Operations Research Letters 51 (2023) 709–716

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

Communication-aware scheduling of precedence-constrained tasks on related machines

Yu Su, Shai Vardi *, Xiaoqi Ren, Adam Wierman

ARTICLE INFO

Article history:
Received 17 August 2022
Received in revised form 31 October 2023
Accepted 2 November 2023
Available online 13 November 2023

Keywords:
Scheduling on related machines
Scheduling with communication
Precedence-constrained scheduling

ABSTRACT

Scheduling precedence-constrained tasks is a classical problem that has been studied for more than fifty years. However, little progress has been made in the setting where there are non-uniform communication delays between tasks. In this work, we propose a new scheduler, Generalized Earliest Time First (GETF), and provide the first provable, worst-case approximation guarantees for the goals of minimizing both the makespan and total weighted completion time of tasks with precedence constraints on related machines with machine-dependent communication speeds.

1. Introduction

In this paper we study scheduling precedence-constrained tasks onto a set of related machines with non-uniform (machine dependent) communication delays between the machines in order to minimize the makespan or the total weighted completion time. This problem is timely due to the prominence of large-scale, general-purpose machine learning platforms. For example, in systems such as Google's TensorFlow [1] and Microsoft's Azure Machine Learning (AzureML) [4], machine learning workflows are expressed via a directed acyclic graph (DAG), where jobs are made up of tasks, represented as vertices, and precedence relationships between the tasks, represented as edges. This abstraction allows data scientists to quickly develop and incorporate modular components into their machine learning pipeline (e.g., data preprocessing, model training, and model evaluation) and then easily specify a workflow. The graphs that specify the workflows in platforms such as TensorFlow and AzureML can be made up of hundreds or even thousands of tasks, and the jobs may be run on systems with thousands of machines. As a result, the performance of the platforms depends on how these precedence-constrained tasks are scheduled across machines.

The study of scheduling jobs composed of precedence-constrained tasks was initiated by [9], who studied scheduling a single job with n precedence-constrained tasks on m identical parallel machines with the goal of minimizing the *makespan*: the time until the last task completes. More generally, the goal of minimizing the *total weighted completion time* is considered, where the total weighted completion time is a weighted average of the completion time of each task in the job (these problems are denoted by $P|\text{prec}|C_{\max}$ and $P|\text{prec}|\sum_j w_j C_j$ respectively in 3-field notation[1]). Note that makespan is a special case of total weighted completion time as a dummy task with weight one can be added as the final task of the job, with all other tasks given weight zero. For $P|\text{prec}|C_{\max}$, Graham showed that a simple list scheduling algorithm can find a schedule of length within a multiplicative factor of $(2-1/m)$ of the optimal. This result is still the best guarantee known for this setting.

Researchers have generalized this setting in several directions, in particular: (i) to heterogeneous machines and (ii) to accommodate inter-task communication. The majority of progress has been made on generalizations to heterogeneous machines. The focus has been on *related machines*, where each machine i has a speed s_i, each task j has a size $p(j)$, and the time to run task j on machine i is $p(j)/s_i$. A sequence of results in the 1980s and 1990s culminated in a result that showed how to use list scheduling algorithms in combination with a partitioning of machines into groups according to their speeds in order to achieve an $O(\log m)$-approximation algorithm for $Q|\text{prec}|C_{\max}$ [5]. This result was also extended in the same work to total weighted completion time by proposing a time-indexed linear programming technique, giving an $O(\log m)$-approximation for total weighted completion time. The idea of using a *group assignment* rule to partition machines into groups of machines with similar speeds and then to assign tasks to

* Corresponding author.
E-mail address: svardi@purdue.edu (S. Vardi).

https://doi.org/10.1016/j.orl.2023.11.001
0167-6377/

[1] Introduced by [8]: the first field denotes the machine environment: P for identical machines, Q for related machines; the second field denotes the job characteristics: prec for precedence constraints, Ψ for identical communication delays, $c_{i,j}$ machine-dependent delays; the third field denotes the objective: $C_{\max}$ for minimizing makespan, $\sum_j w_j C_j$ for minimizing total weighted sum of completion times.

a group has shown up frequently in the years since; it recently led to a breakthrough when this idea was combined with a variation of list scheduling to obtain a $O(\log m/\log\log m)$-approximation algorithm for both makespan and total weighted completion time [12].

There has been little progress toward the goal of incorporating communication delays. Researchers of [10] studied minimizing weighted completion time on *identical machines* with machine-dependent communication speeds.[2] They showed that algorithms for communication-free problems (ones without inter-task communication delays) cannot be adapted in a straightforward manner to this setting. Specifically, if one uses Graham's list scheduling algorithm on the communication-free version of a problem and adds the communication delays incurred by the schedule, the overhead can be (asymptotically) the sum of *all* of the communication delays. They showed that a greedy algorithm called Earliest Time First (ETF) produces schedules with a makespan bounded by $(2-1/m)OPT^{(CF)}+\Psi$, where $OPT^{(CF)}$ is the optimal schedule length for the communication-free problem and Ψ is the amount of communication delays caused by the longest chain generated by the ETF algorithm (which is in turn upper bound by the maximal communication time required by any chain in the precedence graph). In other words, the overhead from the communication-free problem can be reduced from the sum of all communication delays to the cost of a single chain in the precedence-constraint DAG.

The analysis of [10] has proven difficult to generalize to the related machines setting and there has been no progress on scheduling with machine-dependent communication delays outside the context of identical machines. Recently, there has been a surge of progress on scheduling with machine-*independent* delays [6,7,13]. However, extensions of those results to machine-dependent communication delays have proven difficult and the state-of-the-art result in the case of machine-dependent communication delays is still due to [10].

Contributions. In this paper we propose a new scheduler for scheduling precedence-constrained tasks on related machine with machine-dependent communication speeds to minimize the makespan. We show that the scheduler, Generalized Earliest Time First (GETF), computes a schedule S whose makespan is at most $O(\log m/\log\log m)OPT^{(CF)}+\Psi$, where $OPT^{(CF)}$ is the optimal schedule length for the communication-free problem and Ψ is the maximal communication delay that can be induced by a single chain in the precedence graph. We then generalize our result to the objective of minimizing the total weighted completion time and show that GETF produces a schedule $\mathcal{S}$ whose total weighted completion time is at most $O(\log m/\log\log m)\ OPT^{(CF)}+\sum_j \omega_j\Psi_j^S$, where $OPT^{(CF)}$ is the optimal total weighted completion time, ω_j is the weight of task v_j in the objective, and Ψ_j^S is the maximal communication delay caused by a single chain that terminates with job j. The makespan result matches state-of-the-art bounds for the special case when there are no communication delays ([12]). In addition, our proof technique yields a short and cleaner proof for the result of [10] for identical machines. In fact, we slightly improve upon the bound of [10], as the additive component of our bound is with respect to the average communication delay for each task in the chain, instead of the worst case delay. In the case of total weighted completion time, no previous result exists for the case of identical machines with machine-dependent communication times, but the result matches the bound of [12] for $Q|\text{prec}|\sum_j \omega_j C_j$ – the case with related machines and no communication delays.

The key technical advance that enables our new result is a new *Separation Principle*, which allows us to separate the communication delay analysis from the analysis of the communication-free problem in the case of related machines. To prove the Separation Principle, we show that we can choose a chain in the DAG in such a way that the makespan is at most the sum of the solution to the communication-free problem and the communication delays on that particular chain.

