Minimum Makespan Scheduling

- Minimum makespan scheduling: Definition and variants
- Factor 2 algorithm for identical machines
- PTAS for identical machines
- Factor 2 algorithm for unrelated machines

Martin Zachariasen, DIKU

December 4, 2006

Minimum Makespan Scheduling

General framework considered:

- We are given a set of jobs and a set of machines.
- The jobs have either identical or different processing times on the given machines.
- The task is to assign jobs to machines so that the completion time, also called the *makespan* is minimized. (We may also say that we minimize the maximum total processing time on any machine.)
- The order in which the jobs are processed on a particular machine does not matter, and we may assume that they are completely “packed”.

Other variants:

- Jobs have precedence constraints.
- There may be setup times for different types of jobs.
- There may be due and release times for some jobs.

Minimum makespan scheduling on identical machines

Given a set of n jobs with processing times $p_i \in \mathbb{Z}^+$, $i = 1, \ldots, n$, and a positive integer m.

Find an assignment of the jobs to m identical machines such that the makespan is minimized.

Minimum makespan scheduling on unrelated machines

Given a set J of n jobs and a set M of m machines. The processing time for a job $j \in J$ on machine $i \in M$ is $p_{ij} \in \mathbb{Z}^+$.

Find an assignment of the jobs J to the machines M such that the makespan is minimized.

Algorithm 10.2 (Minimum makespan, identical machines)

1. Order the jobs arbitrarily.
2. Schedule jobs on machines in this order: schedule the next job on the machine that has been assigned the least amount of work so far.

◇ ◇ ◇

Theorem
Algorithm 10.2 achieves an approximation guarantee of 2 for the minimum makespan scheduling problem on identical machines.

Proof
Two lower bounds on OPT: $\frac{1}{m}\sum_i p_i$ and $\max_i\{p_i\}$

Let j be the index of a job with maximum completion time. The starting time of this job is at most

$$\frac{1}{m}\sum_i p_i \leq \text{OPT}$$

Since $p_j \leq$ OPT we get that the processing times on all machines is bounded by

$$\frac{1}{m}\sum_i p_i + \max_i\{p_i\} \leq 2 \cdot \text{OPT}.$$

Tight example
m^2 jobs with unit processing times followed by one job with processing time m.

PTAS for identical machines

Reduction to bin packing where $I = \{p_1, \ldots, p_n\}$ are the sizes of the objects that are to be packed.

bins(I, t) = Minimum number of bins of size t that are required.

Minimum makespan: $\min\{t : \text{bins}(I, t) \leq m\}$

Perform binary search for makespan t over interval $[\text{LB}, 2 \cdot \text{LB}]$, where

$$\text{LB} = \max\{\frac{1}{m}\sum_i p_i, \max_i\{p_i\}\}$$

Problem: Bin packing also NP-hard.

Bin packing with fixed number of object sizes

Assume that we have k different object sizes for a given bin size t.

Bin packing problem specified by k-tuple $(i_1, \ldots, i_k)$.

BINS$(i_1, \ldots, i_k)$ = Minimum number of bins that are required.

For a given instance $(n_1, \ldots, n_k)$, $\sum_{j=1}^{k} n_j = n$ let

$$\mathcal{K} = \{(i_1, \ldots, i_k) \mid 0 \le i_j \le n_j,\ j = 1, \ldots, k\}$$

$$\mathcal{Q} = \{(q_1, \ldots, q_k) \in \mathcal{K} \mid \text{BINS}(q_1, \ldots, q_k) = 1\}$$

First we find $\mathcal{Q}$ and then we determine the number of required bins for all k-tuples in $\mathcal{K}$ via dynamic programming:

$$\text{BINS}(i_1, \ldots, i_k) = 1 + \min_{q \in \mathcal{Q}} \text{BINS}(i_1 - q_1, \ldots, i_k - q_k)$$

Running time: $O(n^{2k})$.

Core algorithm

Let $0 < \epsilon < 1$ and $t \in [\mathrm{LB}, 2 \cdot \mathrm{LB}]$.

1. Discard *small* objects of size less than ϵt.

2. Round remaining objects:
   $p_j \in [t\epsilon(1+\epsilon)^i, t\epsilon(1+\epsilon)^{i+1}[\quad \rightarrow \quad p'_j = t\epsilon(1+\epsilon)^i$.

