第8章 近似算法

8.1 近似算法及其近似比

近似算法: A是一个多项式时间算法且对组合优化问题Π的每一个实例I输出一个可行解σ. 记$A(I)=c(\sigma)$, $c(\sigma)$是σ的值

最优化算法: 恒有$A(I)=OPT(I)$, 即A总是输出 I 的最优解.

当Π是最小化问题时, 记 $r_A(I)=A(I)/OPT(I)$;

当Π是最大化问题时, 记 $r_A(I)=OPT(I)/A(I)$.

A的近似比为 r (A是r–近似算法): 对每一个实例 I, $r_A(I) \le r$.

A具有常数比: r是一个常数.

可近似性分类

假设P≠NP , NP难的组合优化问题按可近似性可分成 3 类:

(1) 完全可近似的:对任意小的$\varepsilon>0$, 存在$(1+\varepsilon)$-近似算法.

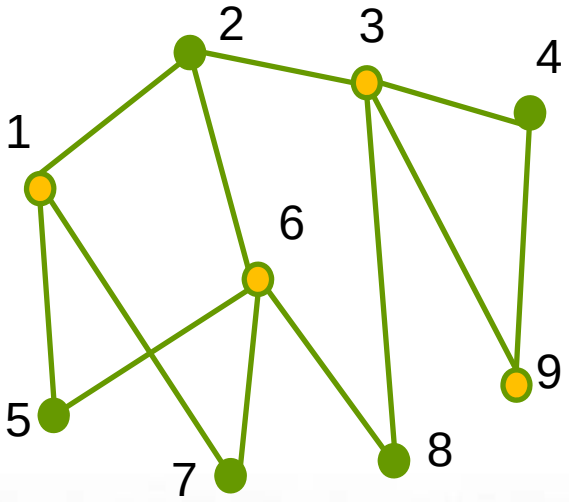
(2) 可近似的: 存在具有常数比的近似算法.

(3) 不可近似的: 不存在具有常数比的近似算法,

最小顶点覆盖问题

问题: 任给图$G=<V,E>$, 求G的顶点数最少的顶点覆盖.

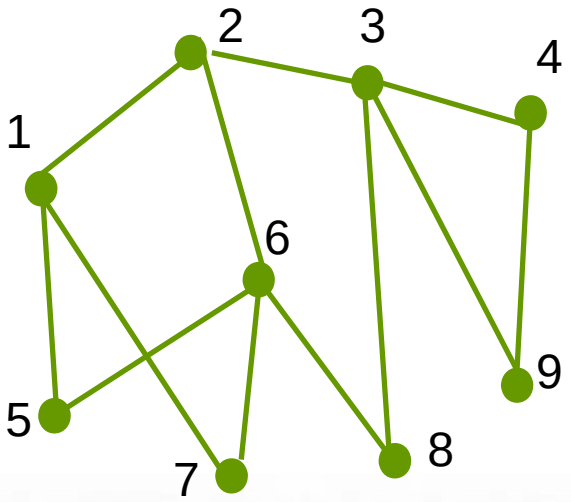
算法MVC: 开始时令$V'=\varnothing$. 任取一条边(u,v), 把u和v加入V'并删去u和v及其关联的边. 重复上述过程, 直至删去所有的边为止. V'为所求的顶点覆盖.



{1,2}

{1,2,3,4}

{1,2,3,4,5,6}



一个最优解：{1,3,6,9}， MVC的解：{1,2,3,4,5,6}

北京大学

最小顶点覆盖问题

分析: 算法时间复杂度为$O(m)$, $m=|E|$.

记 $|V'| = 2k$, V'由 k 条互不关联的边的端点组成. 为了覆盖这 k 条边需要 k 个顶点, 从而 $\mathrm{OPT}(I) \geq k$. 于是,

$$\mathrm{MVC}(I)/\mathrm{OPT}(I) \leq 2k/k = 2.$$

又, 设图G由 k 条互不关联的边构成, 显然

$$\mathrm{MVC}(I) = 2k, \quad \mathrm{OPT}(I) = k,$$

这表明 MVC 的近似比不会小于2, 上面估计的MVC的近似比已不可能再进一步改进.

近似算法的分析

研究近似算法的两个基本方面——设计算法和分析算法的运行时间与近似比.

分析近似比

(1) 关键是估计最优解的值.

