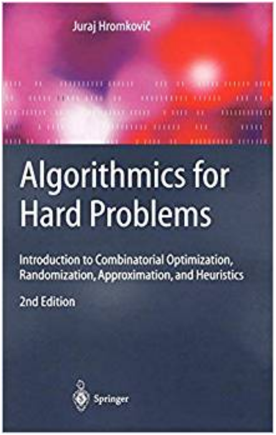
4-12 Approximation Algorithms

Hengfeng Wei

hfwei@nju.edu.cn

June 03, 2019


Juraj Hromkovič
Algorithmics for Hard Problems
Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics
2nd Edition
Springer

2条亚马逊美国的评论

Following the notion of approximability we divide the class NPO of optimization problems into the following five subclasses:

NPO(I): Contains every optimization problem from NPO for which there exists a FPTAS.
{In Section 4.3 we show that the knapsack problem belongs to this class.}

NPO(II): Contains every optimization problem from NPO that has a PTAS.
{In Section 4.3.4 we show that the makespan scheduling problem belongs to this class.}

NPO(III): Contains every optimization problem $U \in$ NPO such that
(i) there is a polynomial-time δ-approximation algorithm for some $\delta > 1$, and
(ii) there is no polynomial-time d-approximation algorithm for U for some $d < \delta$ (possibly under some reasonable assumption like P $\neq$ NP), i.e., there is no PTAS for U.
{The minimum vertex cover problem, MAX-SAT, and $\triangle$-TSP are examples of members of this class.}

NPO(IV): Contains every $U \in NPO$ such that
(i) there is a polynomial-time $f(n)$-approximation algorithm for U for some $f : \mathbb{N} \to \mathbb{R}^+$, where f is bounded by a polylogarithmic function, and
(ii) under some reasonable assumption like P $\neq$ NP, there does not exist any polynomial-time δ-approximation algorithm for U for any $\delta \in \mathbb{R}^+$.
{The set cover problem belongs to this class.}

NPO(V): Contains every $U \in$ NPO such that if there exists a polynomial-time $f(n)$-approximation algorithm for U, then (under some reasonable assumption like P $\neq$ NP) $f(n)$ is not bounded by any polylogarithmic function.
{TSP and the maximum clique problem are well-known members of this class.}

Definition (NPO: NP Optimization)

$$\Pi = (L, sol, cost, goal)$$

Definition (NPO: NP Optimization)

$$\Pi = (L, sol, cost, goal)$$

$$\begin{aligned}
L :\ & l \in L \text{ is an instance} \\
& \textit{decidable in poly. time} \\
sol :\ & x \in sol(l) \text{ is a feasible solution of } l \\
& \textit{verifiable in poly. time} \\
cost :\ & cost(x) \text{ is the cost of } x \\
& \textit{computable in poly. time} \\
goal :\ & goal \in \{\min, \max\}
\end{aligned}$$

$f(n)$-APX: $f(n)$-approximation

$$
\begin{aligned}
\text{Exp-APX:} &\quad f(n) = O(2^{n^c}) \\
\text{Poly-APX:} &\quad f(n) = O(n^c) \\
\text{Log-APX:} &\quad f(n) = O(\log n) \\
\text{APX:} &\quad f(n) = c \; (c > 1)
\end{aligned}
$$

$f(n)$-APX: $f(n)$-approximation

Exp-APX: $f(n) = O(2^{n^c})$

Poly-APX: $f(n) = O(n^c)$

Log-APX: $f(n) = O(\log n)$

APX: $f(n) = c \; (c > 1)$

PTAS: Poly. time approximation scheme

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

FPTAS: Fully poly. time approximation scheme

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

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

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

(if $P \neq NP$)