We show that we can apply the Separation Principle to both makespan minimization and total weighted completion time by adapting the group assignment rules of [12]. In addition, we use the same proof technique to give a novel (and shorter) proof of the approximation ratio for ETF in the case of identical machines.

Related literature.

The best positive result for $P|\text{prec}|\sum_j w_j C_j$ is currently a $(2+2\ln 2+\epsilon)$-approximation by [12] via a time-indexed linear programming relaxation technique. The work of [14] proved that it is NP-hard to achieve an approximation factor less than 2, given the assumption of a new variant of the Unique Game Conjecture introduced by [3]. The negative results for $P|\text{prec}|C_{max}$ carry over to $P|\text{prec}|\sum_j \omega_j C_j$ as makespan is a special case of total weighted completion time. For related machines, authors of [5] proposed a Speed-based List Scheduling algorithm that obtains an approximation of $O(\log m)$ for $Q|\text{prec}|C_{max}$, and a time-indexed linear programming technique that gives a $O(\log m)$ approximation for $Q|\text{prec}|\sum_j \omega_j C_j$. Recently, an improvement to $O(\log m/\log\log m)$ for both objectives was proven in [12].

When there are positive communication delays, much less is known. No approximation ratio is known for $P|\text{prec}, c_{i,j}|C_{max}$ (scheduling precedence-constrained tasks with machine-dependent communication delays), and this was noted by [2] as one of the 10 most important open questions in scheduling theory. The only algorithm with a guaranteed worst-case performance bound in this setting is ETF [10]. We note that the setting of [10] is a generalization of $P|\text{prec}, c_{i,j}|C_{max}$, as it concerns machine-dependent *speeds*, and each edge $(v_j, v_{j'})$ in the precedence graph is parameterized by a weight $p_{j,j'}$ which denotes the amount of data that needs to be transferred between the machine that executed v_j and the one that will execute $v_{j'}$. Given that $p_{j,j'}$ units of data are passed from machine i to machine i', and the communication speed between these two machines is $s_{i,i'}$, the communication delay is $\frac{p_{j,j'}}{s_{i,i'}}$. $P|\text{prec}, c_{i,j}|C_{max}$ is recovered by setting $p_{j,j'}=1$ for all tasks $v_j, v_{j'}$.

Recently, there have been several breakthroughs for settings with constant communication delays. A $O(\log c\cdot\log m)$-approximation algorithm was proposed in [6] in the case of identical machines for $P|\text{prec}, c|C_{max}$, where there is a fixed communication delay of c between each pair of machines (but no delay for two tasks scheduled on the same machine). This is the first result that is not linear in the communication delays in the approximation ratio for settings with communication delays. For related machines ($Q|\text{prec}, c|C_{max}$), researchers of [13] proposed a $O(\log m\log c/\log\log c)(OPT+c)$-approximation algorithm, where OPT is the optimal makespan for the problem when duplication is allowed. This translates to an $O(\log^5 n/\log\log n)$-approximation to the makespan. Further, they were able to bound the duplication advantage to compute a no-duplication schedule. The work of [7] improved this result to a $O(\log^3 n)$-approximation in the case of minimizing the makespan, and gave a $O(\log^4 n)$-approximation for minimizing total weighted completion time. However, their results do not apply to machine-dependent communication delays.

[2] We distinguish between communication *speeds* and communication *delays*: a communication delay is the time between the completion of a job on one machine and the start of another task on another machine, while communication speed is how fast one unit of data can be transferred between two machines.

2. Problem formulation

We consider the task of scheduling a job made up of a set $V = [n]$ tasks on a heterogeneous system composed of a set $M = [m]$ of machines with different processing and communication speeds. The precedence constraints on the tasks form a directed acyclic graph (DAG) $G = (V, E)$, in which each node j represents a task and an edge (j, j') represents a precedence constraint. We interchangeably use node or task, as convenient. Precedence constraints are denoted by a partial order $\prec$, where $j \prec j'$ means that task j' can only be scheduled after task j completes. The processing demand of task j is represented by $p(j)$, and the amount of data that needs to be transmitted between task j and task j' is represented by $d(j, j')$.

The system is heterogeneous with respect to both processing and communication speeds. For processing speed, we consider the classical *related machines* model: machine i has speed $s(i)$, and it takes $p(j)/s(i)$ uninterrupted time units for task j to complete on machine i. Without loss of generality, we assume that the speed of the fastest machine is m. For communication speeds, we use a similar notion: machines i and i' have communication speed $s(i, i')$ (possibly $s(i, i) \neq 0$). We denote the time at which task j starts executing by $t(j)$, and the machine to which task j is assigned by $h(j)$. If $j \prec j'$, $h(j) = i$ and $h(j') = i'$, the communication delay between task j and j' is $d(j, j')/s(i, i')$. We note this is essentially identical to the model of [10] (they parameterize pairs of machines by $1/s(i, i')$ as opposed to $s(i, i')$). For simplicity, we consider a setting where the machines are fully connected to each other, so any machine can communicate with any other machine. This is without loss of generality as one can simply set the communication speed between any two disconnected machines to 0. We also assume that the preference constraint DAG is connected. This is also without loss of generality since we can connect all of the tasks in G to a dummy task $n+1$, such that $p(n+1) = 0$ and $d(j, n+1) = 0$ for all j. Hence, our results trivially apply to the case of multiple jobs. Additionally, we assume that each machine can process at most one task at a time and the machines are assumed to be *non-preemptive*, i.e., once a task starts on a machine, the scheduler must wait for the task to complete before assigning any new task to this machine. This is a natural assumption in many settings, as interrupting a task and transferring it to another machine can cause significant processing overhead and communication delays due to data locality, e.g., [11].

We focus on two objective functions: minimizing the *makespan*, denoted C_{max}, the time it takes for the final task to complete, and minimizing the *total weighted completion time* of the job, denoted $\sum_j \omega_j C_j$, where C_j is the completion time of task j and ω_j is the weight of task j. We denote our problem settings by $Q\,|\mathsf{prec}, c^*_{i,j}|C_{\max}$ and $Q\,|\mathsf{prec}, c^*_{i,j}|\sum_j \omega_j C_j$ respectively. For any scheduling problem Π in $Q\,|\mathsf{prec}, c^*_{i,j}|C_{\max}$ or $Q\,|\mathsf{prec}, c^*_{i,j}|\sum_j \omega_j C_j$, its *communication-free* version, denoted $\Pi^{(CF)}$, is identical to Π, except that there are no communication delays in $\Pi^{(CF)}$. For any such Π, the optimal solution for $\Pi^{(CF)}$ is denoted by $OPT^{(CF)}$.

A *chain* in the DAG is a sequence of immediate predecessor-successor pairs, whose first node has no predecessor. A *terminal chain* is a chain whose last node is a leaf node with no successors. Note that, because the DAG is connected, there must exist at least one terminal chain.

3. Generalized Earliest Time First (GETF) scheduling

In this section, we introduce an algorithm, Generalized Earliest Time First (GETF), and describe the group-assignment rules that we will use to provide worst-case guarantees for the goals of minimizing our objective functions. Like ETF, GETF seeks to greedily run tasks that can be started earliest, thereby minimizing the idle time created by the precedence constraints. ETF, which is an algorithm for identical machines, does not take into account the potential difference between the service rates of different machines. To account for this, we use group assignment rules, similarly to [5]. More precisely, GETF groups machines with similar speeds together, and assigns the tasks to the groups. Within each group, GETF uses a greedy allocation rule.

