Multiple DAG Scheduling on Multiple Parallel Machines

ABSTRACT

ACM Reference format:
. 2016. Multiple DAG Scheduling on Multiple Parallel Machines. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 3 pages.
DOI: 10.475/123_4

1 ASSIGNING AND SCHEDULING JOB TASKS ON MACHINES

1.1 Model

Suppose that there is a set of jobs $\mathcal{J}$ where each job $J \in \mathcal{J}$ can be separated into a set of individual tasks $\mathcal{T}_J$. For a particular set tasks for a job $\mathcal{T}_J$, there is a directed acyclic graph (DAG) that represents the precedence constraints between the tasks. A precedence constraint $j \to k$ means that task j must completely finish before task k can start. An in-tree is a special type of DAG in which each task has at most one successor.

Each task i requires a total of p_i units of processing time to be done. Let c_i be the time at which task i is completed. Therefore, the completion time for a particular job J is the time at which its latest task completes $C_J = \max_{i \in \mathcal{T}_J}\{c_i\}$.

There are M parallel identical machines that can perform work on the job tasks. Each task can only be processed by one machine at a time and each machine can only process one task at a time. We assume that each task can be preempted an unlimited number of times by any machine.

1.2 Problem

The goal is to minimize the sum of the average completion time among all of the jobs:

$$\min \frac{1}{|\mathcal{J}|} \sum_{J \in \mathcal{J}} C_J \tag{1}$$

1.2.1 Decisions. For each task i and time t we must decide the allocation $a_i(t) \in \{0, 1\}$ for one or no machines to work on it. Let $\mathcal{A}$ be the schedule of all allocations for all tasks and times.

1.3 Shortest Remaining Processing Time

When deciding which job should have priority over another when scheduling, Shortest Remaining Processing Time (SRPT) is a common intuitive rule. However, when comparing different in-trees, the challenge becomes on how to decide which metric defines the "processing time" of an in-tree. We explore three intuitive metrics and give examples to show their suboptimality.

1.3.1 Height. Let us define the height of a job as the minimum length of time it would take to complete with an infinite number of machines. In other words, this is the length of its critical path. The intuition behind giving priority to a job with a smaller height is that it leaves the system faster and minimally delays jobs with larger heights that are scheduled later.

However using height as the "processing time" metric in a SRPT rule to schedule a set of in-tree jobs gives an arbitrarily bad approximation ratio. We use the following example to show this. Suppose we have a set of $M + K$ jobs where $M \geq 2$ and $K \geq 1$. We separate these jobs into Type A and Type B jobs. There are M Type A jobs in which each job is made up of one task with a processing time of $M + 1$. There are K Type B jobs in which each job has $M + 1$ tasks; the first M tasks each has a processing time of M, while the $(M + 1)$-th task has a processing time of 0. Also, the $(M + 1)$-th task has a precedence constraint from all other tasks within that job.

SRPT with height as the processing time metric gives priority to the Type B jobs since the Type A jobs all have a height of $M + 1$ while the Type B jobs all have a height of M. The Type B jobs are scheduled first and occupy the M machines for KM units of time and completes at times $C_{\mathrm{B}i} = Mi$ for $i \in \{1, ..., K\}$. Afterwards, the Type A jobs each take a single machine and run for $M + 1$ units of time giving completion times of $C_{\mathrm{A}i} = M(K + 1) + 1$ for $i \in \{1, ..., M\}$. This schedule results in the following average completion time:

$$\begin{aligned}
\overline{C}_{\text{SRPT-height}} &= \frac{1}{M+K}\left(\sum_{i=1}^{M} C_{\mathrm{A}i} + \sum_{i=1}^{K} C_{\mathrm{B}i}\right) \\
&= \frac{1}{M+K}\left(\sum_{i=1}^{M}(M(K+1)+1) + \sum_{i=1}^{K} Mi\right) \\
&= \frac{1}{M+K}\left(M(M(K+1)+1) + M\frac{K(K+1)}{2}\right) \\
&= \frac{M((2M+K)(K+1)+2)}{2(M+K)}
\end{aligned} \tag{2}$$

On the other hand, the optimal schedule would first process all of the Type A jobs using all of the machines for $M + 1$ units of time giving completion times of $C_{\mathrm{A}i} = M + 1$ for $i \in \{1, ..., M\}$. Afterwards, the Type B jobs are processed sequentially by all of the machines which gives the completion time $C_{\mathrm{B}} = M + 1 + Mi$ for $i \in \{1, ..., K\}$. This schedule results in the following average completion time:

$$\begin{aligned}
\overline{C}_{\text{OPT}} &= \frac{1}{M+K}\left(\sum_{i=1}^{M} C_{\mathrm{A}i} + \sum_{i=1}^{K} C_{\mathrm{B}i}\right) \\
&= \frac{1}{M+K}\left(\sum_{i=1}^{M}(M+1) + \sum_{i=1}^{K}(M+1+Mi)\right) \\
&= \frac{1}{M+K}\left((M+K)(M+1) + M\frac{K(K+1)}{2}\right) \\
&= \frac{2(M+K)(M+1) + MK(K+1)}{2(M+K)}
\end{aligned} \tag{3}$$

Conference'17, Washington, DC, USA

DOI: 10.475/123_4

From this example we have a lower bound on the approximation ratio for SRPT with height as the processing time metric:

$$\frac{\overline{C}_{\text{SRPT-height}}}{\overline{C}_{\text{OPT}}} \geq \frac{M((2M+K)(K+1)+2)}{2(M+K)(M+1)+MK(K+1)} = K+1-\frac{((K+1)(K^2+2K+2)-2)M+2K(K+1)}{2M^2+(K+2)(K+1)M+2K} \tag{4}$$

As M approaches infinity, the approximation ratio goes toward any fixed integer value of $K+1$.

