependencies. Work on tighter response-time bounds of DAG tasks has focused on systems without inter-instance and inter-task interference [7], [27], [29].

Contributions. Our contributions are threefold. First, we give a server-based scheduling policy for periodic DAG tasks that are subject to all of (i)–(iv) mentioned earlier. Second, utilizing the repetition of server schedules, we give a simulation-based approach to derive exact response-time bounds for periodic DAG tasks under our proposed scheduling policy. To our knowledge, this is the first work that considers exact response-time bounds of DAG tasks under any scheduler in the presence of (i)–(iv). We also give slack-reallocation methods to reclaim unused budget (if possible) without violating our response-time bounds. Third, we provide results from simulation experiments that show the benefit of our approach.

Organization. After covering needed background (Sec. II), we describe our server-based approach (Sec. III), explain how to obtain exact DAG response-time bounds (Sec. IV), discuss our experiments (Sec. VI), and conclude (Sec. VII).

II. Preliminaries

We consider a task system Γ consisting of N DAG tasks globally scheduled on m identical processors. Each *DAG task* G^v has n^v nodes that represent tasks $\{\tau_1^v, \tau_2^v, \ldots, \tau_{n^v}^v\}$. A directed edge from τ_i^v to τ_k^v represents a precedence constraint between the *predecessor* task τ_i^v and the *successor* task τ_i^v. The set of predecessors of τ_i^v is denoted by $pred(\tau_i^v)$. Each DAG task G^v has a unique *source* task τ_1^v with no incoming edge and a unique *sink* task $\tau_{n^v}^v$ with no outgoing edge.

Each DAG task G^v has a *period* T^v. G^v releases a *DAG job* every T^v time units. The j^{th} DAG job of G^v is denoted by G_j^v. G^v has an *offset* O^v, which is the time when G_1^v is released. The DAG job G_j^v consists of a *job* $\tau_{i,j}^v$ for each task τ_i^v in that DAG. Each job $\tau_{i,j}^v$ is preemptive. The *release time* and *finish time* of $\tau_{i,j}^v$ are denoted by $r(\tau_{i,j}^v)$ and $f(\tau_{i,j}^v)$, respectively. The source task τ_i^v releases its j^{th} job when G_j^v is released. The j^{th} job of each non-source task is released once the j^{th} job of all of its predecessor tasks finish, *i.e.*, $r(\tau_{i,j}^v) = \max_{\tau_k^v \in pred(\tau_i^v)}\{f(\tau_{k,j}^v)\}$. The *response time* of $\tau_{i,j}^v$ is $f(\tau_{i,j}^v) - r(\tau_{1,j}^v)$. Task τ_i^v's response time is $R(\tau_i^v) = \max_j\{f(\tau_{i,j}^v) - r(\tau_{1,j}^v)\}$. G_j^v completes when $\tau_{n^v,j}^v$ finishes. G_j^v's response time equals $\tau_{n^v,j}^v$'s response time. G^v's response time equals $\tau_{n^v}^v$'s response time, *i.e.*, $R(\tau_{n^v}^v)$.

The WCET of τ_i^v is denoted by $C_i^v > 0$. The *utilization* of τ_i^v is $u_i^v = C_i^v/T^v$. The utilization of G^v is $U^v = \sum_{i=1}^{n^v} u_i^v$. The total utilization of all DAG tasks is $U = \sum_{v=1}^{N} U^v$. We let $T_{max} = \max_v T^v$, $C_{max} = \max_{v,i} C_i^v$, and $O_{max} = \max_v O^v$. The *hyperperiod* H is the least common multiple of all periods. A task system is *pseudo-harmonic* if each period divides T_{max}, *i.e.*, $H = T_{max}$ holds. We do not require Γ to be pseudo-harmonic, *i.e.*, our results apply to non-pseudo-harmonic task systems too. We summarize all introduced notation in Tbl. I.

We assume time to be discrete and a unit of time to be 1.0. All scheduling decisions are taken at integer points in time. We also assume all task parameters to be integers. Therefore, when a job $\tau_{i,j}^v$ executes during an unit interval $[t-1, t)$, it continuously executes during $[t-1, t)$.

Restricted parallelism. We allow a job of each task to execute in parallel with other jobs of that task according to a recently introduced model for specifying intra-task parallelism called

TABLE I: Notation summary.

Symbol	Meaning	Symbol	Meaning
N	No. of DAG tasks	P_i^v	Parallelization level of τ_i^v
m	No. of processors	u_i^v	C_i^v/T^v
Γ	Task system	U^v	$\sum_{i=1}^{n^v} u_i^v$
G^v	v^{th} DAG task	U	$\sum_{v=1}^{N} U^v$
n^v	No. of nodes in G^v	C_i^v	τ_i^v' WCET and S_i^v's budget
τ_i^v	i^{th} task of G^v	C_{max}	$\max_{v,i} C_i^v$
$\tau_{i,j}^v$	j^{th} job of τ_i^v	$d(S_{i,j}^v)$	Deadline of $S_{i,j}^v$
T^v	Period of G^v	H	Hyperperiod
S_i^v	Server of τ_i^v	$R(\cdot)$	Response-time bound
$S_{i,j}^v$	j^{th} job of S_i^v	T_{max}	$\max_v T^v$
Γ_s	Set of servers	$f(\cdot)$	Finish time
O^v	offset of G^v	$r(\cdot)$	Release time
O_{max}	$\max_v O^v$	h^v	H/T^v

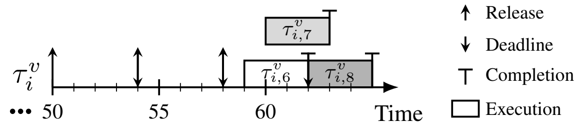


Fig. 1: Example illustrating restricted parallelism. Subsequent jobs are shaded darker.

the *restricted parallelism (rp)* model [4]. Under the rp model, each task τ_i^v has a *parallelization level* P_i^v, which denotes the number of consecutive jobs of τ_i^v that can execute in parallel at any time instant. In particular, job $\tau_{i,j}^v$ with $j > P_i^v$ cannot start execution until $\tau_{i,j-P_i^v}^v$ completes.

Ex. 1. Fig. 1 depicts a task τ_i^v with $P_i^v = 2$. Assume that $m = 3$. Jobs $\tau_{i,6}^v, \tau_{i,7}^v$, and $\tau_{i,8}^v$ are released at times 50, 54, and 58, respectively. At time 59, $\tau_{i,6}^v$ is scheduled. Assume that $\tau_{i,7}^v$ and $\tau_{i,8}^v$ become one of the top-m-priority jobs at time 60. Since $P_i^v = 2$, $\tau_{i,7}^v$ is scheduled at time 60. However, $\tau_{i,8}^v$ is not scheduled until time 62 when $\tau_{i,6}^v$ completes execution. $\Diamond$

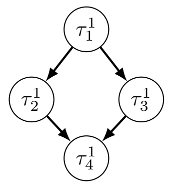
Task τ_i^v has *unrestricted* intra-task parallelism if $P_i^v \geq m$, and *no* intra-task parallelism if $P_i^v = 1$. A task's response time may decrease with increased parallelism [4], but unrestricted parallelism is not always possible, *e.g.*, if job $\tau_{i,j}^v$ requires output from job $\tau_{i,k}^v$ with $k < j$, then $P_i^v \leq j - k$ holds.

Feasibility conditions. Under the rp model, the following condition ensures a bounded response time for each DAG task: $U \leq m \wedge (\forall i, v :: u_i^v \leq P_i^v)$ [4]. We assume that Γ admits this condition.

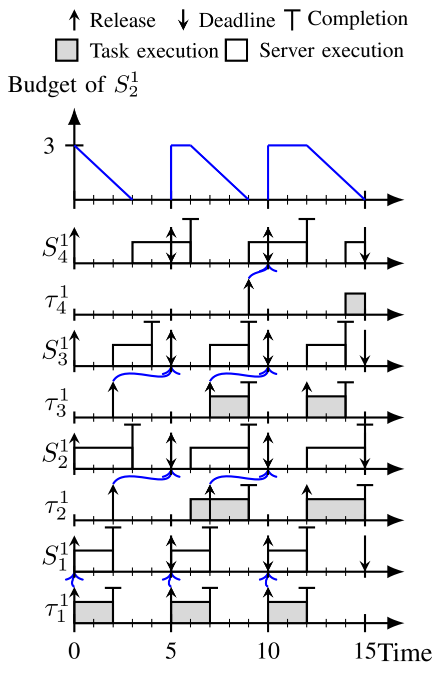
III. Server-based Scheduling of DAGs

To schedule DAG tasks, we adopt a server-based policy where a global scheduler allocates time to per-node reservation servers, upon which task jobs are scheduled.

Reservation servers. For each task τ_i^v, we define a periodic *reservation server* S_i^v. We denote the set of all servers as Γ_s. Each server S_i^v has a period T^v and a budget C_i^v. Note that S_i^v's period and budget are the same as G^v's period and τ_i^v's WCET, respectively. Each server S_i^v releases a (potentially infinite) sequence of *server jobs* $S_{i,1}^v, S_{i,2}^v, \ldots$. The *relative deadline* of S_i^v is denoted by $D(S_i^v)$; we assume implicit deadlines, *i.e.*, $D(S_i^v) = T^v$. The *release time*, (absolute) *deadline*, and *finish time* of $S_{i,j}^v$ are denoted by $r(S_{i,j}^v)$, $d(S_{i,j}^v)$, and $f(S_{i,j}^v)$, respectively. Server S_i^v releases its first job $S_{i,1}^v$ at



(a) DAG G^1.



(b) A schedule of G^1.

Fig. 2: Illustration of server-based scheduling. Blue arrows between job and server job releases represent job linking.

time O^v. Its subsequent jobs are released periodically, *i.e.*, $S^v_{i,j}$ is released at time $O^v + (j-1)T^v$. Therefore, we have

$$\forall v, i, j : r(S^v_{i,j}) = r(\tau^v_{1,j}).$$

We do not require deadlines to be hard, *i.e.*, server jobs can miss their deadlines. Server S^v_i has a parallelization level of P^v_i matching that of τ^v_i.

Server jobs are scheduled according to a global earliest-deadline-first (GEDF) scheduler, but our results apply under a wider class of schedulers called global-EDF-like (GEL) schedulers.[1] Before elaborating on the scheduling of servers, we first give budget consumption and replenishment rules.

Replenishment Rule. The budget of $S^v_{i,j}$ is replenished to C^v_i when it is released.

Consumption Rule. $S^v_{i,j}$ consumes budget at the rate of one execution unit per unit of time when it is scheduled until its budget is exhausted.

Def. 1. *A server job $S^v_{i,j}$ is* complete *after its budget is exhausted. $S^v_{i,j}$ is* pending *at time t if $r(S^v_{i,j}) \le t < f(S^v_{i,j})$. $S^v_{i,j}$ is* ready *if it is pending and $S^v_{i,j-P^v_i}$ (if $j > P^v_i$) is complete.*

Under GEDF scheduling, the (up to) m ready server jobs with earliest deadlines are scheduled. We assume ties are broken arbitrarily but consistently by DAG and task indices, *i.e.*, if $d(S^v_{i,j}) = d(S^u_{k,\ell})$ and $d(S^v_{i,p}) = d(S^u_{k,q})$, then $S^v_{i,j}$ has higher priority than $S^u_{k,\ell}$ if and only if $S^v_{i,p}$ has higher priority than $S^u_{k,q}$. The response time of $S^v_{i,j}$ (resp., S^v_i) is $f(S^v_{i,j}) - r(S^v_{i,j})$ (resp., $\max_j\{f(S^v_{i,j}) - r(S^v_{i,j})\}$).

[1]Under a GEL scheduler, each job has a *priority point* within a constant distance of its release; an earliest-priority-point-first order is assumed.

Ex. 2. Fig. 2(a) depicts a DAG G^1 consisting of four nodes. The period of G^1 is 5.0 time units. Assume that $C^1_1 = 2.0, C^1_2 = 3.0, C^1_3 = 2.0$, and $C^1_4 = 3.0$. The parallelization level of each task is one. There are four servers each corresponding to a task. Each server has a period of 5.0 time units and releases is first server job at time 0 when τ^1_1 releases its first job. The server S^1_2 corresponding to the task τ^1_2 has a budget of 3.0 units. Server job $S^1_{2,1}$'s budget is replenished to 3.0 units when $S^1_{2,1}$ is released at time 0. It consumes its budget by one unit per unit of time when it is scheduled. $S^1_{2,1}$ completes at time 3 when its budget is exhausted. ◊

Scheduling tasks on servers. In describing how jobs are scheduled on servers, we use the following terminology.

Def. 2. *A job $\tau^v_{i,j}$ is* pending *at time t if $r(\tau^v_{i,j}) \le t < f(\tau^v_{i,j})$ holds. $\tau^v_{i,j}$ is* ready *at time t if it is pending at time t and job $\tau^v_{i,j-P^v_i}$ (if $j > P^v_i$) finishes execution by time t.*

Jobs are scheduled on servers via the following rules.