Algorithm 1 Generalized Earliest Time First (GETF).

INPUT: tasks V; machines M; precedence constraints $\prec$; group assignment rule f
OUTPUT: a schedule $\mathcal{S} = (\vec{h}, \vec{t})$

$R \leftarrow \{1, 2, \ldots, n\}$
while $R \neq \emptyset$ **do**
  $\mathcal{A} = \{j : j \in R, \nexists j' \text{ s.t. } j' \in R \text{ and } j' \prec j\}$
  For each $j \in \mathcal{A}, i \in f(j)$:
    $\tau_{j,i}$ = earliest starting time on machine i;
    $m_j = \arg\min_{i \in f(j)}\{\tau_{j,i}\}$; $\tau_j = \min_{i \in f(j)}\{\tau_{j,i}\}$
  $\mathcal{B} = \{j : j \in \arg\min_{j:j \in A} \tau_j\}$
  Arbitrarily choose a task j from $\mathcal{B}$.
  $h(j) = m_j$; $t(j) = \tau_j$
  $R \leftarrow R \setminus \{j\}$
end while

GETF is parameterized by a group assignment function $f : V \to [K]$, where $K = \lceil \log_\gamma m \rceil$, $\gamma = \log m / \log\log m$. The output of GETF is a schedule $\mathcal{S}$, which we represent by a pair $\mathcal{S} = (\vec{h}, \vec{t})$, where $\vec{h}$ and $\vec{t}$ represent the machine assignments and start times of the tasks, respectively. In each iteration, GETF computes $\mathcal{A}$, the set of all of the tasks that are ready to process and are not yet scheduled. For every task j in $\mathcal{A}$, GETF calculates $\tau_{j,i}$, the earliest possible starting time of j, if j was to execute on machine i. GETF then sets $\tau_j = \min_{i \in f(j)} \tau_{j,i}$, $m_j = \arg\min_{i \in f(j)} \tau_{j,i}$. In other words, τ_j is the earliest possible starting time for j if it is constrained to execute on a machine in $f(j)$, and m_j is some machine in $f(j)$ that it can execute on at start time. GETF then computes $\mathcal{B}$, the set of tasks in $\mathcal{A}$ with the earliest starting times, arbitrarily chooses a task j from $\mathcal{B}$, and sets $h(j) = m_j$ and $t(j) = \tau_j$. The pseudocode for GETF is given in Algorithm 1. We describe our group assignment functions in the following subsections.

3.1. A group assignment rule for makespan

The group assignment rule f_{mksp} that we use for the goal of minimizing the makespan is adapted from [12], where it was designed for the setting *without* communication delay. First, all machines with speed less than 1 are discarded (recall that the fastest machine has speed m). The remaining machines are divided into K groups $M_1, M_2, \ldots, M_K$ where $K = \lceil \log_\gamma m \rceil$, $\gamma = \log m / \log\log m$, such that group $M_k, k \in [K-1]$ contains machines with speeds in range $[\gamma^{k-1}, \gamma^k)$ and M_K contains machines with speeds in $[\gamma^{K-1}, \gamma^K]$. Note that $K = O(\log m / \log\log m)$. The group assignment rule f_{mksp} is based on the solution of a linear program (LP), which is a relaxation of the following mixed integer linear program (MILP). We note that the solution of the MILP does not necessarily give a feasible schedule, as it allows more than one job to concurrently execute on the same machine.

$$\min_{x_{i,j}, C_j, T} \quad T$$

$$\sum_{i \in M} x_{i,j} = 1 \qquad \forall j \tag{1a}$$

$$p(j) \sum_{i \in M} \frac{x_{i,j}}{s(i)} \leq C_j \qquad \forall j \tag{1b}$$

$$C_{j'} + p(j) \sum_{i \in M} \frac{x_{i,j}}{s(i)} \leq C_j \quad \forall j' \prec j \tag{1c}$$

$$\sum_{j\in V}\frac{p(j)x_{i,j}}{s(i)} \le T \qquad \forall i \tag{1d}$$

$$C_j \le T \qquad \forall j \tag{1e}$$

$$x_{i,j} \in \{0,1\} \quad \forall i,j \tag{1f}$$

The variable C_j denotes the completion time of task j. The binary variable $x_{i,j}$ denotes whether task j is assigned to machine i. Constraint (1a) ensures that every task is processed on some machine; Constraint (1b) guarantees that the processing time of task j is bounded by its completion time; Constraint (1c) enforces the precedence constraints between predecessor-successor pairs; Constraint (1d) guarantees that the total load assigned to any machine is not be greater than the makespan; Constraint (1e) ensures the makespan is not smaller than the completion time of any task.

We relax the above MILP to an LP by replacing constraint (1f) with $x_{i,j} \ge 0$. Denote this LP by LP (1), and let x^*, C^*, T^* denote the optimal solution of this LP. Note that T^* provides a lower bound on $OPT^{(CF)}$, the optimal makespan for the same problem with zero communication delay. For each $k \in [K]$, let $s(M_k) = \sum_{i\in M_k} s(i)$ denote the total speed of the machines in M_k. Let $x^\dagger_{k,j} = \sum_{i\in M_k} x^*_{i,j}$ be the total fraction of task j assigned to machines in M_k. For any task j, define $\ell(j)$ to be the largest index such that at least half of j is (fractionally) assigned to machines in groups $M_{\ell(j)}, \dots, M_K$: $\ell(j) = \arg\max_\ell \left\{\sum_{k=\ell}^K x^\dagger_{k,j} \ge \frac{1}{2}\right\}$. Task j is then assigned to the group from $M_{\ell(j)}, \dots, M_K$ for which the total speed is maximized, i.e.,

$$f_{\mathsf{mksp}}(j) = \arg\max_{\ell(j)\le k\le K} s(M_k).$$

3.2. A group assignment rule for total weighted completion time

The group assignment rule f_{twct} for the goal of minimizing the total weighted completion time is similar in spirit to f_{mksp}. We first divide machines into groups as in Section 3.1. Without loss of generality, we assume that $\frac{p(j)}{s(i)} \ge 1$ for any task j and machine i. Thus, we can divide the time horizon into the following time-indexed intervals of possible task completion times: $[1,2], (2,4], (4,8], \dots, (2^{Q-1}, 2^Q]$ where $Q = \left\lceil \log\left(\sum_j \frac{p(j)}{\min_i s(i)}\right)\right\rceil$. The following MILP forms the basis for the group assignment:

$$\min_{x_{i,j,q}, C_j} \quad \sum_j \omega_j C_j$$

$$\sum_i \sum_q x_{i,j,q} = 1 \qquad \forall j \tag{2a}$$

$$p(j)\sum_i\sum_q \frac{x_{i,j,q}}{s(i)} \le C_j \qquad \forall j \tag{2b}$$

$$C_{j'} + p(j)\sum_i\sum_q \frac{x_{i,j,q}}{s(i)} \le C_j \qquad \forall j' \prec j \tag{2c}$$

$$\sum_{t=1}^q\sum_i x_{i,j,t} - \sum_{t=1}^q\sum_i x_{i,j',t} \le 0 \qquad \forall q, j' \prec j \tag{2d}$$

