EECS Rising Stars 2021

Scheduling with Communication Delay in Near-Linear Time

Quanquan C. Liu[1], Manish Purohit[2], Zoya Svitkina[2], Erik Vee[2], Joshua R. Wang[2]

[1] MIT CSAIL quanquan@mit.edu

[2] Google Research {mpurohit, zoya, erikvee, joshuawang}@ncsu.edu

Motivation

How **computationally expensive** is it to perform **approximately-optimal scheduling**?

With **growing input sizes and large data centers**, highly desirable to obtain a scheduling algorithm whose **running time is linear** in the size of the input.

All current state-of-the-art algs take super-linear time.

Problem Definition

Classical scheduling problem with **communication delay** on **identical machines**, **unit size jobs**

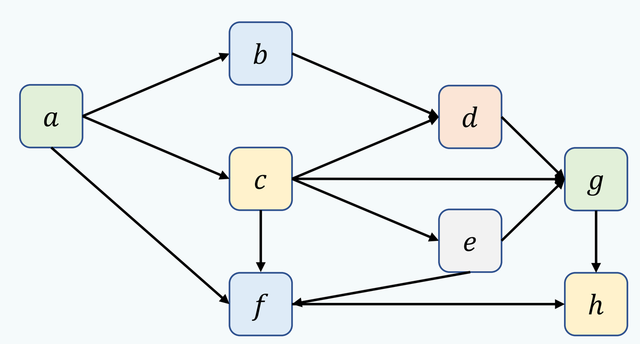
Precedence-constrained jobs modeled as **directed acyclic graph (DAG)**, **vertex is job, edge indicates order**

Given DAG $G = (V, E)$, n unit-sized jobs, M identical machines and communication delay ρ, provide a near-optimal schedule in near-linear time.

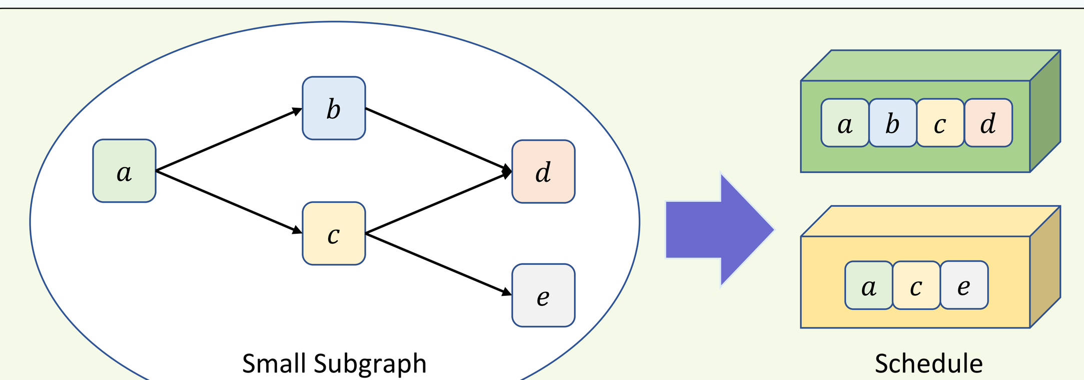
Near-Linear Time Algorithm

❖ **Previous Result:** Computing an optimal schedule is NP-hard in general. **Best previous result** by Lepere and Rapine obtain $O\left(\frac{\ln \rho}{\ln \ln \rho}\right)$-approx. in $\Omega(m\rho + n \ln M)$ time