R1 Jobs of τ^v_i are scheduled on server jobs of S^v_i. A job $\tau^v_{i,j}$ is *linked* to a single server job $S^v_{i,\ell}$ (via Rule R2 given below) and at most one job can be linked to a server job.

R2 Assume that a server job $S^v_{i,\ell}$ is released at time t. If $r(\tau^v_{i,1}) \le t$ holds and $\tau^v_{i,1}$ is not linked to any server job at time t, then $\tau^v_{i,1}$ is linked to $S^v_{i,\ell}$. Otherwise, if $\tau^v_{i,j}$ is the last job of τ^v_i that is linked to some server job and $r(\tau^v_{i,j+1}) \le t$ holds, then $\tau^v_{i,j+1}$ is linked to $S^v_{i,\ell}$.

R3 If $\tau^v_{i,j}$ is linked to $S^v_{i,\ell}$, then $\tau^v_{i,j}$ executes whenever $S^v_{i,\ell}$ is scheduled until $\tau^v_{i,j}$ completes.

Ex. 2 (Cont'd). Fig. 2(b) depicts a schedule of G^1. At time 0, the first job of each server is released. Since $\tau^1_{1,1}$ is released at time 0, by Rule R2, it is linked to $S^1_{1,1}$. By Rule R3, $\tau^1_{1,1}$ executes when $S^1_{1,1}$ is scheduled during $[0, 2)$. At time 2, $\tau^1_{2,1}$ and $\tau^1_{3,1}$ are released. At time 5, when $S^1_{2,1}$ (resp., $S^1_{3,1}$) is released, $\tau^1_{2,1}$ (resp., $S^1_{3,1}$) is linked to it. ◊

As seen in Fig. 2(b), an unlinked server job has unused budget. In Sec. V, we give slack-reallocation rules to utilize such unused budgets without violating response-time bounds.

IV. Response-Time Bounds

In this section, we give response-time bounds for DAG tasks under the server-based scheduling given in Sec. III. Our goal is to use these response-time bounds as a basis for deriving exact response-time bounds in Sec. V.

Server response-time bounds. As server tasks are periodic and have restricted parallelism, previously derived response-time bounds apply to them [4].

Def. 3. *Let $\mathcal{U}_b = \sum_{b \text{ largest values of } \tau^v_i \text{ with } P^v_i < m} u^v_i$ and $\mathcal{C}_b = \sum_{b \text{ largest values of } \tau^v_i \text{ with } P^v_i < m} u^v_i$.*

From [4], S^v_i has a response-time bound $R(S^v_i)$, where

$$R(S^v_i) = T^v + \frac{(m-1)C_{max} + 2\mathcal{C}_{m-1}}{m - \mathcal{U}_{m-1}} + C^v_i. \quad (1)$$

Response-time bounds of DAG tasks. Using the response-time bounds of servers given in (1), we now derive response-time bounds of DAG tasks under server-based scheduling.

Lemma 1. *If $S^v_{i,j}$ and $S^v_{i,k}$ are ready at time t where $j < k$ and $S^v_{i,k}$ is scheduled at time t, then $S^v_{i,j}$ is also scheduled at time t.*

Proof. Since $j < k$ and S^v_i releases periodically, $r(S^v_{i,j}) < r(S^v_{i,k})$ holds. Thus, $d(S^v_{i,j}) < d(S^v_{i,k})$ holds. Having higher priority and being ready, $S^v_{i,j}$ is thus scheduled if $S^v_{i,k}$ is. □

Lemma 2. *For any j and k such that $j \le k$, $f(S^v_{i,j}) \le f(S^v_{i,k})$.*

Proof. Assume for a contradiction that the lemma does not hold, in which case $j < k$ clearly holds. Let t be the first time instant such that there are server jobs $S^v_{i,j}$ and $S^v_{i,k}$ such that $j < k$, $t < f(S^v_{i,j})$, and $t = f(S^v_{i,k})$. Since, by the budget Consumption Rule, $S^v_{i,j}$ and $S^v_{i,k}$ are scheduled for C^v_i time units before their completion and $f(S^v_{i,j}) > f(S^v_{i,k})$, there is a time instant $t' \le t$ such that $S^v_{i,j}$ is not scheduled at t', but $S^v_{i,k}$ is scheduled at t'.

We now prove that $S^v_{i,j}$ is ready at time t', which by Lem. 1 implies it is scheduled at time t', a contradiction. Since $S^v_{i,k}$ is scheduled at t' and $j < k$, $r(S^v_{i,j}) < r(S^v_{i,k}) \le t'$ holds. By the definition of t and t', $t' < f(S^v_{i,j})$. If $j < P^v_i$, then $S^v_{i,j}$ is ready at time t' as claimed, so assume $j \ge P^v_i$. Since $S^v_{i,k}$ is scheduled (hence ready) at time t', $S^v_{i,k-P^v_i}$ completes by time t'. As $j < k$, by the definition of t, $f(S^v_{i,j-P^v_i}) \le f(S^v_{i,k-P^v_i})$. Thus, $S^v_{i,j-P^v_i}$ completes by time t' and $S^v_{i,j}$ is ready then. □

Lemma 3. *If a job $\tau^v_{i,j}$ is ready when the server job $S^v_{i,k}$ to which it is linked is first scheduled, then $f(\tau^v_{i,j}) \le f(S^v_{i,k})$.*

Proof. By the budget Consumption Rule, $S^v_{i,k}$ is scheduled for C^v_i time units. Since $\tau^v_{i,j}$ is ready when $S^v_{i,k}$ is first scheduled, by Rule R3, $\tau^v_{i,j}$ completes execution at or before $S^v_{i,k}$'s budget is exhausted. Therefore, $f(\tau^v_{i,j}) \le f(S^v_{i,k})$ holds. □

From Rule R2, we have the following lemmas whose proofs are given in an online appendix [2].

Lemma 4. *If $\tau^v_{i,j}$ is linked to a server job $S^v_{i,k}$, then $j \le k$.*

Lemma 5. *If $\tau^v_{i,j}$ and $\tau^v_{i,j+c}$ are linked to $S^v_{i,k}$ and $S^v_{i,\ell}$, respectively, then $\ell - k \ge c$ holds.*

We now define a response-time bound $R(\tau^v_i)$ for each τ^v_i. $R(\tau^v_i)$ is recursively computed according to τ^v_i's predecessors' response-time bounds. Let

$$R(\tau^v_i) = \Phi^v_i + R(S^v_i) + T^v, \tag{2}$$

$$\text{where} \quad \Phi^v_i = \begin{cases} 0 & i = 1 \\ \max\limits_{\tau^v_j \in pred(\tau^v_i)} \{R(\tau^v_j)\} & \text{otherwise.} \end{cases} \tag{3}$$

In Thm. 1, we show that $R(\tau^v_i)$ is a response-time bound of τ^v_i using Lems. 6–9 given below.

Lemma 6. *For any job $\tau^v_{i,j}$, $\tau^v_{i,j}$ is ready at or before the server job $S^v_{i,k}$ to which it is linked starts execution.*

Proof. Assume otherwise. Let t be the first time instant such that there is a job $\tau^v_{i,j}$ that is not ready, but the server job $S^v_{i,k}$ to which it is linked starts execution at time t. By Rule R2, $r(\tau^v_{i,j}) \le t$. Since $\tau^v_{i,j}$ is not ready at time t, $j > P^v_i$ holds and $\tau^v_{i,j-P^v_i}$ does not complete execution at or before time t. By Lem. 4, $k \ge j > P^v_i$ holds.

We now prove that $\tau^v_{i,j-P^v_i}$ completes by time t, *i.e.*, $f(\tau^v_{i,j-P^v_i}) \le t$, thereby reaching a contradiction. By Lem. 5, $\tau^v_{i,j-P^v_i}$ is linked to $S^v_{i,\ell}$ with $\ell \le k - P^v_i$. Let t' be the first time instant when $S^v_{i,\ell}$ is scheduled. Since $S^v_{i,k}$ is scheduled at time t, $f(S^v_{i,k-P^v_i}) \le t$. Thus, by Lem. 2, $f(S^v_{i,\ell}) \le t$. Since $C^v_i > 0$, we have $t' < f(S^v_{i,\ell}) \le t$. Hence, by the definition of t, $\tau^v_{i,j-P^v_i}$ is ready when $S^v_{i,\ell}$ is first scheduled. By Lem. 3, $f(\tau^v_{i,j-P^v_i}) \le f(S^v_{i,\ell}) \le t$. Thus, $\tau^v_{i,j}$ is ready at time t. □

By Lems. 6 and 3, we have the following lemma.

Lemma 7. *For any job $\tau^v_{i,j}$, $\tau^v_{i,j}$ completes execution at or before the server job $S^v_{i,k}$ to which it is linked completes.*

Using (2) and (3), we have the following lemma.

Lemma 8. *For any non-source task τ^v_i, $\Phi^v_i \ge \Phi^v_k + R(S^v_k) + T^v$ holds, where $\tau^v_k \in pred(\tau^v_i)$.*

Lemma 9. *For any job $\tau^v_{i,j}$, $\tau^v_{i,j}$ is linked to a server job at or before time $r(\tau^v_{1,j}) + \Phi^v_i + T^v$.*

Proof. Assume for a contradiction that t is the first time instant such that a job $\tau^v_{i,j}$ is not linked to any server job and $t = r(\tau^v_{1,j}) + \Phi^v_i + T^v$ holds. Let $S^v_{i,k}$ be the latest server job of S^v_i released at or before time t. We will show that $\tau^v_{i,j}$ is linked to $S^v_{i,k}$, thereby reaching a contradiction. Since S^v_i releases jobs periodically, $r(S^v_{i,k}) \ge t - T^v = r(\tau^v_{1,j}) + \Phi^v_i$. By Rule R2, it suffices to prove that $r(\tau^v_{i,j}) \le r(\tau^v_{1,j}) + \Phi^v_i$ holds and $\tau^v_{i,j-1}$ (if $j > 1$) is linked to a server job by time $r(\tau^v_{1,j}) + \Phi^v_i$.

Claim 9.1. *$\tau^v_{i,j}$ is released at or before $r(\tau^v_{1,j}) + \Phi^v_i$.*

Proof. Assume otherwise. Since $\tau^v_{1,j}$ is released at time $r(\tau^v_{1,j})$ and by (3), $\Phi^v_1 = 0$, we have $i \ne 1$. Thus, τ^v_i is a non-source task. Since τ^v_i releases $\tau^v_{i,j}$ once the j^{th} job of each of its predecessors completes, there is a job $\tau^v_{p,j}$ such that $\tau^v_p \in pred(\tau^v_i)$ and $f(\tau^v_{p,j}) > r(\tau^v_{1,j}) + \Phi^v_i$ hold. By Lem. 8, we have $\Phi^v_p < \Phi^v_i$. Thus, $r(\tau^v_{1,j}) + \Phi^v_p + T^v < r(\tau^v_{1,j}) + \Phi^v_i + T^v = t$. Therefore, by the definition of t, $\tau^v_{p,j}$ is linked to a server job at or before time $r(\tau^v_{1,j}) + \Phi^v_p + T^v$. Assume that $\tau^v_{p,j}$ is linked to $S^v_{p,\ell}$. Then, by Rule R2, $r(S^v_{p,\ell}) \le r(\tau^v_{1,j}) + \Phi^v_p + T^v$. Since the response time of $S^v_{p,\ell}$ is at most $R(S^v_p)$, we have

$$\begin{aligned} f(S^v_{p,\ell}) &\le r(S^v_{p,\ell}) + R(S^v_p) \\ &\le \{\text{Since } r(S^v_{p,\ell}) \le r(\tau^v_{1,j}) + \Phi^v_p + T^v\} \\ &\quad r(\tau^v_{1,j}) + \Phi^v_p + T^v + R(S^v_p) \\ &\le \{\text{By Lem. 8 and since } \tau^v_p \in pred(\tau^v_i)\} \\ &\quad r(\tau^v_{1,j}) + \Phi^v_i. \end{aligned} \tag{4}$$

By Lem. 7 and (4), we have $f(\tau^v_{p,j}) \le f(S^v_{p,\ell}) \le r(\tau^v_{1,j}) + \Phi^v_i$, a contradiction. □

Claim 9.2. *If $j > 1$, then $\tau^v_{i,j-1}$ is linked to a server job at or before time $r(\tau^v_{1,j}) + \Phi^v_i$.*

Proof. Since source task τ^v_1 releases jobs periodically, we have $r(\tau^v_{1,j}) = r(\tau^v_{1,j-1}) + T^v$. Thus, $r(\tau^v_{1,j-1}) + \Phi^v_i + T^v = r(\tau^v_{1,j}) + \Phi^v_i < r(\tau^v_{1,j}) + \Phi^v_i + T^v$ holds. Therefore, by the definition of t, $\tau^v_{i,j-1}$ is linked to a server job at or before time $r(\tau^v_{1,j-1}) + \Phi^v_i + T^v = r(\tau^v_{1,j}) + \Phi^v_i$. □

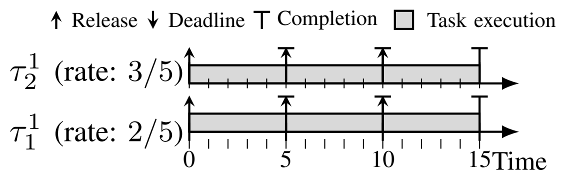


Fig. 3: An ideal schedule of τ_1^v and τ_2^v of G^1 in Fig. 2(a). Server jobs are not shown as they have the same schedule.

