# Resource Allocation Algorithms
Lecture 04

Lecture by Dr. Tami Tamir
Typeset by Steven Karas

2018-03-20
Last edited 17:12:23 2018-04-17

**Disclaimer** These lecture notes are based on the lecture for the course Resource Allocation Algorithms, taught by Dr. Tami Tamir at IDC Herzliyah in the spring semester of 2017/2018. Sections may be based on the lecture slides written by Dr. Tami Tamir.

**Agenda**

- Job Scheduling
- Facility Locations

# 1 Job Scheduling

## 1.1 Precedence Constraints

For the problem $P|\text{prec}|C_{\max}$, there are many variants, from the number of machines to constraints on the precedence graph (in-trees and out-trees have efficient algorithms).

Note that for multiple machines, the problem $P||C_{\max}$ is already $\mathcal{NP}$-hard. We can try List-Scheduling on the topological sort of the precedence graph.

Graham presented in 1966 that for every topological sort $L'$, every relaxed set of precedence constraints $\Theta' \subseteq \Theta$, every vector of processing times $p' \leq p$, and a different set of machines $m'$, it holds that:

$$\frac{C'_{\max}}{C_{\max}} \leq 1 + \frac{m-1}{m'}$$

As a special case, we see that if $m' = m$, then we get the general case for List-Scheduling.

## 1.2 Unrelated machines

For the problem $R||\sum_j C_j$, we are given a processing time matrix $P$ such that $p_{ij}$ is the processing time of job $J_j$ on machine $M_i$. An important observation is that the $k$-th last job to run on a machine contributes exactly $k$ times its processing time to the sum of completion times.

### 1.2.1 Reduction to bipartite matching

A perfect bipartite matching is defined as $|M| = |A| \leq |B|$. If a minimum-weight perfect matching exists, it is possible to compute it in polynomial time.

Let $G$ be a bipartite graph $V = A \cup B$ where $A$ is the set of jobs, and $B$ consists of $nm$ vertices $b_{ik}$ where vertex $b_{ik}$ represents the $k$-th from the last position on machine $i$. The edges $E$ contain an edge $(v_j, w_{ik})$ with weight $k \cdot p_{ij}$ for every node in $A$ and every node in $B$.

**Proof of reduction** We will prove that a minimum weight prefect matching corresponds to an optimal schedule.

In an optimal schedule, every job appears exactly once, and every processing slot has exactly one job. Thus, the edges in a matching represent a valid schedule. The weights of the edges are such that they represent the objective function.

We claim that for all machines, the vertices that participate in a perfect matching represent a sequence $w_{i1}, \ldots, w_{il}$ where $l$ is the number of jobs that are allocated in the schedule. For the

purpose of contradiction assume that $w_{ia}$ is not in the matching yet $w_{i,a+1}$ is. For every $V$, the weight of the edge....

## 1.3 Open shop scheduling

The problem $O||C_{\max}$ is called open shop scheduling, where each job $J_j$ consists of $m$ sub-jobs. The order in which the different operations are performed is not constrained. Two sub-jobs of the same job cannot be performed simultaneously.

This problem is $\mathcal{NP}$-hard for $m > 2$.

There are two basic lower bounds on any correct schedule: the maximum sum of processing times on a particular sub-job type, and the maximum sum of processing times for a given job's sub-jobs.

**Example:**

| | Alice | Bob | Charlie |
|---|---|---|---|
| Bank | 8 | 4 | 5 |
| Post | 3 | 9 | 2 |

### 1.3.1 2-approximation

A busy schedule for $O||C_{\max}$ is a schedule where there is no point in time where a job could be scheduled that it is not. This can be implemented by a simple greedy algorithm. Any such schedule is a 2-approximation.

**Proof** Consider the machine $M'$ that finishes last. Let $j'$ be the last job on $M'$. At any time during the schedule, either $M'$ is processing a job or $j'$ is being processed by some other machine. This follows

### 1.3.2 $\mathcal{NP}$-hardness

We will show that $O_3||C_{\max}$ is $\mathcal{NP}$-hard by reduction from PARTITION. Let $a_1, \ldots, a_n$ be an input for PARTITION such that $\sum a_i = 2B$.

