Real-time GDA

ABSTRACT

ACM Reference format:
. 2016. Real-time GDA. In *Proceedings of ACM Conference, Washington, DC, USA, July 2017 (Conference'17),* 2 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. Each task i requires w_i units of work to be done. Let c_i be time at which task i is fully completed. Therefore, the completion time for a particular job J is $C_J = \max_{i \in \mathcal{T}_J}\{c_i\}$.

There is a set of machines $\mathcal{M}$ that perform work on the job tasks. Each machine $m \in \mathcal{M}$ has a specific speed s_m at which it performs work. Therefore, if task i is assigned to machine m, then it takes w_i/s_m units of time to complete the task. Let W_m be the total amount of work that machine m does and b_m be the per unit cost for doing that work.

We make the following assumptions:

(1) Each task can only be completed by one machine but can be preempted an unlimited number of times at that machine.
(2) Each machine can only work on one task at any moment in time.

1.2 Problem

The goal is to minimize the sum of the average completion time among all of the jobs and the cost of using the machines:

$$\min \frac{1}{|\mathcal{J}|}\sum_{J\in\mathcal{J}} C_J + \sum_{m\in\mathcal{M}} b_m W_m \tag{1}$$

1.2.1 Decisions. For each task i we must assign it to a particular machine m to be worked on. Let $\mathcal{A} : \cup_{J\in\mathcal{J}}\mathcal{T}_J \to \mathcal{M}$ be the assignment. For each machine $m \in \mathcal{M}$ we must decide a priority ordering Q_m of its assigned tasks. A machine will always be busy as long as there is an available task assigned to it that is not yet fully completed. From a group of available tasks assigned to it, a machine will choose to work on the task that has the highest priority at any moment of time.

1.3 Special Cases: Single machine, $|\mathcal{M}| = 1$

By restricting to $|\mathcal{M}| = 1$, we eliminate the assignment decision problem.

1.3.1 Single job, $|\mathcal{J}| = 1$. By restricting to only a single job with arbitrary precedence constraints, we only need to focus on the following minimax objective function:

$$\min_{Q} \max_{i\in\mathcal{T}}\{c_i\} \tag{2}$$

which can be solved in $O(|\mathcal{T}|^2)$ using a more general solution by [2]. The general solution allows for arbitrary release times for each task, and the more general objective $\min\max\{f_i(c_i)\}$ where $f_i(\cdot)$ is a monotonically nondecreasing function which could include deadlines.

1.3.2 Single-task jobs, $|\mathcal{T}_J| = 1, \forall J \in \mathcal{J}$. By restricting to jobs with only one task we eliminate the precedence constraints. We get the following objective function:

$$\min_{Q} \frac{1}{|\mathcal{J}|}\sum_{J\in\mathcal{J}} C_J \tag{3}$$

which can be solved in $O(|\mathcal{J}|\log|\mathcal{J}|)$ using a more general solution by [1] or §4.3.1 in [3]. The general solution allows for arbitrary release times for each job.

1.4 If preemptions are not allowed

Suppose we decide to restrict the first assumption to: *Each task can only be completed by one machine and cannot be preempted.* Then our problem becomes NP-hard since it is a special case in [6] which was shown to be NP-complete for $|\mathcal{M}| = 2$, $s_1 = s_2$, $b_1 = b_2 = 0$, $|\mathcal{J}| = 1$, and no precedence constraints between the tasks for the single job.

1.5 If you allowing tasks to be split between machines

Suppose we decide to restrict the first assumption to: *Each task can be completed by any number of machines and can be preempted an unlimited number of times at those machines. However, a task can only be worked on by one machine at any given point of time.* Essentially this means "preemption" in the literature for multiple machine scheduling.

1.5.1 Single-task jobs, $|\mathcal{T}_J| = 1, \forall J \in \mathcal{J}$, and $b_m = 0$ for all $m \in \mathcal{M}$. This allows for a solution to be found in $O(|\mathcal{J}|\log|\mathcal{J}| + |\mathcal{M}||\mathcal{J}|)$ by [4] or §5.1.2 in [3].

1.5.2 Two machines and Single job, $|\mathcal{M}| = 2$, $|\mathcal{J}| = 1$, and $b_1 = b_2 = 0$. This allows for a solution to be found in $O(|\mathcal{T}_J|^2)$ by [5].

1.5.3 Single job, $|\mathcal{J}| = 1$ and $b_m = 0$ for all $m \in \mathcal{M}$. This is NP-hard even when restricted to identical machines ($s_m = s$ for all $m \in \mathcal{M}$) and all tasks the same size ($w_i = w$ for all $i \in \mathcal{T}$) [7].

1.6 Next steps

(1) Combine the first two special cases so that on a single machine, we can minimize

$$\min_{Q} \frac{1}{|\mathcal{J}|}\sum_{J\in\mathcal{J}} C_J \tag{4}$$

for multiple jobs, each with arbitrary precedence constraints.

Conference'17, Washington, DC, USA

DOI: 10.475/123_4

(2) Expand the previous step to cover multiple machines starting with two machines. Assume that the assignment is already given. Find equations that describe an optimal structure. If the problem seems NP-hard, try to prove it.
(3) Use the equations to find to optimize for the assignment. Maybe the problem becomes NP-hard at this point (prove it).

REFERENCES

[1] Kenneth R Baker. 1974. *Introduction to sequencing and scheduling.* John Wiley & Sons.
[2] Kenneth R Baker, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. 1983. Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Operations Research* 31, 2 (1983), 381–386.
[3] Peter Brucker and P Brucker. 2007. *Scheduling algorithms.* Vol. 3. Springer.
[4] Jacques Labetoulle, Eugene L Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. 1984. Preemptive scheduling of uniform machines subject to release dates. (1984).
[5] EL Lawler. 1982. Preemptive scheduling of precedence-constrained jobs on parallel machines. *Deterministic and stochastic scheduling* 84 (1982), 101–123.
[6] Jan Karel Lenstra, AHG Rinnooy Kan, and Peter Brucker. 1977. Complexity of machine scheduling problems. *Annals of discrete mathematics* 1 (1977), 343–362.
[7] Jeffrey D Ullman. 1976. Complexity of sequencing problems. *Computer and Job-Shop Scheduling Theory, EG Co man, Jr.(ed.)* (1976).

APPENDIX