$$\sum_q 2^{q-1}\sum_i x_{i,j,q} \le C_j \qquad \forall j \tag{2e}$$

$$\sum_j \frac{p(j)}{s(i)}\sum_{t=1}^q x_{i,j,t} \le 2^q \qquad \forall i,q \tag{2f}$$

$$x_{i,j,q} \in \{0,1\} \quad \forall i,j,q \tag{2g}$$

Here, C_j denotes the completion time of task j and ω_j represents its weight in the objective function. The binary variable $x_{i,j,q}$ denotes whether task j is assigned to machine i and it completes in the q-th interval $(2^{q-1}, 2^q]$. Constraints (2a) – (2c) are analogous to Constraints (1a) – (1c). Constraints (2c) and Constraint (2d) enforce the precedence constraint for every predecessor-successor pair. Constraint (2e) guarantees that the completion time of task j is not smaller than the left boundary of the q-th interval $(2^{q-1}, 2^q]$. The total load assigned to machine i up to q-th interval is $\sum_j \frac{p(j)}{s(i)}\sum_{t=1}^q x_{i,j,t}$, and it should not be greater than the upper bound 2^q as in constraint (2f).

We relax constraint (2g) to form an LP, and denote this LP by LP (2). Let $\tilde{x}, \tilde{C}$ denote the optimal solution for this LP. Note that $\sum_j \omega_j \tilde{C}_j$ provides a lower bound for $OPT^{(CF)}$. To set the group assignment rule f_{twct}, define $\tilde{\ell}(j)$ similarly to $\ell(j)$ but with respect to $\tilde{x}$ instead of x^*: let $x^\ddagger_{k,j} = \sum_{q\in[Q]}\sum_{i\in M_k}\tilde{x}_{i,j,q}$, and set $\tilde{\ell}(j) = \arg\max_\ell \left\{\sum_{k=\ell}^K x^\ddagger_{k,j} \ge \frac{1}{2}\right\}$.

The group assignment rule f_{twct} for the goal of minimizing the total weighted completion time is as follows:

$$f_{\mathsf{twct}}(j) = \arg\max_{\tilde{\ell}(j)\le k\le K} s(M_k).$$

4. Results

For the goal of minimizing the makespan, our main result provides a bound in terms of the total communication delay of a terminal chain in the precedence graph. Specifically, let $\mathbb{C} = \mu_1 \prec \mu_2 \prec \dots \prec \mu_{|\mathbb{C}|}$ be a terminal chain in the DAG and define $\Psi(\mathbb{C}, \vec{h}, f)$ to be the maximal communication delay for $\mathbb{C}$ when the tasks are allocated according to $\vec{h}$ and the group assignment rule is f. Formally,

$$\Psi(\mathbb{C}, \vec{h}, f) := \sum_{j=2}^{|\mathbb{C}|} \frac{d(\mu_{j-1}, \mu_j)}{\bar{s}_f(\mu_{j-1}, \mu_j)}, \tag{3}$$

where $\bar{s}_f(\mu_{j-1}, \mu_j) := \min_{i\in f(\mu_j)} s(h(\mu_{j-1}), i)$ is the slowest speed between $h(\mu_{j-1})$, the machine assigned to μ_{j-1}, and any machine in the group $f(\mu_j)$. With a slight abuse of notation, let $\Psi(\vec{h}, f) := \max \Psi(\mathbb{C}, \vec{h}, f)$, where the maximum is over all (terminal) chains in the precedence DAG.

Theorem 4.1. *For any problem in $Q|\mathsf{prec}, c_{i,j}|C_{\max}$, define $\Psi(\vec{h}, f)$ as above, and denote the schedule produced by GETF with group assignment rule f_{mksp} by $\mathcal{S} = (\vec{h}, \vec{t})$. Then*

$$C_{\max}(\mathcal{S}) \le O(\log m/\log\log m)\, OPT^{(CF)} + \Psi(\vec{h}, f_{\mathsf{mksp}}).$$

The bound of Theorem 4.1 depends on a chain in the computed solution; we show that this result translates to a bound that only depends on the problem parameters. Let $\mathbb{C}$ be a terminal chain in the DAG, as above, and let $s_{\max}$ denote the slowest speed between any two machines. Let $\Psi(\mathbb{C}) := \sum_{j=2}^{|\mathbb{C}|} \frac{d(\mu_{j-1},\mu_j)}{s_{\max}}$, and let $\Psi := \max \Psi(\mathbb{C})$, where the maximum is over all chains in the precedence DAG. As $\Psi(\mathbb{C}) \ge \Psi(\mathbb{C}, \vec{h}, f)$ for any $\mathbb{C}, \vec{h}, f$, we have that $\Psi \ge \Psi(\vec{h}, f_{\mathsf{mksp}})$, giving the following corollary.

Corollary 4.2. *For any problem in $Q|\mathsf{prec}, c_{i,j}|C_{\max}$, let Ψ be defined as above, and denote the schedule produced by GETF with group assignment rule f_{mksp} by $\mathcal{S}$. Then*

$$C_{\max}(\mathcal{S}) \le O(\log m/\log\log m)\, OPT^{(CF)} + \Psi.$$

In the special case of identical machines, the group assignment rule f_{mksp} is redundant, as all machines have the same speed. In this case, we can bound the makespan of GETF with respect to a different parameter. Let $\Psi'(\mathbb{C},\vec{h}) = \frac{1}{m}\sum_{j=2}^{|\mathbb{C}|}\sum_{i=1}^{m}\frac{d(\mu_{j-1},\mu_j)}{s(h(\mu_{j-1}),i)}$, and set $\Psi'(\vec{h}) = \max\Psi'(\mathbb{C},\vec{h})$, where the maximum is over all chains generated by the assignment $\vec{h}$.

Proposition 4.3. *For any problem in $P|\text{prec}, c_{i,j}|C_{\max}$, define Ψ' as above, and denote the schedule produced by GETF by $\mathcal{S} = (\vec{h},\vec{t})$. Then*

$$C_{max}(\mathcal{S}) \le \left(2-\frac{1}{m}\right) OPT^{(CF)} + \Psi'(\vec{h}).$$

We note that this improves upon the bound of [10], as their bound is with respect to the maximal possible communication delay, while ours is with respect to the average communication delay.

For total weighted completion time, assume that the tasks are indexed with respect to their order in the schedule determined by GETF (breaking ties arbitrarily). Let $\mathbb{C}_j = \mu_1 \prec \mu_2 \prec \cdots \prec \mu_{\mathbb{C}_j}$ be a terminal chain that includes task j, truncated at $j = \mu_{\mathbb{C}_j}$ and define $\Psi(\mathbb{C}_j,\vec{h},f,j) := \sum_{\ell=2}^{\mathbb{C}_j}\frac{d(\mu_{\ell-1},\mu_\ell)}{\bar{s}_f(\mu_{\ell-1},\mu_\ell)}$ to be the worst case communication time for the chain $\mathbb{C}_j$ for these choices of $\vec{h}$ and f. As before, let $\Psi(\vec{h},f,j) := \max\Psi(\mathbb{C}_j,\vec{h},f,j)$, where the maximum is over chains ending in task j that are consistent with $\vec{h}$ and f. This definition generalizes the notion of $\Psi(\vec{h},f)$ used in Theorem 4.1 for makespan.