(2) 构造使算法产生最坏的解的实例. 如果这个解的值与最优值的比达到或者可以任意的接近得到的近似比(这样的实例称作紧实例), 那么说明这个近似比已经是最好的、不可改进的了; 否则说明还有进一步的研究余地.

(3) 研究问题本身的可近似性, 即在**P≠NP**(或其他更强)的假设下, 该问题近似算法的近似比的下界.

8.2 多机调度问题

多机调度问题:

任给有穷的作业集 A 和 m 台相同的机器, 作业 a 的处理时间为正整数 $t(a)$,每一项作业可以在任一台机器上处理. 如何把作业分配给机器才能使完成所有作业的时间最短? 即, 如何把 A 划分成 m 个不相交的子集 A_i 使得

$$\max\left\{\sum_{a\in A_i} t(a) \mid i=1,2,\cdots,m\right\}$$

最小?

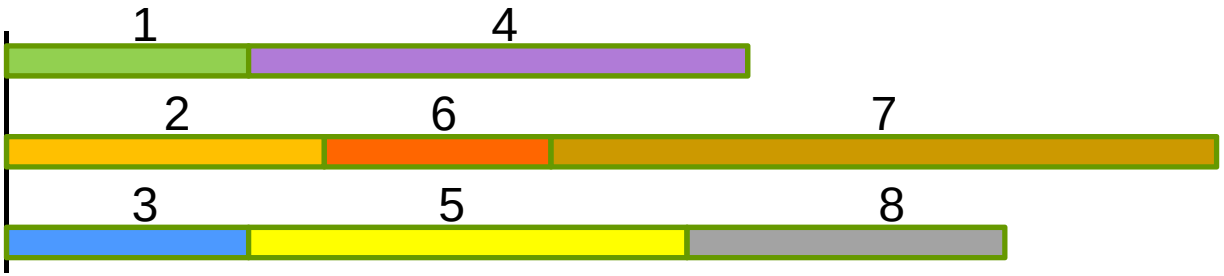
负载: 分配给一台机器的作业的处理时间之和.

贪心法 G-MPS: 按输入的顺序分配作业, 把每一项作业分配给当前负载最小的机器. 如果当前负载最小的机器有2台或 2台以上, 则分配给其中的任意一台.

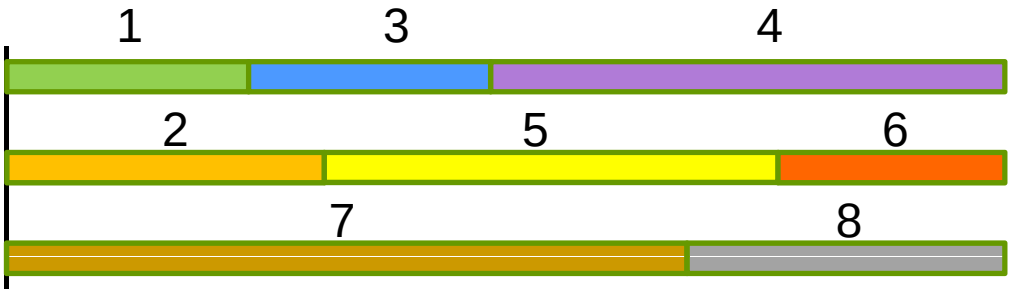
北京大学

实例

例如， 3 台机器， 8 项作业，处理时间为 3, 4, 3, 6, 5, 3, 8, 4



G-MPS的解
完成时间 15



最优解
完成时间 12

算法给出的分配方案是 {1,4}, {2,6,7}, {3,5,8},
负载分别为 3+6=9, 4+3+8=15, 3+5+4=12
最优的分配方案是 {1,3,4}, {2,5,6}, {7,8},
负载分别为 3+3+6=12, 4+5+3=12, 8+4=12

北京大学

贪心法的性能

定理8.1 对多机调度问题的每一个有 m 台机器的实例 I，

$$\mathrm{G-MPS}(I) \le \left(2-\frac{1}{m}\right)\mathrm{OPT}(I).$$

证 显然，$(1)\ \mathrm{OPT}(I) \ge \frac{1}{m}\sum_{a\in A} t(a)$，$(2)\ \mathrm{OPT}(I) \ge \max_{a\in A} t(a)$.