These two claims yield a contradiction, as noted above. □

By (2) and Lems. 9 and 7, we have the following theorem.

Theorem 1. *The response time of any job $\tau_{i,j}^v$ is at most $R(\tau_i^v)$.*

V. Exact Response-Time Bounds

In this section, we give a simulation-based method to compute exact response-time bounds of DAG tasks under the server-based scheduling policy given in Sec. III. We initially assume the following, which we relax later.

A Each job of any task τ_i^v executes for its WCET C_i^v.

Note that the response-time bounds (and associated lemmas and theorems) given in Sec. IV do not rely on Assumption A.

A. Definitions and Notation

We denote a GEDF schedule of Γ_s as $\mathcal{G}$. We denote a schedule of Γ on the server schedule $\mathcal{G}$ as $\mathcal{S}$.

Ideal schedule. Let $\{\widehat{\pi}_1^1, \widehat{\pi}_2^1, \ldots, \widehat{\pi}_{n^N}^N\}$ be $\sum_{v=1}^N n^v$ processors, where $\widehat{\pi}_i^v$ has speed of u_i^v. In an *ideal schedule $\mathcal{I}$*, each task τ_i^v and corresponding server S_i^v is partitioned to be scheduled on processor $\widehat{\pi}_i^v$. Each server's budget is replenished according to the budget Replenishment Rule given in Sec. IV. However, its budget is consumed via the following rule.

Ideal Consumption Rule. $S_{i,j}^v$ consumes budget at the rate of u_i^v execution unit per unit of time when it is scheduled until its budget is exhausted.

Each server job $S_{i,j}^v$ is scheduled at time $r(S_{i,j}^v) = r(\tau_{1,j}^v)$ and remains scheduled until its budget is exhausted. Therefore, $S_{i,j}^v$ completes at time $r(S_{i,j}^v) + C_i^v/u_i^v = r(S_{i,j}^v) + T^v$.

Each job $\tau_{i,j}^v$ executes at a rate of u_i^v whenever $S_{i,j}^v$ is scheduled. Thus, $\tau_{i,j}^v$ begins execution at time $r(\tau_{1,j}^v)$ and completes execution at time $r(\tau_{1,j}^v) + T^v$. Therefore, all jobs corresponding to a DAG job G_j^v complete execution at time $r(\tau_{1,j}^v) + T^v$ when G_{j+1}^v is released. Note that precedence constraints among tasks are not maintained in $\mathcal{I}$.

Ex. 2 (Cont'd). Fig. 3 depicts an ideal schedule $\mathcal{I}$ corresponding to the tasks τ_1^1 and τ_2^1 of DAG task G^1 in Fig. 2. Although job $\tau_{1,1}^1$ is not complete at time 0, $\tau_{2,1}^1$ starts execution at time 0 at the rate of 3/5 execution units per unit of time. ◊

We now define the term *allocation*. To avoid repetition, we use the notation $J_{i,j}^v$ to denote $\tau_{i,j}^v$ or $S_{i,j}^v$. Similarly, we use J_i^v (resp., Ψ) to denote τ_i^v or S_i^v (resp., Γ or Γ_s). Finally, we use $\mathcal{H}$ to denote a schedule which can be either $\mathcal{S}$ or $\mathcal{G}$ or $\mathcal{I}$.

Allocation. The *cumulative processor capacity* allocated to $J_{i,j}^v$, J_i^v, and Ψ in a schedule $\mathcal{H}$ over an interval $[t, t')$ is denoted by $\mathsf{A}(J_{i,j}^v, t, t', \mathcal{H})$, $\mathsf{A}(J_i^v, t, t', \mathcal{H})$, and $\mathsf{A}(\Psi, t, t', \mathcal{H})$, respectively. Therefore, we have the following equations.

$$\mathsf{A}(J_i^v, t, t', \mathcal{H}) = \sum_j \mathsf{A}(J_{i,j}^v, t, t', \mathcal{H}) \tag{5}$$

$$\mathsf{A}(\Psi, t, t', \mathcal{H}) = \sum_{v,i} \mathsf{A}(J_i^v, t, t', \mathcal{H}) \tag{6}$$

Since the processor capacity allocated to a job or server job over an interval is non-negative, for any intervals $[t, t')$ and $[t, t'')$ with $t'' \geq t' \geq t$, we have the following properties.

$$\mathsf{A}(J_{i,j}^v, t, t', \mathcal{H}) \leq \mathsf{A}(J_{i,j}^v, t, t'', \mathcal{H}) \tag{7}$$

$$\mathsf{A}(J_{i,j}^v, t, t', \mathcal{H}) = \mathsf{A}(J_{i,j}^v, 0, t', \mathcal{H}) - \mathsf{A}(J_{i,j}^v, 0, t, \mathcal{H}) \tag{8}$$

In $\mathcal{I}$, server job $S_{i,j}^v$ and job $\tau_{i,j}^v$ are scheduled when G_j^v is released, *i.e.*, at time $r(\tau_{1,j}^v)$. $S_{i,j}^v$ (resp., $\tau_{i,j}^v$) consumes budget (resp., executes) at a rate of u_i^v until its completion. Therefore, for any interval $[r(\tau_{1,j}^v), t)$, we have

$$\mathsf{A}(J_{i,j}^v, r(\tau_{1,j}^v), t, \mathcal{I}) = \min\{u_i^v(t - r(\tau_{1,j}^v)), C_i^v\}. \tag{9}$$

Similarly, for interval $[t, t')$ with $t \geq O^v$ (resp., $t \geq O_{max}$).

$$\mathsf{A}(J_i^v, t, t', \mathcal{I}) = u_i^v(t' - t) \tag{10}$$

$$\mathsf{A}(\Psi, t, t', \mathcal{I}) = U(t' - t) \tag{11}$$

lag and LAG. The lag of job $J_{i,j}^v$ in schedule $\mathcal{H}$ is defined as

$$\mathsf{lag}(J_{i,j}^v, t, \mathcal{H}) = \mathsf{A}(J_{i,j}^v, 0, t, \mathcal{I}) - \mathsf{A}(J_{i,j}^v, 0, t, \mathcal{H}). \tag{12}$$

The lag (resp., LAG) of J_i^v (resp., Ψ) at time t in $\mathcal{H}$ is given by

$$\begin{aligned}\mathsf{lag}(J_i^v, t, \mathcal{H}) &= \sum_j \mathsf{lag}(J_{i,j}^v, t, \mathcal{H}) \\ &= \mathsf{A}(J_i^v, 0, t, \mathcal{I}) - \mathsf{A}(J_i^v, 0, t, \mathcal{H}) \end{aligned}\tag{13}$$

$$\begin{aligned}\mathsf{LAG}(\Psi, t, \mathcal{H}) &= \sum_{v,i} \mathsf{lag}(J_i^v, t, \mathcal{H}) \\ &= \mathsf{A}(\Psi, 0, t, \mathcal{I}) - \mathsf{A}(\Psi, 0, t, \mathcal{H}) \end{aligned}\tag{14}$$

Since $\mathsf{lag}(J_i^v, 0, \mathcal{H}) = 0$ and $\mathsf{LAG}(\Psi, 0, \mathcal{H}) = 0$, for $t' \geq t$ we have the following equations.

$$\begin{aligned}\mathsf{lag}(J_i^v, t', \mathcal{H}) = \mathsf{lag}(J_i^v, t, \mathcal{H}) + \mathsf{A}(J_i^v, t, t', \mathcal{I}) \\ - \mathsf{A}(J_i^v, t, t', \mathcal{H}) \end{aligned}\tag{15}$$

$$\mathsf{LAG}(\Psi, t', \mathcal{H}) = \mathsf{LAG}(\Psi, t, \mathcal{H}) + \mathsf{A}(\Psi, t, t', \mathcal{I}) - \mathsf{A}(\Psi, t, t', \mathcal{H}) \tag{16}$$

Def. 4. *Let $h^v = H/T^v$.*

Proof overview. We aim to derive an upper bound on the length of the prefix of a schedule of DAG tasks after which the response times of DAG tasks do not increase (Thm. 2). We do so by showing that if the LAG of Γ remains the same at hyperperiod boundaries for a sufficiently long interval of time, then it continues to remain the same at hyperperiod boundaries at any time in the future (Lem. 40). Furthermore, when this happens, response times do not increase afterwards (Lem. 41). The key steps in our proof are as follows.

Step 1. The amount of time a server job $S_{i,j+h^v}^v$ is scheduled by time $t + H$ is at most the amount of time $S_{i,j}^v$ is scheduled by time t (Lem. 18). The amount of time S_i^v

(resp., Γ_s) is scheduled over an an interval $[t, t+H)$ is at most Hu_i^v (resp., HU) (Lems. 23 and 24).

Step 2. If the time allocated to Γ_s over an interval $[t, t+H)$ equals HU, then scheduling decisions and the state of Γ_s are identical at times t and $t+H$ (Lem. 31).

Step 3. If the time allocated to Γ over H-sized intervals remains HU for sufficiently long, then Γ continues to be scheduled for HU time units in any future H-sized interval (Lem. 40) and each DAG task's maximum response time has stabilized (Lem. 41).

Step 4. There is a time instant when the condition mentioned in Step 3 holds (Lem. 43).

We cover Steps 1 and 2 in Sec. V-B and Steps 3 and 4 in Sec. V-C. Due to space constraint, we omit some proofs, which are provided in an online appendix [2].

B. Analysis of Servers

We begin by addressing Steps 1 and 2. To complete Step 1, we will first show, in Lem. 18, that the amount of time allocated to a server job $S^v_{i,j+h^v}$ by time $t+H$ is at most the amount of time allocated to $S^v_{i,j}$ by time t. We will prove the existence of higher-priority jobs that cause $S^v_{i,j+h^v}$ to maintain this property under GEDF. We begin by proving Lems. 10–16, which establish the existence of such higher-priority jobs. The lemma below holds as servers release jobs periodically.

Lemma 10. *For any integer c, $r(S^v_{i,j+ch^v}) = r(S^v_{i,j}) + cH$ and $d(S^v_{i,j+ch^v}) = d(S^v_{i,j}) + cH$ hold.*

The following two lemmas show that the pending jobs of a task are consecutive at any time instant. Informally, this is because, by Lem. 2, a server job cannot finish before a prior server job of the same server finishes.

Lemma 11. *If $S^v_{i,j}$ and $S^v_{i,\ell}$ with $j \le \ell$ are pending at time t, then each server job $S^v_{i,k}$ with $j \le k \le \ell$ is pending at time t.*

Proof. By Def. 1, we have $r(S^v_{i,\ell}) \le t$. Since $k \le \ell$, we have $r(S^v_{i,k}) \le r(S^v_{i,\ell}) \le t$. By Def. 1, we have $f(S^v_{i,j}) > t$. Since $k \ge j$, by Lem. 2, we have $f(S^v_{i,k}) \ge f(S^v_{i,j}) > t$. Thus, by Def. 1, $S^v_{i,k}$ is pending at time t. □

Using Lem. 11, we have the following lemma.

Lemma 12. *If at least p server jobs of S^v_i are pending at time t and $S^v_{i,j}$ is the highest-priority server job of S^v_i that is pending at time t, then $S^v_{i,j+1}, S^v_{i,j+2}, \ldots, S^v_{i,j+p-1}$ are pending at time t.*

Similar to Lem. 12, we have the following lemma.

