Motivation

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 ≤ T for some constant T, or
  2. time consumption is smallest possible.

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

Definition

A problem instance $P = (M, O, J)$ 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 $d(o) \in N$, and
- a set $J$ of jobs $(o_1, \ldots, o_n)$ (each operation has exactly one occurrence.)

The ("disjunctive") graph representation

Schedule $S$ has cost $T$ if $b(o) + d(o) \leq T$ for all $o \in O$. 

Definition

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 $o', o \in O$ with $m(o') = m(o)$ and $o \neq o'$

The dotted edges indicate that two operations are on the same machine, and one of the operations has to precede the other.
The ("disjunctive") graph representation

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.

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, 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, (\( \rightarrow, \leftarrow, \) 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.
Lower bounds of schedule cost

Given a disjunctive graph, define for an operation $o$

- $\text{head}(o)$: time necessarily needed before processing $o$
- $\text{tail}(o)$: time necessarily needed after processing $o$

Highest duration of a directed path that ends in $o$

Highest duration of a directed path that starts from $o$

Define for a set $S$ of operations

- the shortest head $H(S) = \min_{o \in S} \text{head}(o)$
- the shortest tail $T(S) = \min_{o \in S} \text{tail}(o)$
- the sum of processing times $P(S) = \sum_{o \in S} d(o)$

Let $O_m$ be the set of operations on machine $m$.

$$\max_{m \in M} \left( \max_{S \subseteq O_m} H(S) + P(S) + T(S) \right)$$

In other words, we compute the lower bounds on all sets $S$ of operations that are computed on one machine.

Algorithms for scheduling: local search

Idea: two schedules are neighbors if one can be obtained from the other by a small modification (to its graph).

Modifications:

- reverse an arrow, or
- reorder consecutive operations in the graph (preserving their locations in their respective jobs.)

Modifications must preserve acyclicity.

Finding good schedules proceeds as follows:

1. Start from a randomly chosen schedule.
2. Go from the current schedule to a neighboring schedule (if the neighboring schedule is sufficiently good.)
3. Algorithms: simulated annealing, tabu search, ...

Optimal solutions for job shop scheduling can be found polynomial time if

- number of jobs is 2,
- number of machines is 2, all jobs have 1 or 2 operations, or
- number of machines is 2, all operations have duration 1.

In all cases the problem obtained by incrementing the number of machines, jobs, operations or durations by 1, is NP-hard.

Approximability of job shop scheduling

Theorem (Williamson et al. 1993)

Deciding if there is a schedule of length 4 is NP-complete.

Corollary

There is no polynomial-time algorithm that finds schedules of length $\leq \frac{1}{2}$ from optimal (unless $P=NP$)

Proof sketch.

A schedule of length 4 exists if and only if $p$-approximation algorithm with $p < \frac{1}{2}$ finds a schedule of length 4. (Schedule of length 5 would be more than $p$ from the optimal.)

Theorem (Shmoys et al. 1994)

There is a poly-time algorithm that produces schedules of length $\log^4 m$ times the optimal.