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

- Knapsack $\in$ FPTAS $\setminus$ PO
- Makespan $\in$ PTAS $\setminus$ FPTAS (TODAY)
- Vertex Cover $\in$ APX $\setminus$ PTAS
- Set Cover $\in$ Log-APX $\setminus$ APX (CLRS 35.3)
- Clique $\in$ Poly-APX $\setminus$ Log-APX
- TSP $\in$ Exp-APX $\setminus$ Poly-APX

Makespan scheduling problem

Makespan scheduling problem (MS)

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

MS is NP-complete

Definition (Partition)

Instance:

$$\forall a \in A : s(a) \in \mathbb{Z}^+$$

Question: Is there a subset $A' \subseteq A$:

$$\sum_{a \in A'} s(a) = \sum_{a \in A \setminus A'} s(a)$$

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

MS is strongly NP-complete

Definition (3-Partition)

Instance:

$$|A| = 3m, B \in \mathbb{Z}^+$$
$$\forall a \in A : s(a) \in \mathbb{Z}^+, B/4 < s(a) < B/2$$

Question: Can A be partitioned into m disjoint sets $S_1, \ldots, S_m$:

$$\forall 1 \leq i \leq m : |S_i| = 3, \sum_{a \in S_i} s(a) = B$$

$$A = \{1, 2, 2, 3, 3, 4, 6, 7, 8\}, m = 3, B = 12 \implies \{1, 3, 8; 2, 4, 6; 2, 3, 7\}$$

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.

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

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

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

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.

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

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

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

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

$$|M_i| \geq 2 \implies p_i \leq \frac{1}{2}T^*$$

$$\implies T = s_i + p_i \leq (\frac{3}{2} - \frac{1}{2m})T^*$$

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

$$p_1 \ge p_2 \ge \cdots \ge p_n$$

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

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

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

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

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

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

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

$$n = 2m + 1$$

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

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

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)

PTAS for MS

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

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

1. $J = J_L \triangleq \{\text{long jobs}\} \uplus J_S \triangleq \{\text{short jobs}\}$
2. S_L: the optimal schedule for J_L
3. S: apply List-Scheduling to S_L and J_S

Reference

- "The Design of Approximation Algorithms" by David P. Williamson and David Shmoys, 2011 (Section 3.2).

PTAS for MS

1. Split J:

$$J_i \in J_S \iff p_i \le \epsilon \cdot \frac{1}{m} \sum_j p_j$$

$$\implies |J_L| < \frac{1}{\epsilon} \cdot m$$

2. Time for S_L (m being a constant!):

$$m^{\frac{1}{\epsilon} \cdot m} \cdot O(n)$$

3. Approx. ratio ($p_i \in J_S$ case):

$$\begin{aligned} T = s_i + p_i &\le \frac{1}{m} \sum_j p_j + \epsilon \cdot \frac{1}{m} \sum_j p_j \\ &= (1+\epsilon) \frac{1}{m} \sum_j p_j \\ &\le (1+\epsilon) T^* \end{aligned}$$

No FPTAS for MS

Theorem (MS $\in$ PTAS $\setminus$ FPTAS)

No FPTAS for MS.

MS is strongly NP-complete $\implies$ MS with $\max_j p_j \leq q(n)$ is NP-complete

Reference

- "The Design of Approximation Algorithms" by David P. Williamson and David Shmoys, 2011 (Section 3.2).

No FPTAS for MS

Theorem

$$\exists FPTAS \text{ for } MS \implies MS \in P.$$

$$A_\epsilon : \epsilon = \frac{1}{\lceil 2nq(n) \rceil}$$

$$\begin{aligned}(1+\epsilon)T^* &= T^* + \epsilon \cdot T^* \\ &\le T^* + \frac{1}{\lceil 2nq(n) \rceil} \cdot nq(n) \\ &\le T^* + \frac{1}{2}\end{aligned}$$

$$\text{Time: } \text{Poly}(\frac{1}{\epsilon}, n) = \text{Poly}(\lceil 2nq(n) \rceil, n) = \text{Poly}(n)$$