Lemma 13. *If at least p server jobs of S^v_i are ready at time t and $S^v_{i,j}$ is the highest-priority server job of S^v_i that is pending at time t, then $S^v_{i,j}, S^v_{i,j+1}, \ldots, S^v_{i,j+p-1}$ are ready at time t.*

Proof. Since $P^v_i \ge 1$ and no server job of S^v_i with higher priority than $S^v_{i,j}$ is pending, $S^v_{i,j}$ is ready at time t. By Lem. 12, server jobs $S^v_{i,j+1}$, $S^v_{i,j+2}, \ldots, S^v_{i,j+p-1}$ are pending at time t. Assume that $S^v_{i,j+k}$ such that $1 \le k \le p-1$ is not ready at time t. Then, $j+k > P^v_i$, and $S^v_{i,j+k-P^v_i}$ does not finish execution by time t, *i.e.*, $f(S^v_{i,j+k-P^v_i}) > t$. As there are at least p ready jobs at time t, there is a ready job $S^v_{i,j+\ell}$ at time t such that $\ell > k$ (thus, $j+\ell > P^v_i$). By Lem. 2, $f(S^v_{i,j+\ell-P^v_i}) \ge f(S^v_{i,j+k-P^v_i}) > t$. Thus, $S^v_{i,j+\ell}$ is not ready at time t, a contradiction. □

Lemma 14. *If at least p server jobs of S^v_i are pending at time t, then at least $\min\{p, P^v_i\}$ server jobs are ready at time t.*

Lemma 15. *For any integer c and job index j such that $j + ch^v \ge 1$, if $S^v_{i,j}$ is pending at time t and $\mathsf{A}(S^v_{i,j}, 0, t, \mathcal{G}) \ge \mathsf{A}(S^v_{i,j+ch^v}, 0, t+cH, \mathcal{G})$ holds, then $S^v_{i,j+ch^v}$ is pending at time $t+cH$.*

Def. 5. *Let $hp(S^u_{k,\ell}, S^v_i, t)$ denote the number of ready jobs of S^v_i at time t that have higher priority than $S^u_{k,\ell}$.*

Using Lems. 10–15, the following lemma shows that the number of ready jobs at time $t+H$ with higher priorities than $S^u_{k,\ell+h^u}$ is no smaller than the number of ready jobs at time t with higher priorities than $S^u_{k,\ell}$, when $\mathsf{A}(S^v_{i,j}, 0, t, \mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v}, 0, t+H, \mathcal{G})$ holds for each server job. Informally, this is because, for any ready server job at time t, there exists a unique ready server job at time $t+H$.

Lemma 16. *Assume that, for each server job $S^v_{i,j}$ of a server S^v_i, $\mathsf{A}(S^v_{i,j}, 0, t, \mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v}, 0, t+H, \mathcal{G})$ holds at time $t \ge O^v$. Then, for any server job $S^u_{k,\ell}$, $hp(S^u_{k,\ell}, S^v_i, t) \le hp(S^u_{k,\ell+h^u}, S^v_i, t+H)$ holds.*

Proof. Let $p = hp(S^u_{k,\ell}, S^v_i, t)$. If $p = 0$, then the lemma trivially holds, so assume $p \ge 1$. By Defs. 1 and 5, $p \le P^v_i$. Let $S^v_{i,j}$ be the highest-priority server job of S^v_i that is pending at time t. By Lem. 13, $S^v_{i,j}, S^v_{i,j+1}, \ldots, S^v_{i,j+p-1}$ are ready at time t. Therefore, by Lem. 15 (replacing c by 1), server jobs $S^v_{i,j+h^v}, S^v_{i,j+1+h^v}, \ldots, S^v_{i,j+p-1+h^v}$ are pending at time $t+H$. Let $S^v_{i,x}$ be the highest-priority server job of S^v_i that is pending at time $t+H$. Then, $x \le j+h^v$ holds. By Lem. 11, each server job $S^v_{i,y}$ such that $x \le y \le j+p-1+h^v$ is pending at time $t+H$. Therefore, there are at least $j+p-1+h^v-x+1 = j+p+h^v-x$ pending server jobs of S^v_i at time $t+H$. By Lem. 14, at least $\min\{j+p+h^v-x, P^v_i\}$ server jobs of S^v_i are ready at time $t+H$. Since $x \le j+h^v$, we have $j+p+h^v-x \ge p$. Since by Defs. 1 and 5, $p \le P^v_i$ holds, we have $p \le \min\{j+p+h^v-x, P^v_i\}$. Thus, there are at least p ready jobs of S^v_i at time $t+H$. By Lem. 13, $S^v_{i,x}, S^v_{i,x+1}, \ldots, S^v_{i,x+p-1}$ are ready at time t.

We now prove that each server job $S^v_{i,x+b}$ with $0 \le b \le p-1$ has higher priority than $S^u_{k,\ell+h^v}$. Since $x \le j+h^v$, we have $x+p-1 \le j+p-1+h^v$. Thus, each $S^v_{i,x+b}$ with $0 \le b \le p-1$ has higher or equal priority than $S^v_{i,j+p-1+h^v}$. So, it suffices to prove that $S^v_{i,j+p-1+h^v}$ has higher priority than $S^u_{k,\ell+h^v}$. As $S^v_{i,j}$ is the highest-priority ready job of S^v_i at t, by Def. 5, each of $S^v_{i,j}, S^v_{i,j+1}, \ldots, S^v_{i,j+p-1}$ has higher priority than $S^u_{k,\ell}$. Thus, $d(S^v_{i,j+p-1}) \le d(S^u_{k,\ell})$ holds, and by Lem. 10,

$$\begin{aligned} d(S^u_{k,\ell+h^v}) &= d(S^u_{k,\ell}) + H \\ &\ge \{\text{Since } d(S^v_{i,j+p-1}) \le d(S^u_{k,\ell})\} \\ &\quad d(S^v_{i,j+p-1}) + H \\ &= \{\text{By Lem. } 10\} \\ &\quad d(S^v_{i,j+p-1+h^v}). \end{aligned} \tag{17}$$

Since ties are broken consistently, $S^v_{i,j+p-1+h^v}$ has higher priority than $S^u_{k,\ell+h^v}$ if $S^v_{i,j+p-1}$ has higher priority than $S^u_{k,\ell}$. Thus, there are at least p ready jobs of S^v_i at time $t+H$ that have higher priorities than $S^u_{k,\ell+h^v}$. ☐

The following lemma gives necessary conditions for $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G})$ to not hold for the first time in $\mathcal{G}$.

Lemma 17. *Let $t \ge O^v$ be the first time instant (if any) such that for a job $S^v_{i,j}$, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) < \mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G})$ holds. Then, the following hold.*

(a) $t > O^v$.

(b) *$S^v_{i,j}$ is not scheduled during $[t-1,t)$, but $S^v_{i,j+h^v}$ is scheduled during $[t+H-1,t+H)$.*

Lem. 17(a) holds as, by Def. 4 and Lem. 10, $S^v_{i,j+h^v}$ is released after time O^v+H. By the definition of t, $\mathsf{A}(S^v_{i,j},0,t-1,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v},0,t+H-1,\mathcal{G})$ holds. Thus, Lem. 17(b) must hold to satisfy the lemma assumptions. Using Lems. 16 and 17, we now prove the following lemma.

Lemma 18. *For any server job $S^v_{i,j}$ and time instant $t \ge O^v$, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G})$ holds.*

Proof. Assume otherwise. Let t be the first time instant such that $t \ge O^v$ and there is a job $S^v_{i,j}$ satisfying the following.

$$\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) < \mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G}) \qquad (18)$$

By Lem. 17(a), $t > O^v$ holds. Thus, by the definition of t,

$$\forall u,k,\ell : t-1 \ge O^u :: \mathsf{A}(S^u_{k,\ell},0,t-1,\mathcal{G}) \ge \mathsf{A}(S^u_{k,\ell+h^u},0,t+H-1,\mathcal{G}). \qquad (19)$$

Since $S^v_{i,j+h^v}$ is scheduled for C^v_i time units in total, we have

$$\mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G}) \le C^v_i. \qquad (20)$$

By Lem. 17(b), $S^v_{i,j+h^v}$ is scheduled during $[t+H-1,t+H)$, so $r(S^v_{i,j+h^v}) \le t+H-1$ holds, and by Lem. 10,

$$r(S^v_{i,j}) = r(S^v_{i,j+h^v}) - H \le t-1. \qquad (21)$$

We now prove two claims.

Claim 18.1. *$S^v_{i,j}$ is pending at time $t-1$.*

Proof. By (21), $S^v_{i,j}$ is released at or before time $t-1$. Thus, it suffices to prove that $f(S^v_{i,j}) > t-1$. Assume to the contrary that $f(S^v_{i,j}) \le t-1$. Thus, $\mathsf{A}(S^v_{i,j},0,t-1,\mathcal{G}) = C^v_i$. By (7), we have $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge C^v_i$. By (18), $\mathsf{A}(S^v_{i,j+h^v},0,t+H,\mathcal{G}) > \mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge C^v_i$, contradicting (20). ☐

Claim 18.2. *$S^v_{i,j}$ is ready at time $t-1$.*

Proof. By Clm 18.1, $S^v_{i,j}$ is pending at time $t-1$. If $j \le P^v_i$, then $S^v_{i,j}$ is also ready at time $t-1$, so assume $j > P^v_i$. Since $S^v_{i,j+h^v}$ is scheduled (hence, ready) at time $t+H-1$, $S^v_{i,j+h^v-P^v_i}$ completes by time $t+H-1$. Hence, $\mathsf{A}(S^v_{i,j+h^v-P^v_i},0,t+H-1,\mathcal{G}) = C^v_i$. By (19), we have $\mathsf{A}(S^v_{i,j-P^v_i},0,t-1,\mathcal{G}) \ge C^v_i$. Thus, $S^v_{i,j-P^v_i}$ completes by time $t-1$ and $S^v_{i,j}$ is ready at time $t-1$. ☐

By Clm 18.2 and Lem. 17(b), $S^v_{i,j}$ is ready but not scheduled at time $t-1$. Therefore, at time $t-1$, there are at least m ready server jobs that have higher priorities than $S^v_{i,j}$. By (19) and Lem. 16, each server S^u_k with p ready server jobs of higher priority than $S^v_{i,j}$ at time $t-1$ has at least p ready server jobs of higher priority than $S^v_{i,j+h^v}$ at time $t+H-1$. Thus, there are at least m ready server jobs of higher priority than $S^v_{i,j+h^v}$ at time $t+H-1$. Therefore, $S^v_{i,j+h^v}$ cannot be scheduled at time $t+H-1$, which contradicts Lem. 17(b). ☐

The following lemma generalizes Lem. 18 for the case of multiple hyperperiods.

Lemma 19. *For any server job $S^v_{i,j}$, positive integer c, and time instant $t \ge O^v$, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+ch^v},0,t+cH,\mathcal{G})$.*

Proof. By Lem. 18, for any $k \ge 0$, we have $\mathsf{A}(S^v_{i,j},0,t+kH,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+h^v},0,t+(k+1)H,\mathcal{G})$. Therefore, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \ge \mathsf{A}(S^v_{i,j+ch^v},0,t+cH,\mathcal{G})$. ☐

We now give Lems. 20–24 that complete Step 1. Since server job $S^v_{i,j}$ consumes budget at the rate of u^v_i in $\mathcal{I}$ during $[r(S^v_{i,j}), r(S^v_{i,j})+T^v)$, we have the following lemma.

Lemma 20. *For any job $S^v_{i,j}$, positive integer c, and time instant $t \ge O^v$, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{I}) = \mathsf{A}(S^v_{i,j+ch^v},0,t+cH,\mathcal{I})$.*

Lemma 21. *For any server job $S^v_{i,j}$, positive integer c, and time instant $t \ge O^v$, $\mathsf{lag}(S^v_{i,j},t,\mathcal{G}) \le \mathsf{lag}(S^v_{i,j+ch^v},t+cH,\mathcal{G})$.*

Proof. By Lems. 20 and 19, $\mathsf{A}(S^v_{i,j},0,t,\mathcal{I}) - \mathsf{A}(S^v_{i,j},0,t,\mathcal{G}) \le \mathsf{A}(S^v_{i,j+ch^v},0,t+cH,\mathcal{I}) - \mathsf{A}(S^v_{i,j+ch^v},0,t+cH,\mathcal{G})$. Thus, by (12), the lemma holds. ☐

Since each server job $S^v_{i,j}$ completes at time $r(S^v_{i,j})+T^v = r(S^v_{i,j+1})$ in $\mathcal{I}$, and S^v_{i,h^v+1} is released at time O^v+H, each server job $S^v_{i,j}$ with $j \le h^v$ is complete by time O^v+H in $\mathcal{I}$. Using this fact, we have the following lemma.

Lemma 22. *For any integer $c \ge 1$, server job $S^v_{i,j}$ with $1 \le j \le ch^v$, and time $t \ge O^v + cH$, $\mathsf{lag}(S^v_{i,j},t,\mathcal{G}) \ge 0$ holds.*

Lemma 23. *For any server S^v_i, positive integer c, and time instant $t \ge O^v$, the following hold.*

(a) $\mathsf{lag}(S^v_i,t,\mathcal{G}) \le \mathsf{lag}(S^v_i,t+cH,\mathcal{G})$.

(b) $\mathsf{A}(S^v_i,t,t+cH,\mathcal{G}) \le cHu^v_i$.