设机器 M_j 的负载最大，记作 $t(M_j)$. 又设 b 是最后被分配给机器 M_j 的作业. 根据算法，在考虑分配 b 时 M_j 的负载最小，故

$$t(M_j)-t(b) \le \frac{1}{m}\left(\sum_{a\in A} t(a)-t(b)\right).$$

北京大学

证 明

于是

$$\mathrm{G-MPS}(I)=t(M_j)$$

$$\le \frac{1}{m}\left(\sum_{a\in A} t(a)-t(b)\right)+t(b)$$

$$=\frac{1}{m}\sum_{a\in A} t(a)+\left(1-\frac{1}{m}\right)t(b)$$

$$\le \mathrm{OPT}(I)+\left(1-\frac{1}{m}\right)\mathrm{OPT}(I)$$

$$=\left(2-\frac{1}{m}\right)\mathrm{OPT}(I).$$

紧实例

M 台机器， $m(m-1)+1$ 项作业，

前 $m(m-1)$ 项作业的处理时间都为 1，最后一项作业的处理时间为m.

算法把前 $m(m-1)$项 作业平均地分配给 m 台机器, 每台 $m-1$ 项，最后一项任意地分配给一台机器.

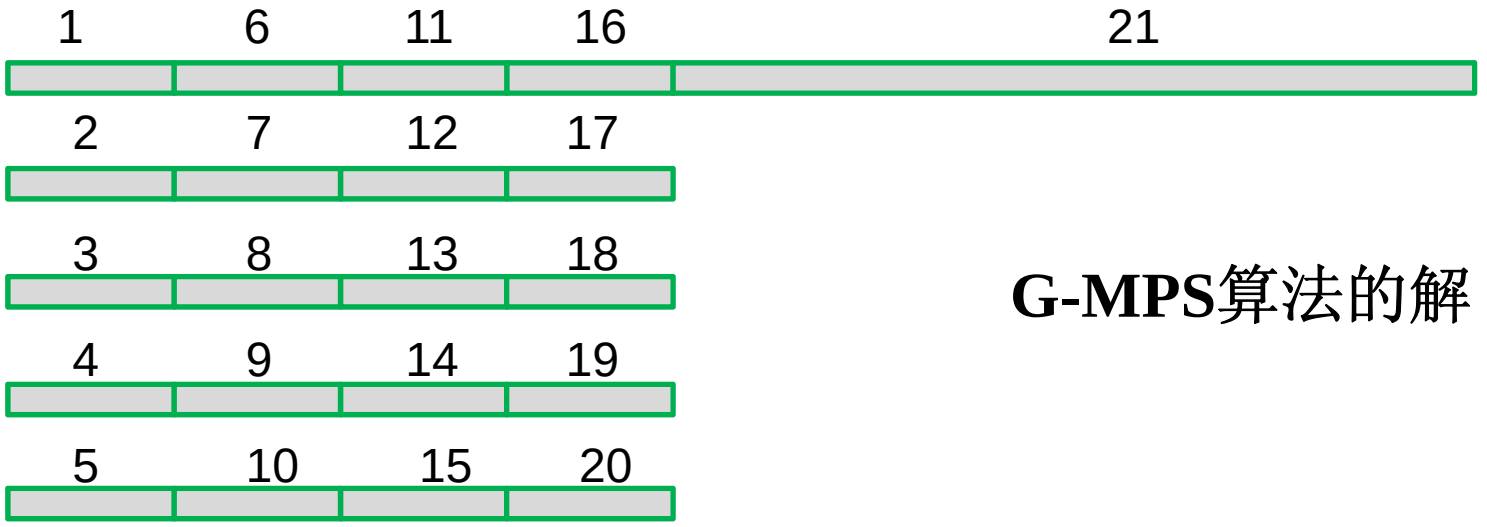
$$\text{G-MPS}(I) = 2m-1.$$

最优分配方案是把前 $m(m-1)$ 项作业平均地分配给 $m-1$ 台机器, 每台 m 项, 最后一项分配给留下的机器,

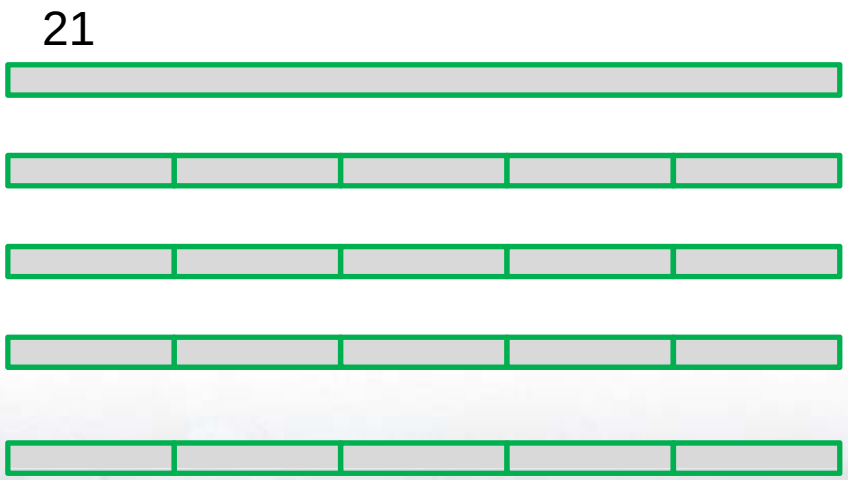
$$\text{OPT}(I) = m.$$

G-MPS是2-近似算法

m=5的紧实例



G-MPS算法的解



最优解

8.2.2 改进的贪心近似算法

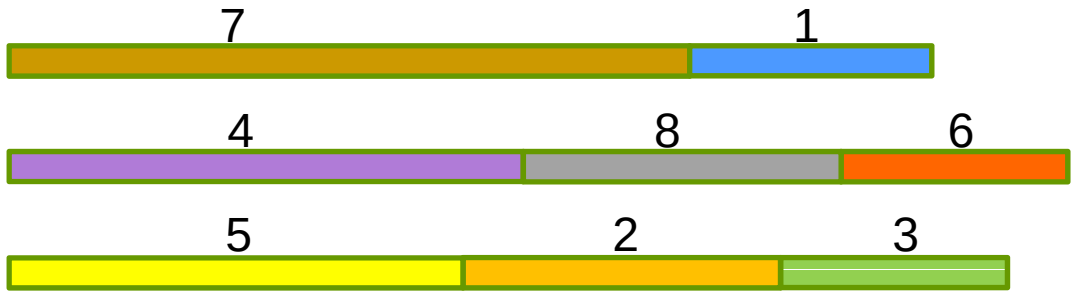