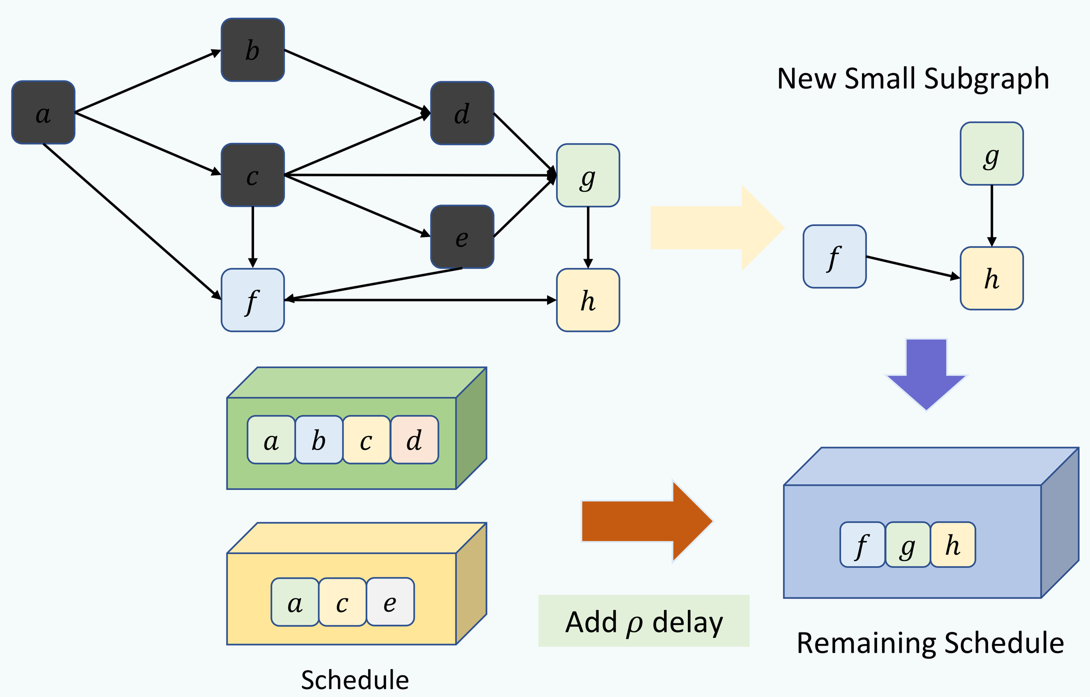
❖ **Our Result:** $O\left(n \ln M + \frac{m \ln^3 n \ln \rho}{\ln \ln \rho}\right)$ time and $O\left(\frac{\ln \rho}{\ln \ln \rho}\right)$-approx. algorithm, **tight up to polylog factors**



(1) Initial Input DAG



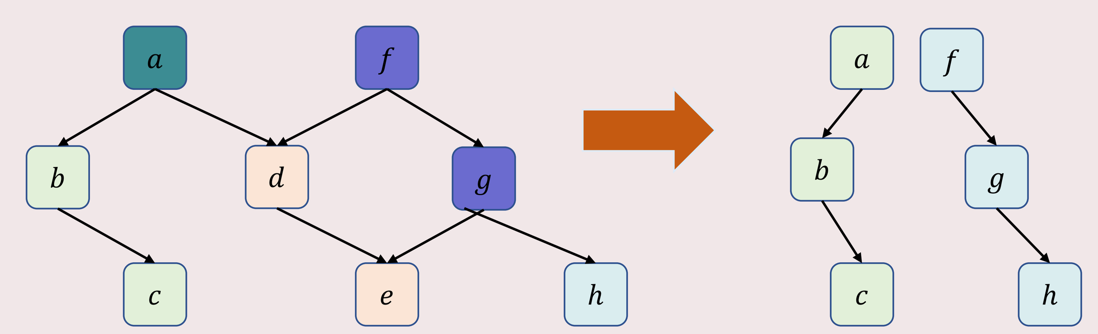
(2) Find a small subgraph in the input DAG and schedule it



(3) Remove scheduled vertices from the graph. Add a ρ communication delay to schedule after previously scheduled jobs. Find a new small subgraph and schedule it.

Graph Problem!

How do you find jobs with ancestor sets that do not overlap too much **quickly**?

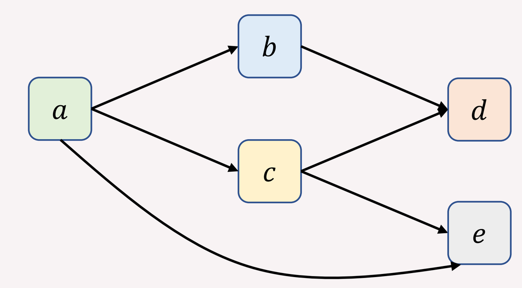


[Lepere and Rapine] Renaud Lepere and Christophe Rapine. An asymptotic O(ln ρ/ ln ln ρ)-approximation algorithm for the scheduling problem with duplication on large communication delay graphs. In STACS, volume 2285 of Lecture Notes in Computer Science, pages 154–165, 2002.