Proof. **(a)** By (13), we have

$$\begin{aligned}
&\mathsf{lag}(S^v_i,t,\mathcal{G})\\
&= \sum_{j>ch^v} \mathsf{lag}(S^v_{i,j-ch^v},t,\mathcal{G})\\
&\le \{\text{By Lem. 21}\}\\
&\quad \sum_{j>ch^v} \mathsf{lag}(S^v_{i,j},t+cH,\mathcal{G})\\
&\le \{\text{By Lem. 22}\}\\
&\quad \sum_{1\le j\le ch^v} \mathsf{lag}(S^v_{i,j},t+cH,\mathcal{G}) + \sum_{j>ch^v} \mathsf{lag}(S^v_{i,j},t+cH,\mathcal{G})\\
&= \{\text{By (13)}\}\\
&\quad \mathsf{lag}(S^v_i,t+cH,\mathcal{G}).
\end{aligned}$$

(b) Assume that for S^v_i and time instant $t \ge O^v$, $\mathsf{A}(S^v_i,t,t+cH,\mathcal{G}) > cHu^v_i$ holds. Then, by (10), we have $\mathsf{A}(S^v_i,t,t+$

$cH, \mathcal{I}) - \mathsf{A}(S_i^v, t, t+cH, \mathcal{G}) < cHu_i^v - cHu_i^v = 0$. Thus, by (15), $\mathsf{lag}(S_i^v, t+cH, \mathcal{G}) < \mathsf{lag}(S_i^v, t, \mathcal{G})$, which contradicts (a). □

Lemma 24. *For any integer c and time instant $t \geq O_{max}$,*

(a) $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) \leq \mathsf{LAG}(\Gamma_s, t+cH, \mathcal{G})$,

(b) $\mathsf{A}(\Gamma_s, t, t+cH, \mathcal{G}) \leq cHU$.

Proof. **(a)** By Lem. 23(a), we have $\sum_{v,i} \mathsf{lag}(S_i^v, t, \mathcal{G}) \leq \sum_{v,i} \mathsf{lag}(S_i^v, t+cH, \mathcal{G})$. Thus, by (14), the lemma holds.

(b) By Lem. 23(b), we have $\sum_{v,i} \mathsf{A}(S_i^v, t, t+cH, \mathcal{G}) \leq \sum_{v,i} cHu_i^v$. Since $\sum_{v,i} u_i^v = U$, by (6), we have $\mathsf{A}(\Gamma_s, t, t+cH, \mathcal{G}) \leq cHU$. □

We now address Step 2. We will show, in Lem. 31, that if $\mathsf{A}(\Gamma_s, t, t+H, \mathcal{G}) = HU$ holds, then scheduling decisions in $\mathcal{G}$ are identical at times t and $t+H$. We begin by proving some lemmas (Lems. 25–28) that establish server- and server-job-level properties at times t and $t+cH$, when Γ_s's LAG values are equal. The next lemma follows from (16) and (11).

Lemma 25. *For any positive integer c and time instant $t \geq O_{max}$, $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) = \mathsf{LAG}(\Gamma_s, t+cH, \mathcal{G})$ holds if and only if $\mathsf{A}(\Gamma_s, t, t+cH, \mathcal{G}) = cHU$.*

By Lem. 23(a), we have the following lemma.

Lemma 26. *For any positive integer c and time instant $t \geq O_{max}$, if $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) = \mathsf{LAG}(\Gamma_s, t+cH, \mathcal{G})$ holds, then for any S_i^v, $\mathsf{lag}(S_i^v, t, \mathcal{G}) = \mathsf{lag}(S_i^v, t+cH, \mathcal{G})$ holds.*

Similarly, by Lem. 21, we have the following lemma.

Lemma 27. *For any positive integer c and time instant $t \geq O_{max}$, if $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) = \mathsf{LAG}(\Gamma_s, t+cH, \mathcal{G})$ holds, then for any server job $S_{i,j}^v$, the following hold.*

(a) $\mathsf{lag}(S_{i,j}^v, t, \mathcal{G}) = \mathsf{lag}(S_{i,j+ch^v}^v, t+cH, \mathcal{G})$.

(b) $\mathsf{A}(S_{i,j}^v, 0, t, \mathcal{G}) = \mathsf{A}(S_{i,j+ch^v}^v, 0, t+cH, \mathcal{G})$.

Lemma 28. *Assume that a server S_i^v, job index j, integer c, and time t exist such that $j + ch^v \geq 1$, $\min\{t, t+cH\} \geq O^v$, and $\mathsf{A}(S_{i,j}^v, 0, t, \mathcal{G}) = \mathsf{A}(S_{i,j+ch^v}^v, 0, t+cH, \mathcal{G})$. Then, $f(S_{i,j}^v) \leq t$ if and only if $f(S_{i,j+ch^v}^v) \leq t+cH$.*

Proof. **Necessity.** Assume that $f(S_{i,j}^v) \leq t$ holds. Then, we have $\mathsf{A}(S_{i,j}^v, 0, t, \mathcal{G}) = C_i^v$. Therefore, $\mathsf{A}(S_{i,j+ch^v}^v, 0, t+cH, \mathcal{G}) = C_i^v$ holds. Thus, $f(S_{i,j+ch^v}^v) \leq t+cH$.

Sufficiency. Assume that $f(S_{i,j}^v) > t$ holds. Then, we have $\mathsf{A}(S_{i,j}^v, 0, t, \mathcal{G}) < C_i^v$. Therefore, $\mathsf{A}(S_{i,j+ch^v}^v, 0, t+cH, \mathcal{G}) < C_i^v$ holds. Thus, $f(S_{i,j+ch^v}^v) > t+cH$. □

Similar to Lem. 2, we have the following lemma.

Lemma 29. *For any positive integers j and k such that $j \leq k$ and time instant t, $\mathsf{A}(S_{i,j}^v, 0, t, \mathcal{G}) \geq \mathsf{A}(S_{i,k}^v, 0, t, \mathcal{G})$ holds.*

If the schedule over the interval $[t, t+H)$ in $\mathcal{G}$ repeats during $[t+H, t+2H)$, then for each server job $S_{i,j}^v$ that is ready (resp., not ready) at time t, $S_{i,j+h^v}^v$ must be ready (resp., not ready) at time $t+H$. The lemma below shows that this condition holds if Γ_s's LAG values at time t and $t+H$ are the same.

Lemma 30. *If there is a time instant $t \geq O_{max}$ such that $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) = \mathsf{LAG}(\Gamma_s, t+H, \mathcal{G})$ holds, then $S_{i,j}^v$ is ready at time t if and only if $S_{i,j+h^v}^v$ is ready at time $t+H$.*

Proof. By Lem. 27, we have

$$\forall v, i, k : \mathsf{A}(S_{i,k}^v, 0, t, \mathcal{G}) = \mathsf{A}(S_{i,k+h^v}^v, 0, t+H, \mathcal{G}). \quad (22)$$

Sufficiency. Assume that $S_{i,j+h^v}^v$ is ready at time $t+H$, but $S_{i,j}^v$ is not ready at time t. Since $S_{i,j+h^v}^v$ is ready (hence, pending) at time $t+H$, by (22) and Lem. 15 (replacing j, t, and c with $j+h^v$, $t+H$, and -1, respectively), $S_{i,j}^v$ is pending at time t. Since $S_{i,j}^v$ is not ready at time t, by Def. 1, $j > P_i^v$ and $f(S_{i,j-P_i^v}^v) > t$ hold. By (22) and Lem. 28, we have $f(S_{i,j+h^v-P_i^v}^v) > t+H$. Thus, by Def. 1, $S_{i,j+h^v}^v$ is not ready at time $t+H$, a contradiction.

Necessity. Assume that $S_{i,j}^v$ is ready at time t, but $S_{i,j+h^v}^v$ is not ready at time $t+H$. Since $S_{i,j}^v$ is ready at time t, by (22) and Lem. 15, $S_{i,j+h^v}^v$ is pending at time t. Since $S_{i,j+h^v}^v$ is not ready at time $t+H$, by Def. 1, $j + h^v > P_i^v$ and $f(S_{i,j+h^v-P_i^v}^v) > t+H$ hold. We now consider two cases.

Case 1. $j > P_i^v$. By (22) and Lem. 28, $f(S_{i,j-P_i^v}^v) > t$. Thus, by Def. 1, $S_{i,j}^v$ is not ready at time t. Contradiction.

Case 2. $j \leq P_i^v$. In this case, $j + h^v - P_i^v \leq P_i^v + h^v - P_i^v = h^v$. Since $S_{i,j+h^v}^v$ is pending but not ready at time $t+H$, $\mathsf{A}(S_{i,j+h^v}^v, 0, t+H, \mathcal{G}) = 0$. By Lem. 29, for each $b \geq j$, $\mathsf{A}(S_{i,b+h^v}^v, 0, t+H, \mathcal{G}) = 0$ holds. Therefore, we have

$$\begin{aligned}
&\mathsf{A}(S_i^v, t, t+H, \mathcal{G}) \\
&= \sum_{1 \leq b \leq j+h^v} \mathsf{A}(S_{i,b}^v, t, t+H, \mathcal{G}) \\
&= \{\text{By (8)}\} \\
&\quad \sum_{1 \leq b \leq j+h^v} (\mathsf{A}(S_{i,b}^v, 0, t+H, \mathcal{G}) - \mathsf{A}(S_{i,b}^v, 0, t, \mathcal{G})). \quad (23)
\end{aligned}$$

By (22), $\mathsf{A}(S_{i,b}^v, 0, t, \mathcal{G}) = \mathsf{A}(S_{i,b+h^v}^v, 0, t+H, \mathcal{G})$ for each $1 \leq b \leq j$. Thus, for each $1 \leq b \leq j$, applying $\mathsf{A}(S_{i,b+h^v}^v, 0, t+H, \mathcal{G}) - \mathsf{A}(S_{i,b}^v, 0, t, \mathcal{G}) = 0$ in (23),

$$\begin{aligned}
&\mathsf{A}(S_i^v, t, t+H, \mathcal{G}) \\
&= \sum_{1 \leq b \leq h^v} \mathsf{A}(S_{i,b}^v, 0, t+H, \mathcal{G}) - \sum_{j+1 \leq b \leq j+h^v} \mathsf{A}(S_{i,b}^v, 0, t, \mathcal{G}) \\
&\leq \sum_{1 \leq b \leq h^v} \mathsf{A}(S_{i,b}^v, 0, t+H, \mathcal{G}) \\
&= \{\text{ Since } j + h^v - P_i^v \leq h^v\} \\
&\quad \mathsf{A}(S_{i,j+h^v-P_i^v}^v, 0, t+H, \mathcal{G}) + \sum_{\substack{1 \leq b \leq h^v \wedge \\ b \neq j+h^v-P_i^v}} \mathsf{A}(S_{i,b}^v, 0, t+H, \mathcal{G}) \\
&< \{\text{Since } \mathsf{A}(S_{i,b}^v, 0, t+H, \mathcal{G}) \leq C_i^v \text{ and } S_{i,j+h^v-P_i^v}^v \text{ is} \\
&\qquad \text{pending at time } t+H\} \\
&\quad C_i^v + (h^v - 1)C_i^v \\
&= \{\text{By Def. 4}\} \\
&\quad Hu_i^v. \quad (24)
\end{aligned}$$

By (15), (10), and (24), we have $\mathsf{lag}(S_i^v, t+H, \mathcal{G}) > \mathsf{lag}(S_i^v, t, \mathcal{G}) + Hu_i^v - Hu_i^v = \mathsf{lag}(S_i^v, t, \mathcal{G})$, which contradicts Lem. 26. □

We now complete Step 2 by giving the following lemma. Using Lem. 30, we show that if the time allocated to Γ_s over an interval $[t, t+H)$ equals HU, then $S^v_{i,j}$ is scheduled at time t if and only if $S^v_{i,j+h^v}$ is scheduled at time $t+H$.

Lemma 31. *For any $t \geq O_{max}$, if $\mathsf{A}(\Gamma_s, t, t+H, \mathcal{G}) = HU$ holds, then the following hold.*

(a) *For any $S^v_{i,j}$, $\mathsf{A}(S^v_{i,j}, t, t+1, \mathcal{G}) = \mathsf{A}(S^v_{i,j+h^v}, t+H, t+H+1, \mathcal{G})$.*

(b) *For any S^v_i, $\mathsf{A}(S^v_i, t, t+1, \mathcal{G}) = \mathsf{A}(S^v_i, t+H, t+H+1, \mathcal{G})$.*

(c) $\mathsf{A}(\Gamma_s, t, t+1, \mathcal{G}) = \mathsf{A}(\Gamma_s, t+H, t+H+1, \mathcal{G})$.

