Makespan Scheduling

Lance Ge

Nanjing University

May 11, 2017

Section 1

Formalism

Formalism of an Optimization Problem

Definition 2.3.2.2. *An* **optimization problem** *is a 7-tuple* $U = (\Sigma_I, \Sigma_O, L, L_I, \mathcal{M}, cost, goal)$*, where*

(i) Σ_I *is an alphabet, called the* **input alphabet** *of* U,
(ii) Σ_O *is an alphabet, called the* **output alphabet** *of* U,
(iii) $L \subseteq \Sigma_I^*$ *is the* **language of feasible problem instances**,
(iv) $L_I \subseteq L$ *is the* **language of the (actual) problem instances of** U,
(v) $\mathcal{M}$ *is a function from* L *to* $Pot(\Sigma_O^*)$,[30] *and, for every* $x \in L$, $\mathcal{M}(x)$ *is called the* **set of feasible solutions** *for* x,
(vi) $cost$ *is the* **cost function** *that, for every pair* (u, x)*, where* $u \in \mathcal{M}(x)$ *for some* $x \in L$*, assigns a positive real number* $cost(u, x)$,
(vii) $goal \in \{minimum, maximum\}$.

A Mathematical Description

Makespan Scheduling Problem (MS)

Input: Positive integers $p_1, p_2, \ldots, p_n$ and an integer $m \geq 2$ for some $n \in \mathbb{N} - \{0\}$.
$\{p_i$ is the processing time of the ith job on any of the m available machines$\}$.

Constraints: For every input instance $(p_1, \ldots, p_n, m)$ of MS,
$\mathcal{M}(p_1, \ldots, p_n, m) = \{S_1, S_2, \ldots, S_m \mid S_i \subseteq \{1, 2, \ldots, n\}$ for $i = 1, \ldots, m$, $\bigcup_{k=1}^{m} S_k = \{1, 2, \ldots, n\}$, and $S_i \cap S_j = \emptyset$ for $i \neq j\}$.
$\{\mathcal{M}(p_1, \ldots, p_n, m)$ contains all partitions of $\{1, 2, \ldots, n\}$ into m subsets. The meaning of $(S_1, S_2, \ldots, S_m)$ is that, for $i = 1, \ldots, m$, the jobs with indices from S_i have to be processed on the ith machine$\}$.

Costs: For each $(S_1, S_2, \ldots, S_m) \in \mathcal{M}(p_1, \ldots, p_n, m)$,
$cost((S_1, \ldots, S_m), (p_1, \ldots, p_n, m)) = \max\left\{\sum_{l \in S_i} p_l \mid i = 1, \ldots, m\right\}$.

Goal: *minimum*.

Subsection 1

Input

Input

- $\Sigma_I := \{0, 1, \#\}$
- A *number* is a word of $\{0, 1\}$.
  - 1
  - 101
- An *instance* is a sequence of numbers with '#'s spliting them.
  - A number is an instance.
  - If A, B are two instances, then $A\#B$ is an instance.
  - There is no other way to obtain an instance.
- L is the set of all instances. L_I is a subset of L. ($L_I = L$ generates the generalized problem.)

Subsection 2

Output

Output

- $\Sigma_O := \{0, 1, \&, \#\}$
- An *load* is a sequence of numbers with '&'s spliting them.
  - A number is a load.
  - If A, B are two loads, then $A\&B$ is a load.
  - There is no other way to obtain a load.
- An *schedule* is a sequence of loads with '#'s spliting them.
  - A load is a schedule.
  - If A, B are two schedules, then $A\#B$ is a schedule.
  - There is no other way to obtain a schedule.
- $\mathcal{M}(x)$ is the set of all schedules s that if x contains n '#'s and the last number is m, then s contains $m - 1$ '#'s and its numbers are 1 to n.

cost

- $cost(s)$ is the maximum value of sums of numbers in each set of s.

Section 2

The (Sorted) Greedy Approach

Description

Algorithm 4.2.1.3 (GMS (Greedy Makespan Schedule)).

Input: $I = (p_1, \ldots, p_n, m)$, n, m, $p_1, \ldots, p_n$ positive integers and $m \geq 2$.

Step 1: Sort $p_1, \ldots, p_n$.
To simplify the notation we assume $p_1 \geq p_2 \geq \cdots \geq p_n$ in the rest of the algorithm.

Step 2: **for** $i = 1$ **to** m **do**
begin $T_i := \{i\}$;
$Time(T_i) := p_i$
end
{In the initialization step the m largest jobs are distributed to the m machines. At the end, T_i should contain the indices of all jobs assigned to the ith machine for $i = 1, \ldots, m$.}

Step 3: **for** $i = m + 1$ **to** n **do**
begin compute an l such that
$Time(T_l) := \min\{Time(T_j) | 1 \leq j \leq m\}$;
$T_l := T_l \cup \{i\}$;
$Time(T_l) := Time(T_l) + p_i$
end

Output: $(T_1, T_2, \ldots, T_m)$.

Subsection 1

Why is it not optimal?

A Lower Bound

- In fact, (sorted) GMS is proved to be $\frac{4}{3}$-approximation by R. L. Graham in 1969. Consider the following instance (See *Bounds on Multiprocessing Timing Anomalies* for details):
- $$I = (2m-1, 2m-1, 2m-2, 2m-2, \cdots, m+1, m+1, m, m, m)$$

  [2]
- $$R_{GMS}(I) = \frac{4}{3} - \frac{1}{3m}$$

Subsection 2

A Tight Bound

- In fact, $\frac{4}{3}$ is not only a lower bound, but also an upper bound.
- Unfortunately, I haven't go through the proof. If anyone is interested, see [Graham, 1969].

Section 3

Acknowledgements

J. Hromkovič.
Algorithmics for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics.
Texts in Theoretical Computer Science. An EATCS Series.
Springer Berlin Heidelberg, 2013.

Ronald L. Graham.
Bounds on multiprocessing timing anomalies.
SIAM journal on Applied Mathematics, 17(2):416–429, 1969.