Communication-Aware Scheduling of Precedence-Constrained Tasks

Yu Su
Caltech
suyu@caltech.edu

Xiaoqi Ren
Google
xiaoqiren@google.com

Shai Vardi
Purdue University
svardi@purdue.edu

Adam Wierman
Caltech
adamw@caltech.edu

Yuxiong He
Microsoft
yuxhe@microsoft.com

ABSTRACT

Jobs in large-scale machine learning platforms are expressed using a computational graph of tasks with precedence constraints. To handle such precedence-constrained tasks that have machine-dependent communication demands in settings with heterogeneous service rates and communication times, we propose a new scheduling framework, Generalized Earliest Time First (GETF), that improves upon state-of-the-art results in the area. Specifically, we provide the first provable, worst-case approximation guarantee for the goal of minimizing the makespan of tasks with precedence constraints on related machines with machine-dependent communication times.

1. INTRODUCTION

Within large-scale machine learning platforms, machine learning workflows are expressed via a computational graph, where jobs are made up of tasks, represented as vertices. Edges in the graph simultaneously indicate a precedence relationship and the communication needed between the tasks. This “precedence graph” 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.

How these precedence-constrained tasks are scheduled across machines is crucial to the performance of machine learning platforms. This scheduling problem is known to be very challenging: both the goal of partitioning the jobs across machines and of scheduling the jobs among a group of machines are NP-hard [7]. Hence, one can only hope to develop approximation algorithms and/or heuristics.

The task of scheduling a job made up of precedence-constrained tasks is classical and has been studied for more than fifty years, starting with the work of Graham [3]. The most classical version of this scheduling problem focuses on scheduling n precedence-constrained tasks on m identical parallel machines to minimize the *makespan*, the time until the last task completes. Graham showed that a simple list scheduling algorithm can find a schedule of length within a factor of two of the optimal, i.e., that list scheduling is a $(2-1/m)$-approximation algorithm. This result is still the best guarantee known for this simple setting. However, this setting is too simplistic in two important ways: (i) machines are not identical in modern platforms and (ii) communication between tasks is a crucial factor in modern platforms.

MAMA 2019 Phoenix, AZ, USA

Addressing these two issues has been the goal of the field since Graham’s initial result fifty years ago. Since that time, considerable progress has been made on generalizations to heterogeneous machines. The focus has been on *(uniformly) related machines*, a model where each machine i has a speed s_i, each task j has a size w_j, and the time to run task j on machine i is w_j/s_i. Under the related machine model, 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 with “similar” speeds in order to achieve an $O(\log m)$-approximation algorithm [1]. A breakthrough happened in 2017 when the idea of partitioning machines was adapted further and combined with a variation of list scheduling to obtain a $O(\log m/\log\log m)$-approximation algorithm [6].

While there has been significant progress in the direction of generalizing from identical machines to heterogeneous machines, this progress has not extended to the goal of incorporating communication costs. The state-of-the-art result for the incorporation of communication costs is from [4], which studies machine-dependent communication costs in the setting of *identical machines*. In this context, a greedy algorithm called Earliest Time First (ETF) has been shown to provide schedules with a makespan bounded by $(2-1/m)\mathrm{OPT}^{(i)}+C$, where $\mathrm{OPT}^{(i)}$ is the optimal schedule length when ignoring communication time and C is the maximum amount of communication of a chain (path) in the precedence graph. However, the analysis for the case of identical machines in [4] has proven difficult to generalize. As a result, outside of the context of identical machines, there has been no progress in the thirty years since [4].

In this paper, we propose a new scheduling framework Generalized Earliest Time First (GETF) and prove that, in the case of related machines and machine-dependent communication times, it computes a makespan that is at most of length $O(\log m/\log\log m)\mathrm{OPT}^{(i)}+C$, where C is the amount of communication time in a chain (path) in the precedence graph. This result has impact both *theoretically* – it addresses a long-standing open problem – and *practically* – these schedulers can handle large, complex precedence graphs in systems with heterogeneous machines and machine-dependent processing times.