Theorem 4.4. *For any problem in $Q|\text{prec}, c_{i,j}|\sum_j \omega_j C_j$, denote the schedule produced by GETF with group assignment rule f_{twct} by $\mathcal{S} = (\vec{h},\vec{t})$. Then*

$$\sum_j \omega_j c_j(\mathcal{S}) \le O(\log m/\log\log m)\, OPT^{(CF)} + \sum_j \omega_j \Psi(\vec{h}, f_{\text{twct}}, j).$$

Similarly to Corollary 4.2, we obtain the following corollary to Theorem 4.4. Let $s_{\max}$ denote the slowest speed between any two machines, let $\Psi(\mathbb{C}_j,j) := \sum_{\ell=2}^{\mathbb{C}_j}\frac{d(\mu_{\ell-1},\mu_\ell)}{s_{\max}}$, and let $\Psi(j) := \max\Psi(\mathbb{C}_j,j)$, where the maximum is over all chains whose last task is j.

Corollary 4.5. *For any problem in $Q|\text{prec}, c_{i,j}|\sum_j \omega_j C_j$, define $\Psi(j)$ as above, and denote the schedule produced by GETF with group assignment rule f_{twct} by $\mathcal{S}$. Then*

$$\sum_j \omega_j C_j(\mathcal{S}) \le O(\log m/\log\log m)\, OPT^{(CF)} + \sum_j \omega_j \Psi(j).$$

5. Proofs

To prove Theorems 4.1 and 4.4, we first prove a *Separation Principle*, which gives a general upper bound for GETF for any choice of group assignment function. We then use the group assignment rules f_{mksp} and f_{twct} to obtain our bounds. The core idea behind the Separation Principle is the construction of a chain which is used to bound the overall makespan.

Theorem 5.1 *(Separation Principle). For any problem in $Q|\text{prec}, c^*_{i,j}|C_{\max}$ and group assignment function f, GETF produces a schedule $\mathcal{S} = (\vec{h},\vec{t})$ of makespan*

$$C_{max}(\mathcal{S}) \le P(\mathbb{C},\vec{h}) + \sum_{k=1}^{K} D_k(f) + \Psi(\mathbb{C},\vec{h},f),$$

where $P(\mathbb{C},\vec{h}) := \sum_{j=1}^{|\mathbb{C}|}\frac{p(\mu_j)}{s(h(\mu_j))}$, $D_k(f) := \frac{\sum_{j:k\in f(j)}p(j)}{s(M_k)}$, and $\Psi(\mathbb{C},\vec{h},f) = \sum_{j=2}^{|\mathbb{C}|}\frac{d(\mu_{j-1},\mu_j)}{\bar{s}_f(\mu_{j-1},\mu_j)}$, where $\mathbb{C} = \mu_1 \prec \mu_2 \prec \ldots \prec \mu_{|\mathbb{C}|}$ is a terminal chain in $\mathcal{S}$.

Proof. For the purposes of the proof, we add two dummy tasks, denoted 0 and $n+1$, which must execute before and after all other tasks respectively. That is, $p(0) = 0$, $p(n+1) = 0$, $d(0,j) = 0$ and $d(j,n+1) = 0$ for all $j \in [n]$. The proof now proceeds in four steps:

(*i*) *Construct a terminal chain.* We inductively construct a terminal chain $\mathbb{C}$ from its end to its start. Add task $n+1$ to the end of the chain, and denote it $\mu_{|\mathbb{C}|}$. From the immediate predecessors of task $\mu_{|\mathbb{C}|}$, pick one of the tasks that finishes last and denote it by $\mu_{|\mathbb{C}|-1}$. Continue inductively to construct the terminal chain $\mu_1 \prec \mu_2 \prec \ldots \prec \mu_{|\mathbb{C}|}$, where μ_1 is the task 0.

(*ii*) *Partition $[0, C_{max}]$ into $K+1$ intervals.* Recall that K is the number of groups of machines in the group assignment rule of Section 3.1. Let $\mathcal{T}_0^{\mathbb{C}}$ denote the union of the (disjoint) time intervals during which the tasks of chain $\mathbb{C}$ are being processed. Define $\mathcal{T}_1^{\mathbb{C}}, \mathcal{T}_2^{\mathbb{C}}, \ldots, \mathcal{T}_K^{\mathbb{C}}$ as follows: for each task μ_j, $j \in \{2,\ldots,|\mathbb{C}|\}$, set $M_k = f(\mu_j)$, and assign the time interval between the end of task μ_{j-1} and the start of task μ_j to $\mathcal{T}_k^{\mathbb{C}}$. That is, $\mathcal{T}_k^{\mathbb{C}}$ is the set of time intervals that tasks in the terminal chain $\mathbb{C}$ assigned to machines in group M_k have to wait before being processed. The sum of the lengths of the time intervals in $\bigcup_k \mathcal{T}_k^{\mathbb{C}}$ is exactly the makespan. Note that $\mathcal{T}_k^{\mathbb{C}}, k = \{1,\ldots,K\}$ can be empty or contain more than one time interval.

(*iii*) *Bound the idle time in between tasks in the chain.* Consider a task μ_j assigned to machine $h(\mu_j)$. For each machine $i \in f(\mu_j)$, let $E(\mu_{j-1},\mu_j,i)$ denote the union of disjoint idle time intervals on machine i between the end time of task μ_{j-1} and the start time of task μ_j in the schedule $\mathcal{S}$. Due to the greedy nature of GETF and the fact that the predecessor of each task in the terminal chain is the one that finished last, we know that no task could have started earlier on any machine in its assigned group, therefore the length of $E(\mu_{j-1},\mu_j,i)$ is bounded above by the communication delays between task μ_{j-1} and task μ_j, i.e., for any $i \in f(\mu_j)$,

$$|E(\mu_{j-1},\mu_j,i)| \le \frac{d(\mu_{j-1},\mu_j)}{s(h(\mu_{j-1}),i)},$$

otherwise task μ_j could have started earlier on machine i. Let $\bar{e}_k$ be the maximal time amount of time that any machine in group M_k is idle in the intervals of $\mathcal{T}_k$, i.e., during the time that tasks in $\mathbb{C}$ waited to be processed on some machine in M_k. Then

$$\sum_{k=1}^{K}\bar{e}_k \le \sum_{j=2}^{|\mathbb{C}|}\frac{d(\mu_{j-1},\mu_j)}{\min_{i\in f(\mu_j)} s(h(\mu_{j-1}),i)} = \sum_{j=2}^{|\mathbb{C}|}\frac{d(\mu_{j-1},\mu_j)}{\bar{s}_f(\mu_{j-1},\mu_j)}. \quad (4)$$

Note that the right hand side of (4) is exactly the definition of $\Psi(\mathbb{C},\vec{h},f)$ (3).