Proof. **(a)** By Lem. 25, we have $\mathsf{LAG}(\Gamma_s, t, \mathcal{G}) = \mathsf{LAG}(\Gamma_s, t+H, \mathcal{G})$. Thus, by Lem. 30, $S^v_{i,j}$ is ready at time t if and only if $S^v_{i,j+h^v}$ is ready at time $t+H$. By Lem. 10, $d(S^v_{i,j+h^v}) = d(S^v_{i,j}) + H$ holds. Thus, for each pair of server jobs $S^v_{i,j}$ and $S^u_{k,\ell}$, we have $d(S^v_{i,j}) - d(S^u_{k,\ell}) = d(S^v_{i,j+h^v}) - d(S^u_{k,\ell+h^u})$. Since ties are broken consistently, $S^v_{i,j}$ has higher priority than $S^u_{k,\ell}$ if and only if $S^v_{i,j+h^v}$ has higher priority than $S^u_{k,\ell+h^u}$. Thus, $S^v_{i,j}$ is the p^{th} highest-priority ready job at time t if and only if $S^v_{i,j+h^v}$ is the p^{th} highest-priority ready job at time $t+H$. Therefore, $S^v_{i,j}$ is scheduled at time t if and only if $S^v_{i,j+h^v}$ is scheduled at time $t+H$. Thus, (a) holds.

(b) Follows from (a) and (5).

(c) Follows from (b) and (6). □

We have the following lemma, which is useful for Step 3.

Lemma 32. *For any positive integer c and time $t' \geq O_{max}$, if $\mathsf{A}(\Gamma_s, t', t'+cH, \mathcal{G}) = cHU$ holds, then, for each time instant $t \in [t', t'+(c-1)H]$, $\mathsf{A}(\Gamma_s, t, t+H, \mathcal{G}) = HU$ holds.*

Proof. We first prove the following claim.

Claim 32.1. $\mathsf{A}(\Gamma_s, t', t'+H, \mathcal{G}) = HU$.

Proof. For $c = 1$, the claim holds by the lemma assumptions, so assume $c \geq 2$. Assume for a contradiction that $\mathsf{A}(\Gamma_s, t', t'+H, \mathcal{G}) \neq HU$. Then, by Lem. 24(b), we have $\mathsf{A}(\Gamma_s, t', t'+H, \mathcal{G}) < HU$. Since $[t', t'+cH) = \cup_{i=0}^{c-1}[t'+iH, t'+(i+1)H)$, we have

$$\begin{aligned}
&\mathsf{A}(\Gamma_s, t', t'+cH, \mathcal{G}) \\
&= \mathsf{A}(\Gamma_s, t', t'+H, \mathcal{G}) + \sum_{i=1}^{c-1} \mathsf{A}(\Gamma_s, t'+iH, t'+(i+1)H, \mathcal{G}) \\
&< \{\text{By Lem. 24(b) and since } \mathsf{A}(\Gamma_s, t', t'+H, \mathcal{G}) < HU\} \\
&\quad HU + (c-1)HU \\
&= cHU,
\end{aligned}$$

a contradiction. □

We now prove the lemma. Assume for a contradiction that time $t \in [t', t'+(c-1)H]$ exists such that $\mathsf{A}(\Gamma_s, t, t+H, \mathcal{G}) \neq HU$. By Clm. 32.1, $t > t'$. Thus, $\mathsf{A}(\Gamma_s, t-1, t+H-1, \mathcal{G}) = HU$. Since $[t, t+H) = ([t-1, t+H-1) \cup [t+H-1, t+H)) \setminus [t-1, t)$, we have $\mathsf{A}(\Gamma_s, t, t+H, \mathcal{G}) = \mathsf{A}(\Gamma_s, t-1, t+H-1, \mathcal{G}) + \mathsf{A}(\Gamma_s, t+H-1, t+H, \mathcal{G}) - \mathsf{A}(\Gamma_s, t-1, t, \mathcal{G})$, which by Lem. 31(c) equals $\mathsf{A}(\Gamma_s, t-1, t+H-1, \mathcal{G}) = HU$, a contradiction. □

C. Analysis of DAG Tasks.

We now give an analysis of schedule $\mathcal{S}$ that completes Steps 3 and 4. We begin by showing, in Lems. 33–38 that there are properties of lag and LAG in $\mathcal{S}$ that are analogous to the properties in $\mathcal{G}$. Intuitively, these properties hold as a job of Γ can execute only when its linked server job is scheduled.

Lemma 33. *If $\tau^v_{i,j}$ is linked to $S^v_{i,k}$, then for any time instant t, $\mathsf{A}(\tau^v_{i,j}, 0, t, \mathcal{S}) = \mathsf{A}(S^v_{i,k}, 0, t, \mathcal{G})$ holds.*

Proof. Follows from the budget Consumption Rule, Assumption A, and Rule R3. □

By Lems. 5, 29, and 33, we have the following lemma.

Lemma 34. *For any positive integers j and k such that $j < k$ and time instant t, $\mathsf{A}(\tau^v_{i,j}, 0, t, \mathcal{S}) \geq \mathsf{A}(\tau^v_{i,k}, 0, t, \mathcal{S})$ holds.*

Lemma 35. *For any job $\tau^v_{i,j}$, positive integer c, and time instant $t \geq O^v$, $\mathsf{A}(\tau^v_{i,j}, 0, t, \mathcal{S}) \geq \mathsf{A}(\tau^v_{i,j+ch^v}, 0, t+cH, \mathcal{S})$ holds.*

Proof. Assume $\tau^v_{i,j}$ (resp., $\tau^v_{i,j+ch^v}$) is linked to $S^v_{i,k}$ (resp., $S^v_{i,\ell}$). By Lem. 19, $\mathsf{A}(S^v_{i,k}, 0, t, \mathcal{G}) \geq \mathsf{A}(S^v_{i,k+ch^v}, 0, t+cH, \mathcal{G})$. By Lem. 5, $\ell \geq k + ch^v$. Thus, by Lem. 29, we have $\mathsf{A}(S^v_{i,k+ch^v}, 0, t+cH, \mathcal{G}) \geq \mathsf{A}(S^v_{i,\ell}, 0, t+cH, \mathcal{G})$. Hence, $\mathsf{A}(S^v_{i,k}, 0, t, \mathcal{G}) \geq \mathsf{A}(S^v_{i,\ell}, 0, t+cH, \mathcal{G})$ holds. By Lem. 33, $\mathsf{A}(\tau^v_{i,j}, 0, t, \mathcal{S}) \geq \mathsf{A}(\tau^v_{i,j+ch^v}, 0, t+cH, \mathcal{S})$. □

By Lem. 35, we can prove lemmas analogous to Lems. 20–27 and 32 for Γ. Among those, we list the important ones that we use in the later proofs.

Lemma 36. *For any positive integer c and time $t \geq O_{max}$, if $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t+cH, \mathcal{S})$, then for any job $\tau^v_{i,j}$, the following hold.*

(a) $\mathsf{lag}(\tau^v_{i,j}, t, \mathcal{S}) = \mathsf{lag}(\tau^v_{i,j+ch^v}, t+cH, \mathcal{S})$.

(b) $\mathsf{A}(\tau^v_{i,j}, 0, t, \mathcal{S}) = \mathsf{A}(\tau^v_{i,j+ch^v}, 0, t+cH, \mathcal{S})$.

Lemma 37. *For any positive integer c and time $t \geq O_{max}$, if $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t+cH, \mathcal{S})$, then the following hold.*

(a) $\mathsf{A}(\Gamma_s, t, t+cH, \mathcal{G}) = cHU$.

(b) *If a server job $S^v_{i,j}$ is scheduled at time $t' \in [t, t+cH)$, then a job is linked to it.*

Lemma 38. *For any positive integer c and time $t' \geq O_{max}$, if $\mathsf{LAG}(\Gamma, t', \mathcal{S}) = \mathsf{LAG}(\Gamma, t'+cH, \mathcal{S})$ holds, then for each $t \in [t', t'+(c-1)H]$, $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t+H, \mathcal{S})$.*

Def. 6. *Let $\Delta = \lceil \max_{v,i}\{R(S^v_i)\}/H \rceil H$. Note that $\Delta \geq \max_{v,i}\{R(S^v_i)\}$ and $\Delta \geq H$ hold.*

We now address Step 3 by giving Lems. 39–41. The repetition of the graph-level schedule $\mathcal{S}$ at time t' requires that the server-level schedule $\mathcal{G}$ repeats at time t' and each server job scheduled at or after t' has a linked job. To ensure the latter, we need to consider a larger interval of length $(2H + \Delta)$ as shown in the following lemma.

Lemma 39. *If $\mathsf{LAG}(\Gamma, t_s, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_s+2H+\Delta, \mathcal{S})$ holds such that $t_s \geq O_{max}$, then the following hold.*

(a) *If a server job $S^v_{i,j}$ is released during $[t_s, t_s+2H)$, then a job is linked to it.*

(b) *For each v and $j \leq h^v$, $f(\tau^v_{n^v,j}) \leq t_s + 2H + \Delta$ holds.*

Proof. **(a)** Assume that there is a server job $S^v_{i,j}$ such that $t_s \leq r(S^v_{i,j}) < t_s + 2H$ holds, but no job is linked to it. Since $S^v_{i,j}$'s response time is at most $R(S^v_i)$, by Def. 6, we have $f(S^v_{i,j}) \leq r(S^v_{i,j}) + R(S^v_i) < t_s + 2H + \Delta$. Therefore, there is a time instant t_b such that $t_s \leq t_b < t_s + 2H + \Delta$ and $S^v_{i,j}$ is scheduled during $[t_b, t_b + 1)$. Since H divides Δ, by Lem. 37(b), a job is linked to $S^v_{i,j}$, a contradiction.

(b) We first prove the following claim.

Claim 39.1. $f(\tau^v_{n^v,1}) \leq t_s + H + \Delta$.

Proof. Let $S^v_{n^v,k}$ be the first server job of $S^v_{n^v}$ that is released at or after t_s. Since server jobs are released periodically and $t_s \geq O_{max}$, $r(S^v_{n^v,k}) < t_s + H$ holds. Since $S^v_{n^v,k}$'s response time is at most $R(S^v_{n^v})$, by Def. 6, we have $f(S^v_{n^v,k}) < t_s + H + R(S^v_{n^v}) \leq t_s + H + \Delta$. By (a), $S^v_{n^v,k}$ is linked to a job $\tau^v_{n^v,j}$. By Lem. 7, $f(\tau^v_{n^v,j}) \leq f(S^v_{n^v,k})$. If $j = 1$, then the claim holds, so assume $j > 1$. Let $S^v_{n^v,\ell}$ be the server job to which $\tau^v_{n^v,1}$ is linked. By Lems. 5 and 2, $\ell < k$ and $f(S^v_{n^v,\ell}) \leq f(S^v_{n^v,k})$ hold. Therefore, by Lem. 7, $f(\tau^v_{n^v,1}) \leq f(S^v_{n^v,\ell}) \leq f(S^v_{n^v,k}) < t_s + H + \Delta$ holds. $\square$

We now prove the lemma. By Clm 39.1, we have $f(\tau^v_{n^v,1}) \leq t_s + H + \Delta$. Hence, since H divides Δ and $\mathsf{LAG}(\Gamma, t_s, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_s + 2H + \Delta, \mathcal{S})$, by Lem. 38, we have $\mathsf{LAG}(\Gamma, t_s + H + \Delta, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_s + 2H + \Delta, \mathcal{S})$. Thus, by Lem. 36(b), we have $\mathsf{A}(\tau^v_{n^v,h^v+1}, 0, t_s + 2H + \Delta, \mathcal{S}) = \mathsf{A}(\tau^v_{n^v,1}, 0, t_s + H + \Delta, \mathcal{S}) = C^v_{n^v}$. By Lem. 34, for each $j \leq h^v$, we have $\mathsf{A}(\tau^v_{n^v,j}, 0, t_s + 2H + \Delta, \mathcal{S}) = C^v_{n^v}$. Thus, for each $j \leq h^v$, $\tau^v_{n^v,j}$ completes at or before time $t_s + 2H + \Delta$. $\square$

Using Lems. 37–39, we now prove that if Γ's LAG values at time t and $t + 2H + \Delta$ are the same, then this value remains the same over any future H-sized interval.

Lemma 40. *If $\mathsf{LAG}(\Gamma, t_s, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_s + 2H + \Delta, \mathcal{S})$ holds such that $t_s \geq O_{max}$, then for each $t \geq t_s$, $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t + H, \mathcal{S})$ holds.*

Proof. Let t be the first time instant at or after t_s such that $\mathsf{LAG}(\Gamma, t, \mathcal{S}) \neq \mathsf{LAG}(\Gamma, t + H, \mathcal{S})$ holds. Since H divides Δ, by Lem. 38 and Def. 6, we have the following.

$$t > t_s + H + \Delta \wedge t > t_s + 2H \tag{25}$$

We first prove the following claim.

Claim 40.1. *If a server job $S^v_{i,j}$ is released during $[t_s + 2H, t)$, then a job is linked to it.*

Proof. Assume otherwise. Let $S^v_{i,j}$ be the first job of S^v_i released during $[t_s + 2H, t)$ to which no job is linked. Let $t_r = r(S^v_{i,j})$. By Lem. 39(a) and the definition of t_r, we have:

(P) Each server job of S^v_i released during $[t_s, t_r)$ has a job that is linked to it.