2. PROBLEM FORMULATION

We study a model that generalizes $Q|prec, c_{i,j}|C_{max}$ by additionally including machine-dependent communication times. Specifically, we consider the task of scheduling a job made up of a set V of n tasks on a heterogeneous system composed of a set M of m machines with potentially different processing speeds and communication speeds. The tasks form a directed acyclic graph (DAG) $G = (V, E)$, in which each node j represents a task and an edge (j', j) between task j and task j' represents a precedence constraint. We interchangeably use node or task, as convenient. Precedence constraints can be denoted by a partial order $\prec$ between two nodes of any edge. For instance, $j' \prec j$ means that task j can only be scheduled after task j' completes. The processing demand of task j is w_j processing units, and the amount of data to be transmitted between task j' and task j is represented by the edge weight $w_{j',j}$ of (j', j).

The system is heterogeneous in two aspects: processing speed and communication speed. For processing speed, we consider the classical *related machines* model: a machine i has speed s_i, and it takes w_j/s_i uninterrupted time units for task j to complete on machine i. On the other hand, the communication speed $s_{i',i}$ between any two machines i', i is heterogeneous across different machine pairs. Machine-dependent communication speeds are crucial for capturing issues such as data locality and the difference between intra-rack and inter-rack communication. We index the machine to which task j is assigned to by $h(j)$. If $i = h(j)$ and $i' = h(j')$, then communication time between task j' and j in the DAG is $w_{j',j}/s_{i',i}$.

For simplicity, we consider a setting where the machines are fully connected to each other, so any machine can communicate with any other machine. The results can be extended to the case where machines are not fully connected in a straightforward manner. We also assume that the DAG is connected, otherwise it can be viewed as multiple DAGs. Additionally, our model assumes that each machine (processing unit) can process at most one task at a time, i.e., there is no *time-sharing*. Further, 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 appropriate for machine learning platforms because, in practice, interrupting a task and transferring it to another machine can cause significant processing overhead and communication delays due to data locality [5].

The goal of the scheduler in our model is to minimize the *makespan* of the job, denoted by C_{max}, which is the time when the the final task in the DAG completes. Minimizing makespan for jobs with precedence constraints is known to be NP-complete [2], and thus we aim to design a polynomial-time algorithm that computes an approximate optimal schedule. We say that an algorithm is a ρ-approximation algorithm if it always produces a solution with an objective value within a factor of ρ of optimal in polynomial time.

Our main results use two important concepts. First, our results provide bounds in terms of $\text{OPT}^{(i)}$, which is the optimal schedule length obtained when ignoring the communication delays. Note that $\text{OPT}^{(i)}$ is a lower bound of the optimal schedule length of the problem when communication delays are included. Second, we provide bounds in terms of the communication time of a *terminal chain* of the schedule. A *chain* in the DAG is a sequence of immediate predecessor-successor pairs, whose first node is a node with no predecessor and last node is a leaf node with zero successor.

For any given schedule, a terminal chain $\mathcal{C}$ of length N can be constructed in the following fashion. We start with one of the tasks that ends last in the given schedule, denoted as c_N. Among all the immediate predecessors of node c_N, we pick one of the tasks that finishes last and define it as c_{N-1}. In such a way, we construct a chain of tasks $c_1 \prec c_2 \prec \ldots \prec c_N$ until the first node c_1 in the chain does not have any predecessor. There may be many such terminal chains, and our results apply to any arbitrary terminal chain for the given schedule.

3. ALGORITHM DESIGN

In this section, we introduce the algorithmic framework, Generalized Earliest Time First (GETF), which is inspired by the classical ETF algorithm [4] and recent advances in list scheduling for related machines [6].

At its core, GETF is a greedy algorithmic framework. Like ETF, it seeks to always run a task that can be started earliest, thus minimizing the idle time created by the precedence constraints in a greedy way. However, this simple heuristic does not take into account the potential difference between the service rates of different machines. For this, GETF takes inspiration from Speed-based List Scheduling (SLS) [1, 6] and uses a group assignment function $f(\cdot)$ to determine sets of "similar" machines and then assign tasks to different groups of machines. Then, within the groups of similar machines, GETF uses the ETF greedy allocation rule. More concretely, GETF is parameterized by a group assignment function $f(\cdot)$, and a tie-breaking rule. GETF proceeds in two stages. The pseudocode for GETF is presented in Algorithm 1.