递降贪心法DG-MPS：首先按处理时间从大到小重新排列作业, 然后运用**G-MPS.**

例如对上一小节的紧实例得到最优解.

对另一个实例：先重新排序 **8, 6, 5, 4, 4, 3, 3, 3;**

3台机器的负载分别为 8+3=11, 6+4+3=13, 5+4+3=12.

比**G-MPS**的结果好.



DG-MPS的解
完成时间13

分析：**DG-MPS**增加排序时间$O(n\log n)$, 仍然是多项式时间

近似比

定理8.2 对多机调度问题的每一个有 m 台机器的实例 I,

$$\mathrm{DG-PMS}(I) \le \frac{1}{m}\left(\frac{3}{2}-\frac{1}{2m}\right)\mathrm{OPT}(I)$$

证 设作业按处理时间从大到小排列为 $a_1, a_2, \ldots, a_n$, 仍考虑负载最大的机器 M_j 和最后分配给 M_j 的作业 a_i.

(1) M_j 只有一个作业, 则 $i = 1$, 必为最优解.

(2) M_j 有 2 个或 2个以上作业, 则 $i \ge m+1$, $\mathrm{OPT}(I) \ge 2t(a_i)$

$$\mathrm{DG-MPS}(I) = t(M_j) \le \frac{1}{m}\left(\sum_{k=1}^{n} t(a_k) - t(a_i)\right) + t(a_i)$$

$$= \frac{1}{m}\sum_{k=1}^{n} t(a_k) + \left(1-\frac{1}{m}\right)t(a_i) \le \mathrm{OPT}(I) + \left(1-\frac{1}{m}\right)\cdot\frac{1}{2}\mathrm{OPT}(I)$$

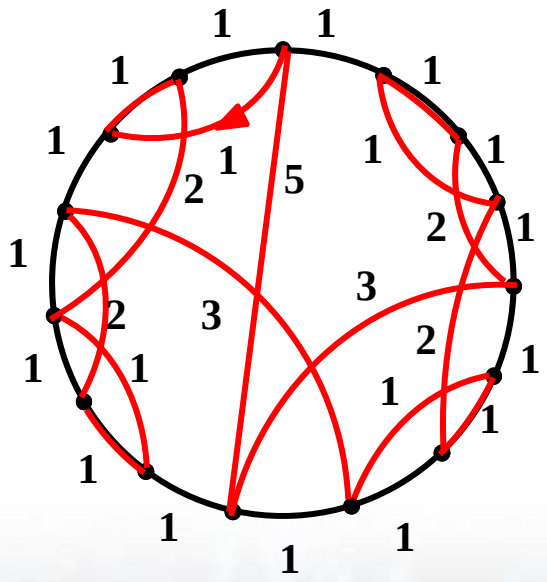
$$= \left(\frac{3}{2}-\frac{1}{2m}\right)\mathrm{OPT}(I)$$