   At most $k = \lceil \log_{1+\epsilon} \frac{1}{\epsilon} \rceil$ different object sizes.

3. Find *optimal* solution to resulting problem in $O(n^{2k})$ time.

4. Increase bin sizes to $t(1+\epsilon)$ and increase objects to original size — gives a valid packing.

5. Fill up with small objects. Resulting number of bins denoted by $\alpha(I, t, \epsilon)$.

Proof of approximation guarantee

Lemma

$$\alpha(I, t, \epsilon) \le \text{bins}(I, t)$$

Corollary

$$t_\alpha^* = \min\{t : \alpha(I, t, \epsilon) \le m\} \le \text{OPT}$$

Assume that we perform a binary search to determine t_α^* within an interval $[T - \epsilon \cdot \text{LB}, T]$. Can be done in $\lceil \log_2 \frac{1}{\epsilon} \rceil$ iterations of the core algorithm.

Lemma

$$T \le (1 + \epsilon) \cdot \text{OPT}$$

Proof

$$T \le t_\alpha^* + \epsilon \cdot \text{LB} \le (1 + \epsilon) \cdot \text{OPT}$$

Theorem
The algorithm produces a valid schedule having makespan at most

$$T \cdot (1 + \epsilon) \le (1 + \epsilon)^2 \cdot \text{OPT} \le (1 + 3\epsilon)\text{OPT}$$

Factor 2 algorithm for unrelated machines

Integer program formulation:

$$\begin{aligned} \text{minimize } \ & t \\ \text{subject to } \ & \sum_{i \in M} x_{ij} = 1, && j \in J \\ & \sum_{j \in J} x_{ij} p_{ij} \le t, && i \in M \\ & x_{ij} \in \{0, 1\}, && i \in M,\ j \in J \end{aligned}$$

Problem: LP-relaxation has *unbounded* integrality gap (e.g., one single job of length m on m machines).

Solution: *Guess* a lower bound $T \in \mathbb{Z}^+$ on the optimal makespan.

Set $S_T = \{(i,j) \mid p_{ij} \le T\}$ and define LP(T) as the following feasibility problem:

$$\begin{aligned} \sum_{i:(i,j) \in S_T} x_{ij} &= 1, && j \in J \\ \sum_{j:(i,j) \in S_T} x_{ij} p_{ij} &\le T, && i \in M \\ x_{ij} &\ge 0, && (i,j) \in S_T \end{aligned}$$

Properties of extreme point solutions to LP(T)

Lemma
Any extreme point solution to LP(T) has at most $n + m$ nonzero variables.

Proof
Let $r = |S_T|$. At least $r - (n + m)$ of the $x_{ij} \geq 0$ inequalities must be set to equality. Thus at most $n + m$ variables are nonzero.

Corollary
Any extreme point solution must set at least $n - m$ jobs integrally.

Proof
Let α be number of integrally set jobs and β be the number of fractionally set jobs.

Since $\alpha + \beta = n$ and $\alpha + 2\beta \leq n + m$ we get $\alpha \geq n - m$.

Algorithm 17.5 (Minimum makespan, unrelated machines)

1. Construct any greedy schedule having makespan α.
2. By a binary search in the interval $[\alpha/m, \alpha]$, find the smallest value T^* for which LP(T^*) has a feasible solution.
3. Find an extreme point solution, say $\mathbf{x}$, to LP(T^*).
4. Assign all integrally set jobs to machines as in $\mathbf{x}$.
5. Construct fractional support graph H and find a perfect matching in it.
6. Assign fractionally set jobs according to the matching in H.

◇ ◇ ◇

For an extreme point solution $\mathbf{x}$ we define the bipartite support graph $G = (J, M, E)$, where $(j, i) \in E$ iff $x_{ij} \neq 0$.

Fractional support graph H is induced from G by jobs being fractionally set.

Proof of approximation guarantee

Pseudo-forest: Graph for which each connected component has at most as many edges as vertices.

Lemma
Graph G is a pseudo-forest.

Lemma
Graph H has a perfect matching.

Theorem
Algorithm 17.5 achieves an approximation guarantee of 2 for the problem of scheduling on unrelated machines.

Proof
Clearly $T^* \leq$ OPT. At most one fractional job is scheduled on each machine (we use a matching); since the processing time of each fractional job is bounded by T^*, the resulting makespan is at most $2 \cdot T^* \leq 2 \cdot$ OPT.