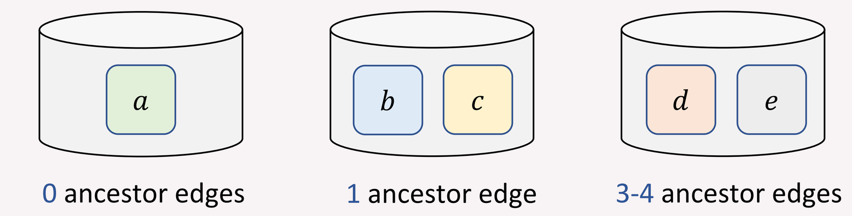
[Graham] R. L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In Proceedings of the May 16-18, 1972, Spring Joint Computer Conference, AFIPS '72 (Spring), page 205–217, New York, NY, USA, 1971. Association for Computing Machinery.

Scheduling Small Subgraphs

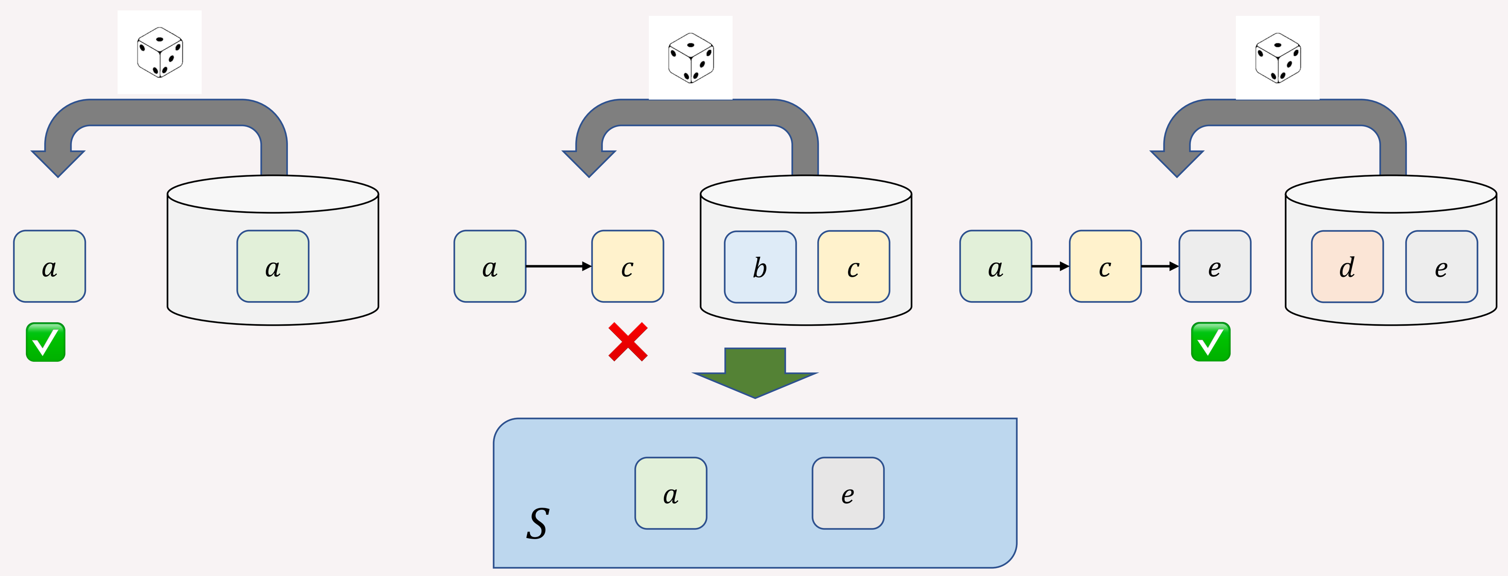
❖ **Small Subgraph:** A subgraph of the input DAG where each vertex has **at most $\rho - 1$ ancestors**