GETF is an algorithmic framework because it can be instantiated with different group assignment and tie-breaking rules. To understand these, consider a situation where the m machines are divided into K groups $M_1, M_2, \ldots, M_K$ by a group assignment rule. Let $f(j)$ denote the group of machines to which task j can be assigned, $j = 1, \ldots, n$. In other words, task j can be only assigned to machines in group $f(j)$. Given this notation, a schedule under GETF consists of two mappings: a mapping $h(\cdot)$ from each task to its assigned machine and a mapping $t(\cdot)$ from each task to its starting time.

The choice of the group assignment rule has a significant impact on the performance of GETF. Our technical results are based on the following specific group assignment function. Note that our results hold for any tie-breaking rule.

Group Assignment Rule. The group assignment rule $f(\cdot)$ that we focus on is adapted from the work of [6], which studies the setting *without* communication time. Specifically, "similar" machines are grouped together as follows.

First, all the machines with speed less than a $\frac{1}{m}$ fraction of the speed of the fastest machine are discarded. Then, the remaining machines are divided into K groups $M_1, M_2, \ldots, M_K$ where $K = \lceil \log_\gamma m \rceil$ and $\gamma = \log m / \log \log m$. Note that $K = O(\log m / \log \log m)$. Given the removal of the slowest machines, we can assume that any remaining machine has speed within a factor of $\frac{1}{m}$ of the fastest machine. Without loss of generality, we assume the speed of the fastest machine is m and the group M_k contains machines with speeds in range $[\gamma^{k-1}, \gamma^k)$.

After dividing machines into K groups in the preprocessing step, the task of assignment is more delicate than that of

Algorithm 1 Generalized Earliest Time First (GETF)

INPUT: group assignment rule $f(\cdot)$, tie-breaking rule
OUTPUT: schedule $\mathcal{S}$ with machine assignment mapping $h(\cdot)$ and starting time mapping $t(\cdot)$

```
$R \leftarrow \{1, 2, \ldots, n\}$
while $R \neq \emptyset$ do
    $A = \{j : j \in R, \nexists j'$ s.t. $j' \in R$ and $j' \prec j\}$
    For $j \in A, t'_j =$ earliest starting time on machine $m'_j$
        s.t. $m'_j \in f(j)$
    $B = \{j : j = \arg\min_{j' \in A} t(j')\}$
    Choose $j$ from $B$ to start on machine $m'_j$ with a starting time $t'_j$ based on the given tie-breaking rule
    $h(j) = m'_j, t(j) = t'_j$
    $R \leftarrow R \setminus \{j\}$
end while
```

division. The design of $f(\cdot)$ is based on the solution of a linear program (LP), which is a relaxed version of the following mixed integer linear program (MILP).

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

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

$$w_j \sum_i \frac{x_{i,j}}{s_i} \leq C_j \qquad \forall j \tag{1b}$$

$$C_{j'} + w_j \sum_i \frac{x_{i,j}}{s_i} \leq C_j \qquad j' \prec j \tag{1c}$$

$$\frac{1}{s_i} \sum_j w_j x_{i,j} \leq T \qquad \forall i \tag{1d}$$

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

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

Note that the MILP is only designed to produce a group assignment rule and the constraints of the MILP are necessary but not sufficient for a feasible schedule. For example, a constraint to guarantee no time-sharing is missing. As a result, the optimal for this MILP is a lower bound of $\text{OPT}^{(i)}$. In the MILP, $x_{i,j} = 1$ if task j is assigned to machine i; otherwise $x_{i,j} = 0$. For each task j, C_j denotes the completion time of task j. Let T denote the objective. Constraint (1a) ensures that every task is processed on some machine. For any task j, processing time $w_j \sum_i \frac{x_{i,j}}{s_i}$ is bounded by its completion time as in constraint (1b). Constraint (1c) enforces the precedence constraints between any predecessor-successor pair (j', j). The total load assigned to machine i is $w_j \sum_i \frac{x_{i,j}}{s_i}$ and it should not be greater than the makespan, as in constraint (1d). Finally, constraint (1e) states that the makespan should not be smaller than the completion time of any task.