(*iv*) *Bound the makespan.* Denote the total length of the intervals in $\mathcal{T}_k$ by t_k. Let $e_k(i)$ denote the total idle time of machine i in the intervals of $\mathcal{T}_k$. By the definition of $e_k(i)$, there must be exactly $(t_k - e_k(i))\,s(i)$ units processed on machine i during $\mathcal{T}_k$. As the total number of units processed on machines in M_k in $\mathcal{S}$ is $\sum_{j:f(j)=M_k} p(j)$, we have that for $1 \le k \le K$,

$$\sum_{i\in M_k}(t_k - e_k(i))\,s(i) \le \sum_{j:f(j)=M_k} p(j). \quad (5)$$

Recall that $s(M_k) = \sum_{i\in M_k} s(i)$. Rearranging (5), we get

$$t_k \le \frac{\sum_{j:f(j)=M_k} p(j)}{s(M_k))} + \frac{\sum_{i\in M_k} e_k(i)s(i)}{s(M_k)}. \quad (6)$$

We now bound the makespan:

$$\begin{aligned}
C_{\max}(\mathcal{S}) &= \sum_{k=1}^{K} t_k + t_0 \\
&\le \sum_{k=1}^{K} \left(\frac{\sum_{j:f(j)=k} p(j)}{s(M_k))} + \frac{\sum_{i\in M_k} e_k(i)s(i)}{s(M_k)} \right) + \sum_{\mu_j \in \mathbb{C}} \frac{p(\mu_j)}{s(h(\mu_j))} \\
&\le P(\mathbb{C}, \vec{h}) + \sum_{k=1}^{K} D_k(f) + \sum_{k=1}^{K} \bar{e}_k \frac{\sum_{i\in M_k} s(i)}{s(M_k)} \\
&\le P(\mathbb{C}, \vec{h}) + \sum_{k=1}^{K} D_k(f) + \Psi(\mathbb{C}, \vec{h}, f),
\end{aligned}$$

where the first inequality is from (6) and the last inequality is due to (4). □

5.1. Proof of Theorem 4.1

In order to apply the Separation Principle to prove Theorem 4.1, we prove bounds on $P(\mathbb{C}, \vec{h})$ and $\sum_{k=1}^{K} D_k(f_{\text{mksp}})$ in the case of the group assignment rule defined in Section 3.1. The bounds are given in the following two lemmas, which are adapted from results in [12]. Theorem 4.1 follows directly from these lemmas, the Separation Principle, and the fact that $T^* \le OPT^{(CF)}$, where T^* is the optimal solution to LP (1).

Lemma 5.2. *Let x^*, C^*, T^* denote the optimal solution to LP (1). Then $P(\mathbb{C}, \vec{h}) \le 2\gamma T^*$ for any $\vec{h}$ consistent with f_{mksp}.*

Proof. For any $j \in \mathbb{C}$, $\sum_{k=1}^{\ell(j)} x^{\dagger}_{k,j} \ge \frac{1}{2}$, by the definition of $\ell(j)$. Thus,

$$\sum_{i\in M} \frac{x^*_{i,j}}{s(i)} = \sum_{k=1}^{K} \sum_{i\in M_k} \frac{x^*_{i,j}}{s(i)} \ge \sum_{k=1}^{\ell(j)} \frac{x^{\dagger}_{k,j}}{\gamma^{\ell(j)}} \ge \frac{1}{2\gamma^{\ell(j)}} \ge \frac{1}{2\gamma s(h(\mu_j))}, \tag{7}$$

where the first inequality is due to the fact that speed of any machine in groups $M_1, \ldots, M_{\ell(j)}$ is at most $\gamma^{\ell(j)}$ and the rightmost inequality is due to the fact that f_{mksp} allocates task j to some group M_k where $k \ge \ell(j)$, hence its speed is at least $\gamma^{\ell(j)-1}$. We can now bound $P(\mathbb{C}, \vec{h})$ as follows:

$$\begin{aligned}
P(\mathbb{C}, \vec{h}) &= \sum_{j=1}^{|\mathbb{C}|} \frac{p(\mu_j)}{s(h(\mu_j))} \le 2\gamma \sum_{j=1}^{|\mathbb{C}|} p(\mu_j) \sum_{i\in M} \frac{x^*_{i,\mu_j}}{s(i)} \\
&\le 2\gamma \left(C^*_{\mu_1} + \sum_{j=2}^{|\mathbb{C}|} p(\mu_j) \sum_{i\in M} \frac{x^*_{i,\mu_j}}{s(i)} \right) \qquad (8a) \\
&\le 2\gamma C^*_{\mu_{|\mathbb{C}|}} \le 2\gamma T^*, \qquad (8b)
\end{aligned}$$

where the first inequality is due to (7), the second is due to Constraint (1b), the third is by repeated application of Constraint (1c), and the last inequality is due to Constraint (1e). □

Lemma 5.3. *Let x^*, C^*, T^* be a feasible solution to LP (1). Then $\sum_{k=1}^{K} D_k(f_{\text{mksp}}) \le 2KT^*$.*

Proof. Define $x^{\dagger}_{k,j}$ and $\ell(j)$ with respect to x^*, similarly to Section 3.1. We first bound $s(f_{\text{mksp}}(j))$, where $s(f(j))$ is $s(M_k)$ with $k = f(j)$:

$$\frac{1}{2s(f_{\text{mksp}}(j))} \le \sum_{k=\ell(j)}^{K} \frac{x^{\dagger}_{k,j}}{s(f_{\text{mksp}}(j))} \le \sum_{k=\ell(j)}^{K} \frac{x^{\dagger}_{k,j}}{s(M_k)} \le \sum_{k=1}^{K} \frac{x^{\dagger}_{k,j}}{s(M_k)},$$

where the first inequality is because $\sum_{k=\ell(j)}^{K} x^{\dagger}_{k,j} \ge \frac{1}{2}$, and the second is because of the definition of f_{mksp}. Therefore

$$\begin{aligned}
\sum_{k=1}^{K} D_k(f_{\text{mksp}}) &= \sum_{k=1}^{K} \frac{\sum_{j:f_{\text{mksp}}(j)=M_k} p(j)}{s(M_k)} = \sum_{j\in V} \frac{p(j)}{s(f_{\text{mksp}}(j))} \\
&\le 2 \sum_{j\in V} \sum_{k=1}^{K} \frac{p(j)x^{\dagger}_{k,j}}{s(M_k)} = 2 \sum_{j\in V} \sum_{k=1}^{K} \sum_{i\in M_k} \frac{p(j)x^*_{i,j}}{s(M_k)} \\
&\le 2 \sum_{k=1}^{K} T^* = 2KT^*,
\end{aligned}$$

where the first inequality is from the bound on $s(f_{\text{mksp}}(j))$ and the second is due to Constraint (1d), noting that the constraint can be rewritten as follows: $\sum_{j\in V} p(j)x_{i,j} \le Ts(i)$. □

5.2. Proof of Theorem 4.4

Given a schedule $\mathcal{S} = (\vec{h}, \vec{t})$ for a DAG G, once again assume that the tasks are indexed with respect to their order in the schedule determined by GETF and let $G(\mathcal{S}, j)$ denote the subgraph of G induced by the set of the tasks that have been scheduled up to task j (i.e., the nodes $1, \ldots, j$ and edges between these nodes). Recall that $\mathbb{C}_j = \mu_1 \prec \mu_2 \prec \cdots \prec \mu_{\mathbb{C}_j}$ is a terminal chain that includes task j, truncated at $j = \mu_{\mathbb{C}_j}$, and define $P(\mathbb{C}_j, \vec{h})$ similarly to $P(\mathbb{C}, \vec{h})$, but with respect to $\mathbb{C}_j$: $P(\mathbb{C}_j, \vec{h}) := \sum_{\mu_j\in\mathbb{C}_j} \frac{p(\mu_j)}{s(h(\mu_j))}$. Let $D_k(f_{\text{twct}}, \mathcal{S}, j)$ denote the load of machines in group M_k in $\mathcal{S}$ resulting from tasks $1, \ldots, j$ (note that we need $\mathcal{S}$ in the notation as it defines the task order):

$$D_k(f_{\text{twct}}, \mathcal{S}, j) := \frac{\sum_{\mu_j\in[j], f_{\text{twct}}(\mu_j)=k} p(\mu_j)}{s(M_k)}. \tag{9}$$

