Lecture 28: Scheduling

• Basic scheduling problems: open shop, job shop, flow job
• The disjunctive graph representation
• Algorithms for solving the job shop problem
• Computational complexity of the job shop problem
Open shop, job shop, flow shop scheduling

- Perform certain jobs, each consisting of operations.

- Each operation can be performed on one of machines. Operations have a duration (an integer). Each machine can handle one operation at a time.

- Objective: schedule operations so that
  1. time consumption is less than or equal some constant $T$, or
  2. time consumption is smallest possible.
Open shop, job shop, flow shop scheduling

1. Open shop: no ordering constraints on operations

2. Job shop: Operations of a job totally ordered

3. Flow shop: in each job exactly one operation for every machine, all jobs go through all the machines in the same order

*Preemptive* scheduling: no operation may be interrupted when it has already been started.
Related problems

• Planning.

• Others:
  – Course scheduling for schools (lecture halls, lecturers)
  – Timetabling for railways
  – Crew scheduling for airlines/railways etc.
  – Flight timetabling for airlines
  – Fleet assignment for airlines
Formalization of job shop scheduling

A problem instance $P = \langle M, O, J \rangle$ in job shop scheduling consists of

- a set $M$ of machines,
- a set $O$ of operations $o$, each associated with a machine $m(o) \in M$ and having a duration $d(o) \in \mathbb{N}$, and
- a set $J$ of jobs $\langle o_1, \ldots, o_n \rangle$ (each operation has exactly one occurrence.)
Formalization of job shop scheduling

A schedule $S$ for $P$ assigns to every operation $o$ a time $b(o)$:

1. $b(o) \geq 0$ for all $o \in O$

2. $b(o) \geq b(o') + d(o')$ for operations $o'$ preceding $o$ in the same job

3. $b(o) \geq b(o') + d(o')$ or $b(o') \geq b(o) + d(o)$ for all $o' \in O$ with $m(o') = m(o)$ and $o \neq o'$

Schedule $S$ has cost $T$ if $b(o) + d(o) \leq T$ for all $o \in O$. 
The ("disjunctive") graph representation
The dotted edges indicate that two operations are on the same machine, and one of the operations has to precede the other.

Finding a schedule proceeds as follows.

1. Assign a direction to every edge without introducing cycles.

2. Topologically sort the graph (total order.)

3. Assign starting and ending times to the operations.

The topologically sorted graph determines the earliest possible starting and ending times of all operations uniquely.
One schedule for the problem instance
The ordering in the schedule

We draw the graph so that all edges go from left to right:
Assignment of time points to the schedule

Given the ordering of operations, assign all the operations the earliest possible time points (here all operations have duration one):
Assignment of time points to the schedule

Obviously, the preceding schedule is not the best possible. E.g. the following is much better.
Algorithms for scheduling

There are two main approaches to finding schedules:

1. branch and bound: systematic binary search in the space of all schedules,

2. local search: schedule is gradually improved.

Both can be used with different schedule representations:

1. the disjunctive graphs (most popular representation), or

2. assignments of time points/intervals to operations.
Algorithms for scheduling: branch and bound

- Labels of search tree nodes are $x_1 x_2 \ldots x_n$, with $x_i \in \{0, 1, ?\}$ representing the undirected edges. ($\to$, $\leftarrow$, undecided.)

- One child assigns 0 to $x_i$ and the other assigns 1.

- Search tree is pruned by computing lower bounds on the cost.

- If the graph becomes cyclic or lower bound exceeds cost of the best schedule so far, prune the subtree.

- When all $x_i$ have value 0 or 1, we have found a schedule.
Branch and bound: an example

```
 | lb = 3 |
-|---|---|
???0? lb = 4 | ???1? lb = 5

-|---|---|
?1?0? | ?0?0? | ???11 lb = 7 | ??10

-|---|---|---|
11000: schedule | 00101: schedule | PRUNE

cost = 7 | cost = 6
```

Jussi Rintanen