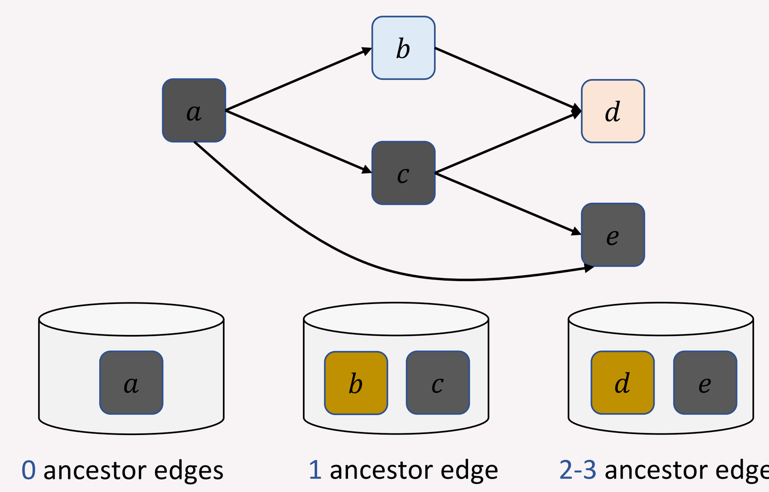
(1) Initial small subgraph



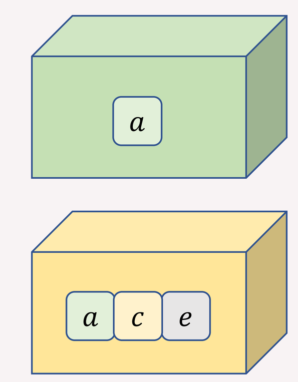
(2) Vertices bucketed according to ancestor edge **estimate**



(3) Vertices **uniformly-at-random sampled from bucket**, checked for ancestor overlap and added to S



(5) Pruning vertices with large overlap with S



(6) List scheduling jobs in S

❖ **Estimating Number of Ancestor Edges:** We use **count-distinct estimators** to estimate the number of ancestors and edges in the induced subgraph of each node and its ancestors

❖ **Partitioning Vertices to Buckets:** First **partition vertices** into $O(\log \rho)$ buckets based on ancestor edge estimates

❖ **Sampling Vertices from Buckets:** Vertices (and ancestors) are sampled from buckets and added to S if its ancestors **do not overlap too much** with S. Keep sampling until we've seen $O(\log n)$ vertices **in a row that we do not add to S** or bucket is empty.

❖ **Pruning All Stale Vertices from Buckets:** Vertices with ancestor sets that overlap too much with S are **pruned** or removed from buckets

❖ **Standard list scheduling jobs in S:** Duplicate all ancestors of a job, schedule a job and all its (duplicated) ancestors on the same machine

Analysis Key Insights

❖ **Estimating Number of Ancestor Edges:** Our **count-distinct estimator algorithm on DAGs** return $(1 \pm \epsilon)$-approximations to the number of ancestors (and edges) with probability at least $1 - \frac{1}{n^d}$ for any constant $d \geq 1$ in $O\left(\frac{1}{\epsilon^2} \log^2 n\right)$ time per vertex

❖ **Partitioning Vertices to Buckets:** A vertex v is in the i-th bucket if the estimate of the number of edges in the induced subgraph of its ancestors, $\hat{e}(v)$, **is in** $[2^i, 2^{i+1})$; we can partition all vertices in $O\left(\frac{m+n}{\epsilon^2} \log^2 n\right)$ time

❖ **Sampling Vertices from Buckets:** A sampled vertex v and all its ancestors **are added to S** if at least a γ fraction of its ancestor set is not in S. Once we stop sampling, **less than a constant fraction of the vertices remaining** in each bucket can be added to S (for any constant)

❖ **Pruning All Stale Vertices from Buckets:** *All* vertices that **cannot be added to S** are pruned; any pruned vertex **has at most a 4γ-fraction** of ancestors not in S

❖ **Standard list scheduling jobs in S:** **Graham's list scheduling** algorithm where scheduling $|S|$ jobs requires $O(|S| \log M)$ time