We apply the Separation Principle for each task j (in the order defined by $\mathcal{S}$) and combine these inequalities as follows:

$$\begin{aligned}
\sum_j \omega_j C_j \le \sum_j \omega_j \left(P(\mathbb{C}_j, \vec{h}) + \sum_k D_k(f_{\text{twct}}, \mathcal{S}, j) \right) \\
+ \sum_j \omega_j \Psi(\vec{h}, f_{\text{twct}}, j).
\end{aligned}$$

As both $P(\mathbb{C}_j, \vec{h})$ and $D_k(f_{\text{twct}}, \mathcal{S}, j)$ are independent of the communication constraints, we can use the group assignment rule f_{twct} to tighten the bound. Divide the time horizon into Q sets, as defined in Subsection 3.2. For any task j, define $q(j)$ to be the minimum value of q such that both $\sum_{t=1}^{q} \sum_i \tilde{x}_{i,j,t} \ge \frac{1}{2}$ and $\tilde{C}_j \le 2^q$ are satisfied. Intuitively, $q(j)$ can be viewed as a rough estimate of the completion time of task j. Such a $q(j)$ must exist as both inequalities trivially hold for $q = Q$, as 2^Q is an upper bound on the completion time (excluding communication times) of any task. For every $q \in [Q]$, define $\mathcal{J}_q = \{j : q(j) = q\}$. We require the following result, which shows that an optimal solution to LP (2) is a feasible solution to LP (1) when it is solved for the jobs in $\mathcal{J}_q$.

Lemma 5.4. *Let $\tilde{x}, \tilde{C}$ denote the optimal solution to LP (2). For every task j, set $\alpha_j = \sum_{t=1}^{q(j)} \sum_i \tilde{x}_{i,j,t}$. Then for any $q \in [Q]$,*

$$x^*_{i,j} = \sum_{t=1}^{q} \frac{\tilde{x}_{i,j,t}}{\alpha_j} \qquad \forall j \in \mathcal{J}_q, i$$
$$C^*_j = 2\tilde{C}_j \qquad \forall j \in \mathcal{J}_q$$
$$T^* = 2^{q+1}.$$

is a feasible solution for LP (1) when it is solved for the tasks in $\mathcal{J}_q$. Furthermore, $\sum_{j\in\mathcal{J}_q} \frac{p(j)}{s(f_{\text{twct}}(j))} \le 2^{q+2}K$.

Proof. It suffices to verify that x^*, C^* and T^* satisfy all the constraints in LP (1). By the definition of α_j, it is clear that Constraint (1a) holds for any task $j \in \mathcal{J}_q$. To validate that constraint (1b) is satisfied, note that $\alpha_j \ge 1/2$ by definition and so a direct substitution on the left hand side yields the right hand side due to (2b). Similarly, constraint (2c) ensures that constraint (1c) is satisfied and constraint (2f) ensures that constraint (1d) is satisfied. Finally, $C^*_j \le 2^q$ by the definition of $q(j)$ and thus constraint (1e) holds. The proof that $\sum_{j\in\mathcal{J}_q} \frac{p(j)}{s(f_{\text{twct}}(j))} \le 2^{q+2}K$ is essentially identical to the proof of Lemma 5.3, with f_{twct}, $\tilde{\ell}(j)$ and $x^{\ddagger}_{k,j}$ instead of f_{mksp}, $\ell(j)$ and $x^{\dagger}_{k,j}$ and the equality in the second line of the second equation block is replaced with an inequality. □

Using Lemma 5.4, we prove the following.

Lemma 5.5. $P(\mathbb{C}_j, \vec{h}) + \sum_k D_k(f_{\text{twct}}, \mathcal{S}, j) \le 32(\gamma + K) \cdot \tilde{C}_j$

Proof. Let $\tilde{x}, \tilde{C}$ denote the optimal solution to LP (2). Similarly to the proof of Lemma 5.2,

$$\sum_{q\in[Q]} \sum_{i\in M} \frac{\tilde{x}_{i,j,q}}{s(i)} \ge \sum_{k=1}^{\tilde{\ell}(j)} \frac{x^{\ddagger}_{k,j}}{\gamma^{\tilde{\ell}(j)}} \ge \frac{1}{2\gamma^{\tilde{\ell}(j)}} \ge \frac{1}{2\gamma s(h(\mu_j))}. \tag{10}$$

Let x^*, C^*, T^* be as in Lemma 5.4, and let $\mathbb{C}$ be any chain in the DAG induced by the nodes of $\mathcal{J}_q$. Similarly to Lemma 5.2,

$$P(\mathbb{C}, \vec{h}) = \sum_{\mu_j\in\mathbb{C}} \frac{p(\mu_j)}{s(h(\mu_j))} \le 2\gamma \sum_{\mu_j\in\mathbb{C}} p(\mu_j) \sum_{q\in[Q]} \sum_{i\in M} \frac{\tilde{x}_{i,\mu_j,q}}{s(i)}$$
$$\le 2\gamma \sum_{\mu_j\in\mathbb{C}} p(\mu_j) \sum_{i\in M} \frac{x^*_{i,\mu_j}}{s(i)} \le 2\gamma \cdot 2^{q+1},$$

where the first inequality is due to (10) and the second is due to the definition of x^*. The third inequality is due to reasoning identical to that of lines (8a) and (8b) in the proof of Lemma 5.2, using the fact that x^*, C^*, T^* is a feasible solution to LP (1).

We can represent the chain $\mathbb{C}_j$ as a concatenation of chains in the DAGs induced by tasks in $\mathcal{J}_1, \ldots, \mathcal{J}_{q(j)}$. Thus,

$$P(\mathbb{C}_j, \vec{h}) \le \sum_{t=1}^{q(j)} 2\gamma \cdot 2^{t+1} \le 8\gamma \cdot 2^{q(j)}. \tag{11}$$

We bound $\sum_k D_k(f_{\text{twct}}, \mathcal{S}, j)$ as follows:

$$\sum_k D_k(f_{\text{twct}}, \mathcal{S}, j) \le \sum_{t=1}^{q(j)} \sum_{\mu_j\in\mathcal{J}_t} \frac{p(\mu_j)}{s(f_{\text{twct}}(\mu_j))}$$
$$\le \sum_{t=1}^{q(j)} 2K \cdot 2^{t+1} \le 8K \cdot 2^{q(j)}, \tag{12}$$

where the first inequality is from the definition of $D_k(f_{\text{twct}}, \mathcal{S}, j)$ in (9) and because the tasks in $G(\mathcal{S}, j)$ form a subset of $\cup_{t=1}^{q(j)} \mathcal{J}_q$, and the second inequality is from Lemma 5.4.