1.3.2 Width. Let us define the width of a job as the minimum number of machines a job needs so that it completes at the same time as it would with an infinite number of machines. This metric would define how much a job could be parallelized. The intuition behind giving priority to a job with a smaller width is that it uses less machines at any given time which would allow other jobs to run simultaneously. Whereas a job with a larger width might block other jobs from being processed and increase all of their completion times.

However using width as the "processing time" metric in a SRPT rule to schedule a set of in-tree jobs gives an approximation ratio $\Omega(M)$ for $M \geq 2$. We use the following example to show this. Suppose we have a set of $M+1$ jobs where $M \geq 2$ and we separate these jobs into Type A and Type B jobs. There are M Type A jobs in which each job has $M+1$ tasks; the first M tasks each have a processing time of 1, while the $(M+1)$-th task has a processing time of 0. Also, the $(M+1)$-th task has a precedence constraint from all other tasks within that job. There is only one Type B job which has M tasks; the first $M-1$ tasks each has a processing time of M^2, while the M-th task has a processing time of 0. Also, the M-th task has a precedence constraint from all other tasks within that job.

SRPT with width as the processing time metric gives priority to the Type B job since the Type A jobs all have a width of M while the Type B job has a width of $M-1$. The one Type B job is scheduled first and so it occupies $M-1$ machines for the first M^2 units of time and completes at time $C_{\text{B}} = M^2$. This leaves only one machine available to schedule all of the Type A jobs for the first M^2 units of time. Putting the Type A jobs in an arbitrary order and consecutively processing each of them until completion on the remaining one machine gives the following completion times $C_{\text{A}i} = Mi$ for $i \in \{1, \ldots, M\}$. This schedule results in the following average completion time:

$$\begin{aligned}\overline{C}_{\text{SRPT-width}} &= \frac{1}{M+1}\left(\sum_{i=1}^{M} C_{\text{A}i} + C_{\text{B}}\right) \\ &= \frac{1}{M+1}\left(\sum_{i=1}^{M} Mi + M^2\right) \\ &= \frac{1}{M+1}\left(M\frac{M(M+1)}{2} + M^2\right) \\ &= \frac{M^2(M+3)}{2(M+1)}\end{aligned} \tag{5}$$

On the other hand, the optimal schedule would first consecutively process the Type A jobs using all M machines giving the following completion times $C_{\text{A}i} = i$ for $i \in \{1, \ldots, M\}$. Then the Type B job would be processed by $M-1$ machines for M^2 units of time giving the completion time $C_{\text{B}} = M + M^2$. This schedule results in the following average completion time:

$$\begin{aligned}\overline{C}_{\text{OPT}} &= \frac{1}{M+1}\left(\sum_{i=1}^{M} C_{\text{A}i} + C_{\text{B}}\right) \\ &= \frac{1}{M+1}\left(\sum_{i=1}^{M} i + (M+M^2)\right) \\ &= \frac{1}{M+1}\left(\frac{M(M+1)}{2} + (M+M^2)\right) \\ &= \frac{3}{2}M\end{aligned} \tag{6}$$

From this example we have a lower bound on the approximation ratio for SRPT with width as the processing time metric:

$$\frac{\overline{C}_{\text{SRPT-width}}}{\overline{C}_{\text{OPT}}} \geq \frac{\frac{M^2(M+3)}{2(M+1)}}{\frac{3}{2}M} = \frac{M(M+1)+2M}{3(M+1)} = \frac{1}{3}M + \frac{2}{3}\left(\frac{M}{M+1}\right) \tag{7}$$

This gives the approximation ratio to be $\Omega(M)$ for $M \geq 2$ to schedule in-trees.

1.3.3 Total Task Processing Time. Let us define the "processing time" of a job as the sum of all of the processing times of the tasks. This metric gives SRPT approximation ratio of $4/3$ for $M \geq 2$. We use the following example to show this. Suppose we have 2 jobs. Job A has $M+1$ tasks: task 1 has a processing time of $2M-1$, and tasks 2 through M have processing times of $M-1$. Task $M+1$ has a processing time of 0 and has a precedence constraint from all other tasks in the job. Job B has $M+1$ tasks: tasks 1 through M have processing times of $M-1$. Task $M+1$ has a processing time of 0 and has a precedence constraint from all other tasks in the job.

SPRT with total task processing time as the metric gives priority to Job B since it has a total task processing time of $M^2 - M$ while Job A has a total task processing time of M^2. Scheduling Job B first gives it a completion time of $C_{\text{B}} = M-1$ and then Job A gives is a completion time of $C_{\text{A}} = M-1+2M-1 = 3M-2$. Therefore, the average completion time is $\overline{C}_{\text{SRPT-total}} = 4M-3$.

However, if we schedule Job A first and then allow Job B to be scheduled before the non-critical tasks of Job A. We get $C_{\text{A}} = 2M-1$ and $C_{\text{B}} = M$. Therefore, the average completion time is $\overline{C}_{\text{SRPT-total}} = 3M-1$.

This gives an approximation ratio of:

$$\frac{\overline{C}_{\text{SRPT-total}}}{\overline{C}_{\text{OPT}}} \geq \frac{4M-3}{3M-1} \tag{8}$$

which approaches $4/3$ as M goes to infinity.

1.4 Proposed Heuristic

(1) SRPT with the metric as sum of task processing times in its critical paths. In the bad example for SPRT total task processing time, this would give us the schedule that gives the long task of Job A priority over the tasks of Job B which is the optimal schedule in that example. However, the old 9/8 example in the presentation would apply to this. There could be a worse example.
   - Maybe Total task processing time of the critical path up to the minimum height when comparing two jobs.

REFERENCES

APPENDIX