We construct an input for $O_3||C_{\max}$ with $3n + 1$ jobs.

Construct 3 jobs for each $a_j$ such that $p_{j1} = a_j$, $p_{j2} = 0$; $p_{j3} = 0$; $p_{j1} = 0$, $p_{j2} = a_j$, $p_{j3} = 0$; and $p_{j1} = 0$, $p_{j2} = 0$, $p_{j3} = a_j$. Construct a final job $p_{3n+1,1} = B$, $p_{3n+1,2} = B$, and $p_{3n+1,3} = B$.

We claim that there exists a schedule for this with $C_{\max} = 3B$ iff there is a partition.

### 1.3.3 Optimal for $O_2||C_{\max}$[1]

The full algorithm is on slides 43-50.

## 1.4 Flow shop scheduling

Denoted as $F_m||$, given $m$ machines, all the jobs must be process by all the machines in the same order. Denote $p_{ij}$ as the processing time required by $J_j$ on $M_i$.

It is known that $F_m||C_{\max}$ is $\mathcal{NP}$-hard for any $m > 2$. Johnson presented a simple optimal algorithm for $m = 2$ in 1954.

In any $F_2$ problem, the machine $M_2$ is dependent upon $M_1$ releasing jobs. As such, $M_1$ is never idle in an optimal schedule. Since all the jobs are available at $t = 0$, our goal is to minimize the time in which $M_2$ is idle.

**Permutation Schedule** A permutation schedule is a schedule in which the jobs are processed in the same order by $M_1$ and $M_2$. There exists an optimal schedule which is a permutation schedule [2].

[1] This will not be in the homeworks or on the exam

[2] Proof follows from exchange argument, can be found on slide 53

**Johnson Ordering Rule** Let $A$ be the set of jobs for which $a_j \leq b_j$. Let $B$ be the rest of the jobs $a_j > b_j$.

The Johnson rule is to sort jobs in $A$ by increasing values of $a_j$, and then jobs in $B$ by decreasing order of $b_j$.

**Optimality** Number the jobs by the order in which they are scheduled in a permutation schedule. Let $J_k$ be the first job on $M_2$ after its last idle section. $J_k$ was not waiting between $M_1$ and $M_2$.

Note that $C_k = a_1 + \ldots + a_k + b_k$. $M_2$ is not idle after $J_k$ is processed, so $C_{\max} = a_1 + \ldots + a_k + b_k + \ldots + b_n$.

For any $c$, we can reduce $c$ from all the $p_i j$ values without changing the relative performance of different permutation schedules.

Formally, the proof is by induction. For $n = 1$, any non-idle schedule is trivially optimal. Note that the Johnson ordering rule holds for this case.

Assume the Johnson rule is optimal for $n - 1$ jobs. Now consider an instance with $n$ jobs. Let $c = \min_j \min a_j, b_j$. Reduce $c$ from all processing times. As a result, there exists a job with $a_j = 0$ or $b_j = 0$. If $a_j = 0$, then $j \in A$, and it is first in $A$, otherwise $j \in B$, and it is last in $B$.

...

The remainder of the proof can be found on slides 57, 58.

# 2 Facility Location

Facility location is the set of problems surrounding where facilities that provide services should be located so as to maximize some objective function (typically min max distance or min sum distance).

Antennas are a currently popular application of such algorithms[3].

There are some different questions we can ask, for example where facilities should be located, how many facilities are required. We can then apply many different constraints on the problem, such as differing costs, exclusive allocation, capacities, dynamic clients, etc. We can also have different location models, from planar models which we consider as a continuous optimization problem, to network models where there are a finite number of locations with differing costs.

Sometimes, these problems are easy, but other times they are $\mathcal{NP}$-hard, and sometimes approximate solutions are sufficient as the input data is estimated anyways.

# 3 Next week

As per Dr. Tamir, the last lecture of the semester will only be a review session.

A homework will be published over the weekend

# References

[1] Michael L Pinedo. *Scheduling: theory, algorithms, and systems.* Springer, 2016.

[3] Considerations for antennas are more complex than we can cover in this course, as they service asymmetric areas, frequency reuse, non-convex areas, etc.