Since H divides Δ and $t_r \in [t_s + 2H, t)$, by the definition of t, we have

$$\mathsf{LAG}(\Gamma, t_r - H, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_r, \mathcal{S}). \tag{26}$$

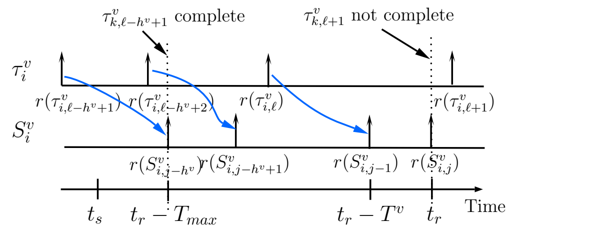


Fig. 4: Illustration of Clm 40.1. Blue arrows from job releases to server job releases represent linking.

Since server jobs are released periodically, $r(S^v_{i,j-1}) = t_r - T^v \geq t_r - H \geq t_s + 2H - H = t_s + H$. Thus, by (P), a job $\tau^v_{i,\ell}$ is linked to $S^v_{i,j-1}$. We now prove that $\tau^v_{i,\ell+1}$ is linked to $S^v_{i,j}$, thereby reaching a contradiction. By Rule R2, it suffices to prove that $r(\tau^v_{i,\ell+1}) \leq t_r$. We now prove the claim by considering two cases.

Case 1. $i = 1$. Thus, τ^v_i is the source node of G^v. Therefore, $r(\tau^v_{i,\ell+1}) = r(\tau^v_{i,\ell}) + T^v$. Since $\tau^v_{i,\ell}$ is linked to $S^v_{i,j-1}$, by Rule R2, we have $r(\tau^v_{i,\ell}) \leq r(S^v_{i,j-1}) = t_r - T^v$. Since τ^v_i releases job periodically, we have $r(\tau^v_{i,\ell+1}) \leq t_r$.

Case 2. $i > 1$. Thus, τ^v_i is a non-source node. Assume to the contrary that $r(\tau^v_{i,\ell+1}) > t_r$ (see Fig. 4). Since a non-source node's $(\ell+1)^{st}$ job is released once each of its predecessors' $(\ell+1)^{st}$ job completes, there is a job $\tau^v_{k,\ell+1}$ such that $\tau^v_k \in pred(\tau^v_i)$ and $f(\tau^v_{k,\ell+1}) > t_r$. Therefore, we have

$$\mathsf{A}(\tau^v_{k,\ell+1}, 0, t_r, \mathcal{S}) < C^v_k. \tag{27}$$

Since $t_r \in [t_s + 2H, t)$, we have $t_r - H \in [t_s, t)$. By Lem. 10, $S^v_{i,j-h^v}$ is released at time $t_r - H$ (thus, $j > h^v$). By (P), each server job of S^v_i released during $[t_r - H, t_r)$ has a job to which it is linked. Thus, since $S^v_{i,j-h^v}$ and $S^v_{i,j-1}$ are released at time $t_r - H$ and $t_r - T^v$, respectively, each server job $S^v_{i,j-b}$ such that $1 \leq b \leq h^v$ has a job to which it is linked. Since $\tau^v_{i,\ell}$ is linked to $S^v_{i,j-1}$, by Rule R2, for each $1 \leq b \leq h^v$, $\tau^v_{i,\ell-b+1}$ is linked to $S^v_{i,j-b}$. Thus, $\tau^v_{i,\ell-h^v+1}$ (hence, $\ell + 1 > h^v$) is linked to $S^v_{i,j-h^v}$. Since $\tau^v_{i,\ell-h^v+1}$ is linked to $S^v_{i,j-h^v}$ and $\tau^v_k \in pred(\tau^v_i)$, $f(\tau^v_{k,\ell-h^v+1}) \leq r(S^v_{i,j-h^v}) = t_r - H$. Therefore, we have $\mathsf{A}(\tau^v_{k,\ell-h^v+1}, 0, t_r - H, \mathcal{S}) = C^v_k$. By (26), Lem. 36(b), we have $\mathsf{A}(\tau^v_{k,\ell+1}, 0, t_r, \mathcal{S}) = \mathsf{A}(\tau^v_{k,\ell-h^v+1}, 0, t_r - H, \mathcal{S}) = C^v_k$, which contradicts (27). $\square$

Claim 40.2. *If a server job $S^v_{i,j}$ is scheduled during $[t-1, t)$, then a job is linked to it.*

Proof. By (25), $t > t_s + H + \Delta$ holds. By Def. 6, any server job released before t_s completes at or before time $t_s + \Delta < t$. Therefore, no server job released before t_s is pending at time $t - 1$. Thus, $S^v_{i,j}$ is released at or after time t_s. By Lem. 39(a) and Clm 40.1, a job is linked to $S^v_{i,j}$. $\square$

Using the above claims, we now prove the lemma. By (25) and the definition of t, we have

$$\mathsf{LAG}(\Gamma, t - H - 1, \mathcal{S}) = \mathsf{LAG}(\Gamma, t - 1, \mathcal{S}). \tag{28}$$

By (28) and Lem. 37(a), we have $\mathsf{A}(\Gamma_s, t - H - 1, t - 1, \mathcal{G}) = HU$. Therefore, by Lem. 31(c), we have

$$\mathsf{A}(\Gamma_s, t - H - 1, t - H, \mathcal{G}) = \mathsf{A}(\Gamma_s, t - 1, t, \mathcal{G}). \tag{29}$$

By (28) and Lem. 37(b), any server job scheduled during $[t-H-1, t-H)$ has a job linked to it. By Clm 40.2, any server job scheduled during $[t-1, t)$ has a job linked to it. Therefore, by (29), we have

$$\mathsf{A}(\Gamma, t-H-1, t-H, \mathcal{S}) = \mathsf{A}(\Gamma, t-1, t, \mathcal{S}). \quad (30)$$

By (28), (11), and (30), we have $\mathsf{LAG}(\Gamma, t-1, \mathcal{S}) + \mathsf{A}(\Gamma, t-1, t, \mathcal{I}) - \mathsf{A}(\Gamma, t-1, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t-H-1, \mathcal{S}) + \mathsf{A}(\Gamma, t-H-1, t-H, \mathcal{I}) - \mathsf{A}(\Gamma, t-H-1, t-H, \mathcal{S})$. Thus, by (16), $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t-H, \mathcal{S})$ holds, a contradiction. □

We now complete Step 3 by giving the following lemma.

Lemma 41. *If $\mathsf{LAG}(\Gamma, t_s, \mathcal{S}) = \mathsf{LAG}(\Gamma, t_s + 2H + \Delta, \mathcal{S})$ holds such that $t_s \geq O_{max}$ and R^v is the maximum response time of DAG jobs of G^v that complete at or before time $t_s + 2H + \Delta$ in $\mathcal{S}$, then the response time of G^v is R^v in $\mathcal{S}$.*

Proof. Assume for a contradiction that G^v_j is the first DAG job of G^v with response time more than R^v. Assume that $\tau^v_{n^v,j}$ completes at time t, *i.e.*, $t = f(\tau^v_{n^v,j})$. Thus, we have

$$t - r(\tau^v_{1,j}) > R^v. \quad (31)$$

Since R^v is the maximum observed response time of G^v at or before $t_s + 2H + \Delta$, we have $t > t_s + 2H + \Delta$. Therefore, by Lem. 39(b), $j > h^v$ holds. At time $t-1$, $\tau^v_{n^v,j}$ is pending. Thus, $\mathsf{A}(\tau^v_{n^v,j}, 0, t-1, \mathcal{S}) < C^v_i$. Since $t-1 \geq t_s + 2H + \Delta$, by Lem. 40, we have $\mathsf{LAG}(\Gamma, t-1, \mathcal{S}) = \mathsf{LAG}(\Gamma, t-H-1, \mathcal{S})$. Then, by Lem. 36(b), $\mathsf{A}(\tau^v_{n^v,j-h^v}, 0, t-H-1, \mathcal{S}) = \mathsf{A}(\tau^v_{n^v,j}, 0, t-1, \mathcal{S}) < C^v_i$. Thus, $\tau^v_{n^v,j-h^v}$ completes after time $t-H-1$, *i.e.*, $f(\tau^v_{n^v,j-h^v}) \geq t-H$. Thus, we have $f(\tau^v_{n^v,j-h^v}) - r(\tau^v_{1,j-h^v}) \geq t - H - r(\tau^v_{1,j-h^v})$. By Lem. 10, $t - H - r(\tau^v_{1,j-h^v}) = t - r(\tau^v_{1,j})$, which by (31), exceeds R^v. Therefore, we have $f(\tau^v_{n^v,j-h^v}) - r(\tau^v_{1,j-h^v}) > R^v$. Thus, $G^v_{j-h^v}$'s response time is more than R^v, a contradiction. □

We now complete Step 4. Our goal is to show that there exists a time instant t_s such that Γ's LAG values at time t_s and $t_s + 2H + \Delta$ are the same. This, by Lem. 41, implies that a DAG job with the maximum response time completes execution at or before time $t_s + 2H + \Delta$. We first give an upper bound and a lower bound of LAG of Γ at any time instant.

Def. 7. *Let $E = \sum_{v=1}^{N} \sum_{i=1}^{n^v} R(\tau^v_i) u^v_i$, $F = \sum_{v=1}^{N} \sum_{i=1}^{n^v} C^v_i$, and $G = \lceil E + F + 1 \rceil$.*

Since τ^v_i's response time is at most $R(\tau^v_i)$ (by Thm. 1), we can show that τ^v_i's lag at any time is at most $R(\tau^v_i) u^v_i$ (the proof is given in an online appendix [2]). Since no job executes before its release, τ^v_i's lag at any time is at least $-C^v_i$. Using these, we have the following lemma.

Lemma 42. *For any time instant t, $-F \leq \mathsf{LAG}(\Gamma, t, \mathcal{S}) \leq E$.*

Since all task parameters are integers, if for any positive c, $\mathsf{LAG}(\Gamma, t + cH, \mathcal{S}) > \mathsf{LAG}(\Gamma, t, \mathcal{S})$ holds, then Γ's LAG at time $t + cH$ is at least one unit larger than its LAG at time t. Therefore, since LAG either increases or remains the same over any interval $[t, t + cH)$, it cannot increase over G consecutive intervals of size cH without violating the LAG upper bound. Thus, we have the following lemma.

Lemma 43. *There is a time instant $t \in [O_{max}, O_{max} + G(2H + \Delta)]$ such that $\mathsf{LAG}(\Gamma, t, \mathcal{S}) = \mathsf{LAG}(\Gamma, t + 2H + \Delta, \mathcal{S})$ holds.*

Thus, by Lem. 43 and 41, we have the following theorem.

Theorem 2. *If R^v is the maximum response time of any DAG job of G^v completed at or before $O_{max} + (G+1)(2H + \Delta)$, then G^v's response time is R^v.*

Thus, simulating schedule $\mathcal{S}$ for at most $O_{max} + (G + 1)(2H + \Delta)$ time units is sufficient to determine the maximum response times of DAGs. However, the simulation can be terminated early by checking whether the condition given in Lem. 41 is met. For pseudo-harmonic task systems, by Def. 7 (resp., Def. 6 and (1)), G (resp., Δ) is polynomial with respect to the task and processor count and task parameters. Thus, for pseudo-harmonic task systems, simulating for $O_{max} + (G+1)(2H + \Delta)$ time takes pseudo-polynomial time.

Removing Assumption A. Let $\mathcal{S}'$ be a schedule of Γ when Assumption A does not hold. Thm. 3 below ensures that no job finishes later in $\mathcal{S}'$ than $\mathcal{S}$. Informally, no job is linked to a later server job in $\mathcal{S}'$ than in $\mathcal{S}$.

Theorem 3. *For each job $\tau^v_{i,j}$, if it completes at time t and t' in $\mathcal{S}$ and $\mathcal{S}'$, respectively, then $t' \leq t$ holds.*

Slack reallocation. The response times of DAG tasks may potentially be improved by utilizing budgets of server jobs that have no linked job. Assuming $S^v_{i,j}$ is scheduled at time t, we propose the following slack reallocation policy.

Q1 If $S^v_{i,j}$ has no linked job or its linked job completes at or before time t, then the highest priority ready but unscheduled job of τ^v_i is scheduled on $S^v_{i,j}$ at time t.

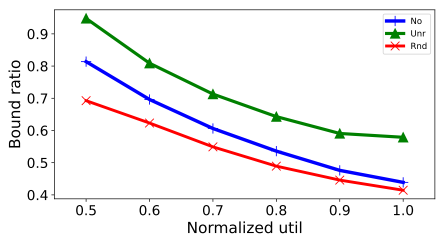
When each P^v_i equals 1, the bounds in Thm. 2 are also exact with slack reallocation. This is because the allocation received by each server over any H-sized interval is at most HU [1], which translates to a similar task-level property.

