Approximation Algorithms

Hengfeng Wei

hfwei@nju.edu.cn

May 14 ∼ May 18, 2017

Approximation Algorithms

Approximation Classes

NPO: NP optimization problems

APX: constant factor ϵ-approximation $(\epsilon > 1)$

$f(n)$-APX: Exp-APX, Poly-APX, Log-APX

PTAS:
- $\forall \epsilon > 0 : (1+\epsilon)$-approximation
- P : Poly(n) $\quad O(n^{2/\epsilon})$ $\quad O(n^{2^{2^{1/\epsilon}}})$

FPTAS:
- $\forall \epsilon > 0 : (1+\epsilon)$-approximation
- FP : Poly$(n, 1/\epsilon)$ $\quad O((1/\epsilon)^2 \cdot n^3)$

PO: polynomial time solvable optimization problems

Approximation Classes

$$\begin{aligned} \mathrm{PO} &\subsetneq \mathrm{FPTAS} \subsetneq \mathrm{PTAS} \subsetneq \mathrm{APX} \\ &\subsetneq \text{Log-APX} \subsetneq \text{Poly-APX} \subsetneq \text{Exp-APX} \subsetneq \mathrm{NPO} \end{aligned}$$

$$(\text{if } \mathrm{P} \neq \mathrm{NP})$$

- Knapsack, Makespan, Vertex Cover, Set Cover, Clique, TSP

Stability of approximation

Stability of approximation

Approximation Algorithms

Makespan scheduling problem

Makespan scheduling problem (MS)

- n jobs: $J_1, \ldots, J_n$
- processing time: $p_1, \ldots, p_n$
- $m \geq 2$ machines: $M_1, \ldots, M_m$
- goal: minimize the makespan

MS is NP-complete

Definition (Partition)

$$A = \{5, 1, 3, 4, 8, 2, 7\} \implies A' = \{5, 3, 7\} \quad A - A' = \{1, 2, 4, 8\}$$

Definition (3-Partition)

$$A = \{3, 3, 2, 2, 2, 2\}, m = 2, B = 7$$

List-Scheduling (LS) algorithm

List-Scheduling algorithm (JH 4.2.1.4)

- online
- assign job to the least heavily loaded

$$J = 2, 3, 4, 6, 2, 2 \qquad m = 3$$

LS is 2-approx.

$$T \textit{ vs. } T^* : \frac{T}{T^*}$$

$$T^* \geq \frac{1}{m} \sum_j t_j$$

$$T^* \geq \max_j t_j$$

1. M_i: with the maximum load
2. J_i: the last job on M_i

$$m s_i \leq \sum_j t_j \implies s_i \leq \frac{1}{m} \sum_j t_j \leq T^*$$

$$T = c_i = s_i + p_i \leq T^* + T^* = 2T^*$$

2-approx. is (almost) tight

$$n = \underbrace{m(m-1)}_{p_i=1} + \underbrace{1}_{p_i=m}$$

$$\frac{T}{T^*} = \frac{2m-1}{m} = 2 - \frac{1}{m}$$

LS is $(2 - \frac{1}{m})$-approx.

$$
\begin{aligned}
m s_i &\le \sum_{j \neq i} p_j = \frac{1}{m}\left(\sum_j p_j - p_i\right) \\
&= \frac{1}{m} \sum_j p_j - \frac{1}{m} p_i \\
&\le T^* - \frac{1}{m} p_i
\end{aligned}
$$

$$
\begin{aligned}
T = c_i &= s_i + p_i \\
&\le T^* + \left(1 - \frac{1}{m}\right) p_i
\end{aligned}
$$

Sorting-Scheduling algorithm

Sorting-Scheduling algorithm (JH 4.2.1.5)

Longest Processing Time (LPT) rule:

- sorting non-increasingly
- applying LS

1. M_i: with the maximum load
2. J_i: the last job on M_i

$$|M_i| = 1 \implies T = T^*$$

$$\begin{aligned} |M_i| \geq 2 &\implies p_i \leq \frac{1}{2}T^* \\ &\implies T = s_i + p_i \leq (\frac{3}{2} - \frac{1}{2m})T^* \end{aligned}$$

LPT rule is $(\frac{4}{3} - \frac{1}{3m})$-approx.

$$p_1 \geq p_2 \geq \cdots \geq p_n$$

CASE $p_i \leq \frac{1}{3}T^*$:

$$T \leq (\frac{4}{3} - \frac{1}{3m})T^*$$

CASE $p_i > \frac{1}{3}T^*$:

$$p_i \equiv p_n (\text{w.l.o.g: } T \text{ unchanged; } T^* \text{ not smaller})$$

$$\implies p_1 \geq p_2 \geq \cdots \geq p_n > \frac{1}{3}T^* \implies |M_i| \leq 2$$

$$\implies n \leq 2m \implies n = 2m - h \xrightarrow[\text{argument}]{\text{exchange}} T = T^*$$

$(\frac{4}{3} - \frac{1}{3m})$-approx. is tight

$$n = 2m + 1$$

$$\begin{aligned} \text{LPT: } & J_1 = x, J_{2m} = y, J_{2m+1} = z \\ \text{OPT: } & J_{2m-1} = J_{2m} = J_{2m+1} = y \\ \textit{vs. } & \frac{x + 2y}{3y} = \frac{4}{3} - \frac{1}{3m} \end{aligned}$$

$$J = \{2m-1, 2m-1, \ldots, m+1, m+1, m, m, m\}$$

Reference

- "Bounds on Multiprocessing Timing Anomalies" by R.L.Graham, 1969
- "Approximation Algorithms for NP-Hard Problems" edited by Dorit Hochbaum, 1996 (Theorem 1.5)


THANK YOU