By the definition of $q(j)$, for task j either *(a)* $\sum_{t=1}^{q(j)-1} \sum_i \tilde{x}_{i,j,t} < \frac{1}{2}$ or *(b)* $\tilde{C}_j > 2^{q(j)-1}$. If *(a)* holds, then

$$2^{q(j)-1} \le 2^{q(j)-1} \cdot 2 \left(\sum_{t=q(j)}^{Q} \sum_i \tilde{x}_{i,j,t} \right)$$
$$\le 2 \left(\sum_{t=q(j)}^{Q} 2^{t-1} \sum_i \tilde{x}_{i,j,t} \right) \le 2\tilde{C}_j,$$

where the last inequality is due to constraint (2e) in the LP (2). If *(b)* holds then $2^{q(j)-1} < C^*_j = 2\tilde{C}_j$. In both cases, $2^{q(j)-1}$ is upper bounded by $2\tilde{C}_j$, and hence $2^{q(j)}$ is upper bounded by $4\tilde{C}_j$. Combining this with Inequalities (11) and (12) completes the proof. □

Combining Lemma 5.5 with the fact that $\sum_j \omega_j \tilde{C}_j$ is lower bounded by $OPT^{(CF)}$ completes the proof of Theorem 4.4:

$$\sum_j \omega_j C_j(\mathcal{S}) \le 32(\gamma + K) \sum_j \omega_j \tilde{C}_j + \sum_j \omega_j \Psi(\vec{h}, f_{\text{twct}}, j)$$
$$\le O(\log m / \log\log m) \cdot OPT^{(CF)} + \sum_j \omega_j \Psi(\vec{h}, f_{\text{twct}}, j).$$

5.3. Proof of Proposition 4.3

The proof technique of the Separation Principle can be used to provide a new, simpler proof of the approximation ratio of ETF in the case of identical machines. Without loss of generality, we assume that the machines all have speed 1. The proof can be broken into three steps.

(*i*) *Choose one of the longest terminal chains* $\mathbb{C}$.

(*ii*) *Bound the idle time in between tasks.* Let $I(\mu_{j-1}, \mu_j)$ be the time interval between the end time of task μ_{j-1} and the start time of μ_j for $j = 2, 3, \ldots, |\mathbb{C}|$, and for each machine $i \in M$, define $E(\mu_{j-1}, \mu_j, i)$ as a union of disjoint idle time intervals on machine i during the time interval $I(\mu_{j-1}, \mu_j)$. For any machine $i \in M$, $j = 2, 3, \ldots, |\mathbb{C}|$, it holds that $|E(\mu_{j-1}, \mu_j, i)| \le \frac{d(\mu_{j-1}, \mu_j)}{s(h(\mu_{j-1}), i)}$, otherwise task μ_j could have started earlier on machine i.

(*iii*) *Bound the makespan.* During the union of the time intervals in between the tasks of $\mathbb{C}$, $\cup_{j\in\{2,\ldots,|\mathbb{C}|\}} I(\mu_{j-1}, \mu_j)$, exactly $\sum_{j=2}^{|\mathbb{C}|} \sum_{i=1}^{m} (|I(\mu_{j-1}, \mu_{j_i})| - |E(\mu_{j-1}, \mu_j, i)|)$ processing units are executed (this is precisely the time that the machines are not idle), and this is bounded by a sum of the processing units for all the tasks except those in the terminal chain. Therefore

$$\sum_{j=2}^{|\mathbb{C}|} \sum_{i=1}^{m} \left(|I(\mu_{j-1}, \mu_j)| - |E(\mu_{j-1}, \mu_j, i)| \right)$$
$$\le \sum_{j=1}^{n} p(j) - \sum_{j=1}^{|\mathbb{C}|} p(\mu_j). \tag{13}$$

We bound $C_{\max}$ as follows.

$$C_{\max}(\mathcal{S}) = \sum_{j=2}^{|\mathbb{C}|} |I(\mu_{j-1}, \mu_j)| + \sum_{j=1}^{|\mathbb{C}|} p(\mu_j)$$
$$\le \frac{1}{m} \sum_{j=1}^{n} p(j) + \frac{m-1}{m} \sum_{j=1}^{|\mathbb{C}|} p(\mu_j) + \frac{1}{m} \sum_{j=2}^{|\mathbb{C}|} \sum_{i=1}^{m} |E(\mu_{j-1}, \mu_j, i)|$$
$$\le \left(2 - \frac{1}{m}\right) OPT^{(CF)} + \Psi'(\vec{h}),$$

where the first inequality is due to Inequality (13) and the last inequality is due to the following: (i) $\frac{1}{m}\sum_{j=1}^{n} p(j) \leq OPT^{(CF)}$, and (ii) the makespan of any schedule should at least cover the processing time of any chain $\mathbb{C}$ in the DAG, therefore $\sum_{j=1}^{|\mathbb{C}|} p(\mu_j) \leq OPT^{(CF)}$ and (iii) from the definitions of $\Psi'(\vec{h})$ and $E(\mu_{j-1}, \mu_j, i)$.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al., Tensorflow: a system for large-scale machine learning, in: OSDI 16, 2016, pp. 265–283.

[2] Nikhil Bansal, Scheduling: open problems old and new, in: MAPSP, 2017, http://www.mapsp2017.ma.tum.de/MAPSP2017-Bansal.pdf.

[3] Nikhil Bansal, Subhash Khot, Optimal long code test with one free bit, in: 2009 50th Annual IEEE Symposium on Foundations of Computer Science, IEEE, 2009, pp. 453–462.

[4] David Chappell, Introducing Azure Machine Learning. A Guide for Technical Professionals, Sponsored by Microsoft Corporation, 2015.

[5] Fabián A. Chudak, David B. Shmoys, Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds, J. Algorithms 30 (2) (1999) 323–343.

[6] Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, Yihao Zhang, Scheduling with communication delays via lp hierarchies and clustering, in: 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), IEEE, 2020, pp. 822–833.

[7] Sami Davies, Janardhan Kulkarni, Thomas Rothvoss, Jakub Tarnawski, Yihao Zhang, Scheduling with communication delays via lp hierarchies and clustering ii: weighted completion times on related machines, in: Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA), SIAM, 2021, pp. 2958–2977.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, in: P.L. Hammer, E.L. Johnson, B.H. Korte (Eds.), Discrete Optimization II, in: Annals of Discrete Mathematics, vol. 5, Elsevier, 1979, pp. 287–326.

[9] Ronald L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (2) (1969) 416–429.

[10] Jing-Jang Hwang, Yuan-Chieh Chow, Frank D. Anger, Chung-Yee Lee, Scheduling precedence graphs in systems with interprocessor communication times, SIAM J. Comput. 18 (2) (1989) 244–257.

[11] Yu-Kwong Kwok, Ishfaq Ahmad, Static scheduling algorithms for allocating directed task graphs to multiprocessors, ACM Comput. Surv. 31 (4) (1999) 406–471.

[12] S. Li, Scheduling to minimize total weighted completion time via time-indexed linear programming relaxations, in: 2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS), Oct 2017, pp. 283–294.

[13] Biswaroop Maiti, Rajmohan Rajaraman, David Stalfa, Zoya Svitkina, Aravindan Vijayaraghavan, Scheduling precedence-constrained jobs on related machines with communication delay, in: FOCS 2020, IEEE, 2020, pp. 834–845.

[14] Ola Svensson, Conditional hardness of precedence constrained scheduling on identical machines, in: Proceedings of the Forty-Second ACM Symposium on Theory of Computing, ACM, 2010, pp. 745–754.