Asynchronous releases. Instead of synchronous server releases, asynchronous server releases are possible. We chose to limit attention to the former for simplicity (*e.g.*, asynchronous releases would necessitate different ideal schedules for tasks and servers). Thm. 2 remains valid with asynchronous server releases but with a different interval length.

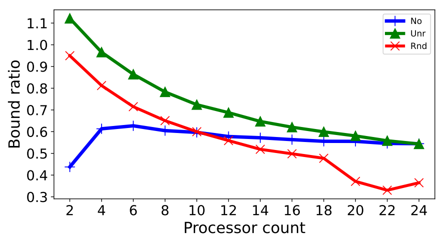
VI. Experiments

We now present the results of simulation experiments we conducted to evaluate the response-time bounds of our proposed scheduler. We compared our scheduler to other schedulers that provide bounded response times without utilization loss and are subject to (i)–(iv) mentioned in Sec. I.

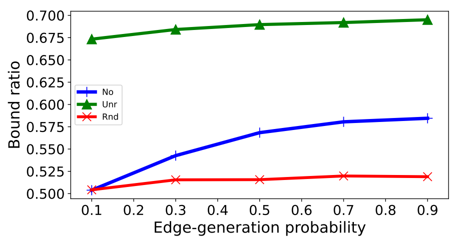
We generated task systems randomly for systems with 2 to 24 processors with a step size of 2.0. Such processor counts are common in real-world use cases [3], [15]. For each processor count, we generated task systems that have *normalized utilization*, *i.e.*, U/m, from 0.5 to 1 with a step size of 0.1. Each task system consists of one or more DAGs. The number of DAGs was chosen uniformly from $[1, \lfloor U/2 \rfloor]$. Motivated by automotive use cases, each DAG's period was uniformly selected from



(a) Bound ratio vs. normalized utilization.



(b) Bound ratio vs. processor count.



(c) Bound ratio vs. edge-generation probability.

Fig. 5: Experimental results.

$\{1, 2, 5, 10, 20, 50, 100, 200\}$ms [16]. The offset of each DAG was uniformly selected between 0 and its period. The number of nodes per DAG was chosen uniformly from $[10, 100)$. Each node's utilization was chosen uniformly following procedures from [10]. The WCET of each node was rounded to the nearest microsecond. Edges were generated following the *Erdős-Rényi method* [8], where an edge was added between two nodes if a uniformly generated number in $[0, 1]$ is at most a predefined *edge-generation probability*. We selected this probability value from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. As in [25], a minimum number of additional edges was added to make each DAG weakly connected. Each edge was directed from a lower-indexed task to a higher-indexed task. For each combination of processor count, normalized utilization, and edge-generation probabilities, we generated 1,000 random task systems.

We considered three scenarios for each generated task system depending on task parallelism levels. In scenarios No and Unr, each task's parallelism level was set to 1 and m, respectively. In scenario Rnd, task parallelism levels were generated uniformly between 1 and m. For scenarios No, Unr, and Rnd, we compared our response-time bounds with those from [18], [32], and [4], respectively. These works convert each DAG task into an "equivalent" independent sporadic task set and schedule the converted tasks by GEDF. The response-time bounds from these prior works are non-exact and can be computed in polynomial time. For each scenario, we computed the average *bound ratio*, which is the ratio of the average response-time bound of our method to that of the corresponding prior method (so ratios below 1.0 show improvement by our method). These ratios are plotted in Fig. 5(a)–(c).

Observation 1. *For No, Rnd, and Unr, the average improvement of our bound over prior methods was around 43%, 48%, and 31%, respectively.*

The improvement is due to the pessimism inherent to prior bounds. Prior bounds that consider arbitrary parallelism levels suffer from pessimism present in the analysis of both no and unrestricted parallelism. This yields a larger improvement for the Rnd scenario. The improvement is less for Unr as prior analysis with unrestricted parallelism is less pessimistic. Note that asynchronous server releases, as discussed earlier, may yield additional improvement.

Observation 2. *Our method provided a larger improvement with increasing (resp., decreasing) normalized utilization (resp., edge-generation probabilities). Except No, our method provided larger improvement as the processor count increases.*

This can be seen in Fig. 5(a)–(c). The large improvement for higher normalized utilizations or processor counts is due to the increased pessimism in the corresponding prior analysis. For scenario No, the large improvement for small processor counts is due to the usage of slack reallocation. In contrast, for scenario Unr and small processor counts, the prior method gave smaller bounds. This happens because prior analysis is reasonably tight under the corresponding scheduling policy for small processor counts, while jobs may be delayed waiting for their linked server jobs in our scheduling strategy. With increasing edge-generation probabilities, DAGs become more *sequential*, which limits improvement under our method.

To determine the tightness of the simulation length, we computed the analytical simulation length from Thm. 2 and the actual simulation length by checking when the condition given in Lem. 41 is met for the first time. The observation below indicates that the analytical simulation length is pessimistic.

Observation 3. *The average analytical simulation length (from Thm. 2) is 3,564,060 times larger than the average actual simulation length.*

Finally, we note that our method was reasonably fast. However, the execution time of our method depends on the hyperperiod and the granularity of time units.

Observation 4. *The average (resp., maximum) simulation time (on a 24-core 2.50 GHz machine) was 6.83s (resp., 923.56s). The average (resp., maximum) time to compute prior bound was 0.10s (resp., 9.71s).*

VII. Conclusion

We have presented a server-based scheduling policy for DAG tasks and a method to compute exact response-time bounds under this policy. We have focused on a generalized DAG task model, where both inter-instance dependencies and intra-task parallelism are allowed. Moreover, our method takes pseudo-polynomial time for pseudo-harmonic DAG tasks.

In future work, we plan to investigate exact response-time bounds of DAG tasks under common schedulers, *e.g.*, GEDF, that do not require servers. We also want to investigate exact response-time bounds for non-preemptive DAG tasks and other DAG task models, *e.g.*, *conditional DAGs*.

References

[1] S. Ahmed and J. Anderson, "Tight tardiness bounds for pseudo-harmonic tasks under global-EDF-like schedulers," in *ECRTS'21*, 2021, pp. 11:1–11:24.

[2] ——, "Exact response-time bounds of periodic DAG tasks under server-based global scheduling (longer version)," 2022. [Online]. Available: http://jamesanderson.web.unc.edu/papers/

[3] B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "An empirical survey-based study into industry practice in real-time systems," in *RTSS'20*, 2020, pp. 3–11.

[4] T. Amert, S. Voronov, and J. Anderson, "OpenVX and real-time certification: The troublesome history," in *RTSS'19*, 2019, pp. 312–325.

[5] S. Baruah, "The federated scheduling of constrained-deadline sporadic DAG task systems," in *DATE'15*, 2015, pp. 1323–1328.

[6] ——, "Federated scheduling of sporadic DAG task systems," in *ISORC'15*, 2015, pp. 179–186.

[7] P. Chen, W. Liu, X. Jiang, Q. He, and N. Guan, "Timing-anomaly free dynamic scheduling of conditional DAG tasks on multi-core systems," *ACM Trans. Embed. Comput. Syst.*, vol. 18, no. 5s, pp. 91:1–91:19, 2019.

[8] D. Cordeiro, G. Mounié, S. Perarnau, D. Trystram, J. Vincent, and F. Wagner, "Random graph generation for scheduling simulations," in *SIMUTools'10*, 2010, p. 60.

[9] L. Cucu-Grosjean and J. Goossens, "Exact schedulability tests for real-time scheduling of periodic tasks on unrelated multiprocessor platforms," *J. Syst. Archit.*, vol. 57, no. 5, pp. 561–569, 2011.

[10] P. Emberson, R. Stafford, and R. I. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *WATERS'10*, 2010, pp. 6–11.

[11] J. C. Fonseca, G. Nelissen, and V. Nélis, "Schedulability analysis of DAG tasks with arbitrary deadlines under global fixed-priority scheduling," *Real-Time Syst.*, vol. 55, no. 2, pp. 387–432, 2019.

[12] J. Goossens, E. Grolleau, and L. Cucu-Grosjean, "Periodicity of real-time schedules for dependent periodic tasks on identical multiprocessor platforms," *Real-Time Syst.*, vol. 52, no. 6, pp. 808–832, 2016.

[13] Q. He, M. Lv, and N. Guan, "Response time bounds for DAG tasks with arbitrary intra-task priority assignment," in *ECRTS'21*, vol. 196, 2021, pp. 8:1–8:21.

[14] X. Jiang, J. Sun, Y. Tang, and N. Guan, "Utilization-tensity bound for real-time DAG tasks under global EDF scheduling," *IEEE Trans. Comput.*, vol. 69, no. 1, pp. 39–50, 2020.

[15] S. Kato, S. Tokunaga, Y. Maruyama, S. Maeda, M. Hirabayashi, Y. Kitsukawa, A. Monrroy, T. Ando, Y. Fujii, and T. Azumi, "Autoware on board: enabling autonomous vehicles with embedded systems," in *ICCPS'18*, 2018, pp. 287–296.

[16] S. Kramer, D. Ziegenbein, and A. Hamann, "Real world automotive benchmarks for free," in *WATERS'15*, 2015.

[17] J. Li, K. Agrawal, C. Lu, and C. Gill, "Analysis of global EDF for parallel tasks," in *ECRTS'13*, 2013, pp. 3–13.

[18] C. Liu and J. H. Anderson, "Supporting soft real-time DAG-based systems on multiprocessors with no utilization loss," in *RTSS'10*, 2010, pp. 3–13.

[19] A. Melani, M. Bertogna, V. Bonifaci, A. Marchetti-Spaccamela, and G. C. Buttazzo, "Response-time analysis of conditional DAG tasks in multiprocessor systems," in *ECRTS'15*, 2015, pp. 211–221.

[20] M. Nasri, G. Nelissen, and B. B. Brandenburg, "Response-time analysis of limited-preemptive parallel DAG tasks under global scheduling," in *ECRTS'19*, 2019, pp. 21:1–21:23.

[21] V. Nélis, P. Yomsi, and J. Goossens, "Feasibility intervals for homogeneous multicores, asynchronous periodic tasks, and FJP schedulers," in *RTNS'13*, 2013, pp. 277–286.

[22] A. Parri, A. Biondi, and M. Marinoni, "Response time analysis for G-EDF and G-DM scheduling of sporadic DAG-tasks with arbitrary deadline," in *RTNS'15*, 2015, pp. 205–214.

[23] M. Qamhieh, F. Fauberteau, L. George, and S. Midonnet, "Global EDF scheduling of directed acyclic graphs on multiprocessor systems," in *RTNS'13*, 2013, pp. 287–296.

[24] M. Qamhieh, L. George, and S. Midonnet, "A stretching algorithm for parallel real-time DAG tasks on multiprocessor systems," in *RTNS'14*, 2014, p. 13.

[25] A. Saifullah, D. Ferry, J. Li, K. Agrawal, C. Lu, and C. D. Gill, "Parallel real-time scheduling of DAGs," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 12, pp. 3242–3252, 2014.

[26] J. Sun, N. Guan, J. Sun, X. Zhang, Y. Chi, and F. Li, "Algorithms for computing the WCRT bound of OpenMP task systems with conditional branches," *IEEE Trans. Comput.*, vol. 70, no. 1, pp. 57–71, 2021.

[27] J. Sun, F. Li, N. Guan, W. Zhu, M. Xiang, Z. Guo, and W. Yi, "On computing exact WCRT for DAG tasks," in *DAC'20*, 2020, pp. 1–6.

[28] S. Voronov, S. Tang, T. Amert, and J. H. Anderson, "AI meets real-time: Addressing real-world complexities in graph response-time analysis," in *RTSS'21*, 2021, pp. 82–96.

[29] P. Voudouris, P. Stenström, and R. Pathan, "Timing-anomaly free dynamic scheduling of task-based parallel applications," in *RTAS'17*, 2017, pp. 365–376.

[30] K. Wang, X. Jiang, N. Guan, D. Liu, W. Liu, and Q. Deng, "Real-time scheduling of DAG tasks with arbitrary deadlines," *ACM Trans. Design Autom. Electr. Syst.*, vol. 24, no. 6, pp. 66:1–66:22, 2019.

[31] K. Yang, G. A. Elliott, and J. Anderson, "Analysis for supporting real-time computer vision workloads using OpenVX on multicore+GPU platforms," in *RTNS'15*, 2015, pp. 77–86.

[32] K. Yang, M. Yang, and J. H. Anderson, "Reducing response-time bounds for DAG-based task systems on heterogeneous multicore platforms," in *RTNS'16*, 2016, pp. 349–358.

[33] M. Yang, T. Amert, K. Yang, N. Otterness, J. H. Anderson, F. D. Smith, and S. Wang, "Making OpenVX really "real time"," in *RTSS'18*, 2018, pp. 80–93.

[34] S. Zhao, X. Dai, I. Bate, A. Burns, and W. Chang, "DAG scheduling and analysis on multiprocessor systems: Exploitation of parallelism and dependency," in *RTSS'20*, 2020, pp. 128–140.