Since we cannot solve the MILP efficiently, we relax it to form an LP by replacing constraint (1f) with $x_{i,j} \geq 0$. Let $x^*_{i,j}, C^*_j, T^*$ denote the optimal solution of this LP. Note that T^* provides a lower bound on $\text{OPT}^{(i)}$. For a set $M' \subseteq M$ of machines, let $s(M')$ denote the total speed of machines in M', i.e., $s(M') = \sum_{i \in M'} s_i$. Define $x^*_{M',j}$ as the total fraction of task j assigned to machines in set M': $x^*_{M',j} = \sum_{i \in M'} x^*_{i,j}$. For any task j, define ℓ_j as the largest group index such that at least more than half of tasks are assigned to machines in groups $M_\ell, \ldots, M_K$: $\ell_j = \max_\ell \ell \quad \text{s.t.} \quad \sum_{k=\ell}^K x^*_{M_k,j} \geq \frac{1}{2}$. Each task j is assigned to the group $f(j)$ that maximizes the total speed of machines in that group among candidates $M_{l_j}, \ldots, M_K$, i.e., $f(j) = \arg\max_{M_k : \ell_j \leq k \leq K} \quad s(M_k)$.

4. RESULTS

We now state our main technical result, which bounds the approximation ratio of GETF in settings with related machines and heterogeneous communication time. This is the first algorithmic framework with a provable approximation ratio in this setting.

Our result provides a bound in terms of the communication time of a terminal chain of the schedule. In particular, let $\mathcal{C} : c_1 \prec c_2 \prec \ldots \prec c_N$ of length N be a terminal chain for the schedule and define C as the communication time over such a chain in the worst case, i.e., $C = \sum_{j=2}^N \frac{w_{c_{j-1},c_j}}{\bar{s}(c_{j-1},c_j)}$, where $\bar{s}(c_{j-1}, c_j)$ is defined as the slowest speed between the assigned machine $h(c_{j-1})$ of task c_{j-1} and any machine in the same group $f(c_j)$ as the assigned machine $h(c_j)$ of task c_j, i.e., $\bar{s}(c_{j-1}, c_j) = \min_{i \in f(c_j)} s_{h(c_{j-1}),i}$.

Theorem 4.1. *For any schedule $\mathcal{S}$ produced by GETF with group assignment rule $f(\cdot)$ described above,*

$$C_{max}(\mathcal{S}) \leq O(\log m / \log \log m) OPT^{(i)} + C,$$

where $OPT^{(i)}$ is the optimal schedule length obtained ignoring the communication time.

The key technical advance that enables our new result is a dramatically simplified analysis of ETF in the setting of identical machines. Theorem 4.1 reduces to state-of-the-art result for ETF in [4] in settings with identical machines.

Proposition 4.2. *Consider a setting with m identical machines. For any schedule $\mathcal{S}$ produced by GETF,*

$$C_{max}(\mathcal{S}) \leq (2 - \frac{1}{m}) OPT^{(i)} + C',$$

where $C' = \frac{1}{m} \sum_{j=2}^N \sum_{i=1}^m \frac{w_{c_{j-1},c_j}}{s_{h(c_{j-1}),i}}$.

5. REFERENCES

[1] F. A. Chudak and D. B. Shmoys. Approximation algorithms for precedence-constrained scheduling problems on parallel machines that run at different speeds. *Journal of Algorithms*, 30(2):323–343, 1999.

[2] M. R. Garey and D. S. Johnson. Computers and intractability: a guide to np-completeness, 1979.

[3] R. L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

[4] J.-J. Hwang, Y.-C. Chow, F. D. Anger, and C.-Y. Lee. Scheduling precedence graphs in systems with interprocessor communication times. *SIAM Journal on Computing*, 18(2):244–257, 1989.

[5] Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys (CSUR)*, 31(4):406–471, 1999.

[6] 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)*, pages 283–294, Oct 2017.

[7] R. Mayer, C. Mayer, and L. Laich. The tensorflow partitioning and scheduling problem: it's the critical path! In *Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning*, pages 1–6. ACM, 2017.