8.3 货郎问题

本节考虑满足三角不等式的货郎问题

8.3.1 最邻近法

最邻近法NN：从任意一个城市开始，在每一步取离当前所在城市最近的尚未到过的城市作为下一个城市. 若这样的城市不止一个，则任取其中的一个. 直至走遍所有城市，最后回到开始出发的城市.



一个NN性能很坏的实例 I，

$NN(I)=21$，$OPT(I)=15$

最邻近法的性能

定理8.3 对于货郎问题所有满足三角不等式的 n 个城市的实例 I, 总有

$$\mathrm{NN}(I) \le \frac{1}{2}\left(\lceil \log_2 n \rceil + 1\right)\mathrm{OPT}(I).$$

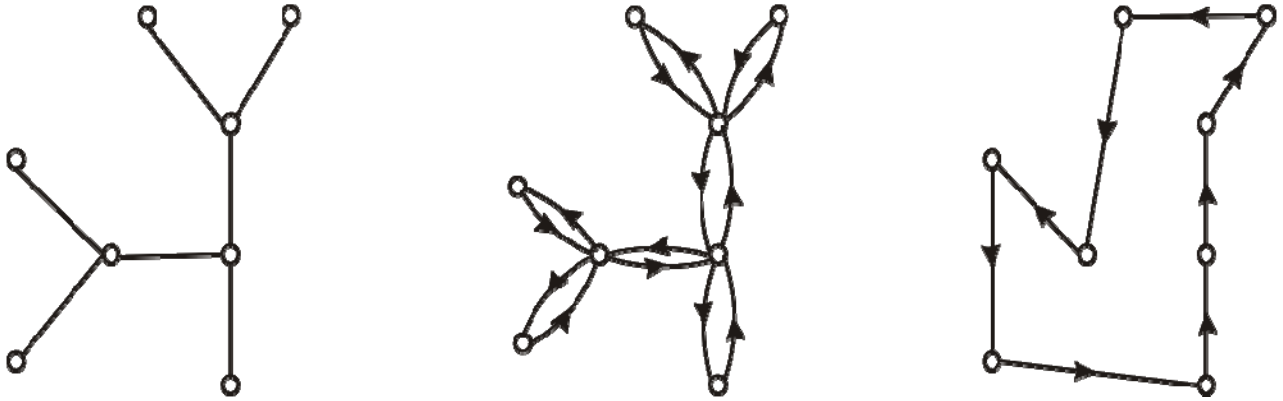
而且, 对于每一个充分大的 n, 存在满足三角不等式的 n 个城市的实例 I 使得

$$\mathrm{NN}(I) > \frac{1}{3}\left(\log_2(n+1) + \frac{4}{3}\right)\mathrm{OPT}(I).$$

8.3.2 最小生成树法

最小生成树法MST: 首先, 求图的一棵最小生成树T. 然后, 沿着 T 走两遍得到图的一条欧拉回路. 最后, 顺着这条欧拉回路, 跳过已走过的顶点, 抄近路得到一条哈密顿回路.

例



求最小生成树和欧拉回路都可以在多项式时间内完成, 故算法是多项式时间的.

最小生成树法的性能

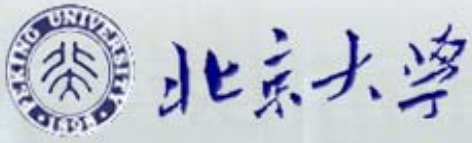
定理8.4 对货郎问题的所有满足三角不等式的实例 I,

$$\mathbf{MST}(I) < 2\mathbf{OPT}(I).$$

证　因为从哈密顿回路中删去一条边就得到一棵生成树, 故 T 的权小于$\mathbf{OPT}(I)$. 沿 T 走两遍的长小于$2\mathbf{OPT}(I)$. 因为满足三角不等式, 抄近路不会增加长度, 故

$$\mathbf{MST}(I) < 2\mathbf{OPT}(I).$$

MST是2-近似算法.

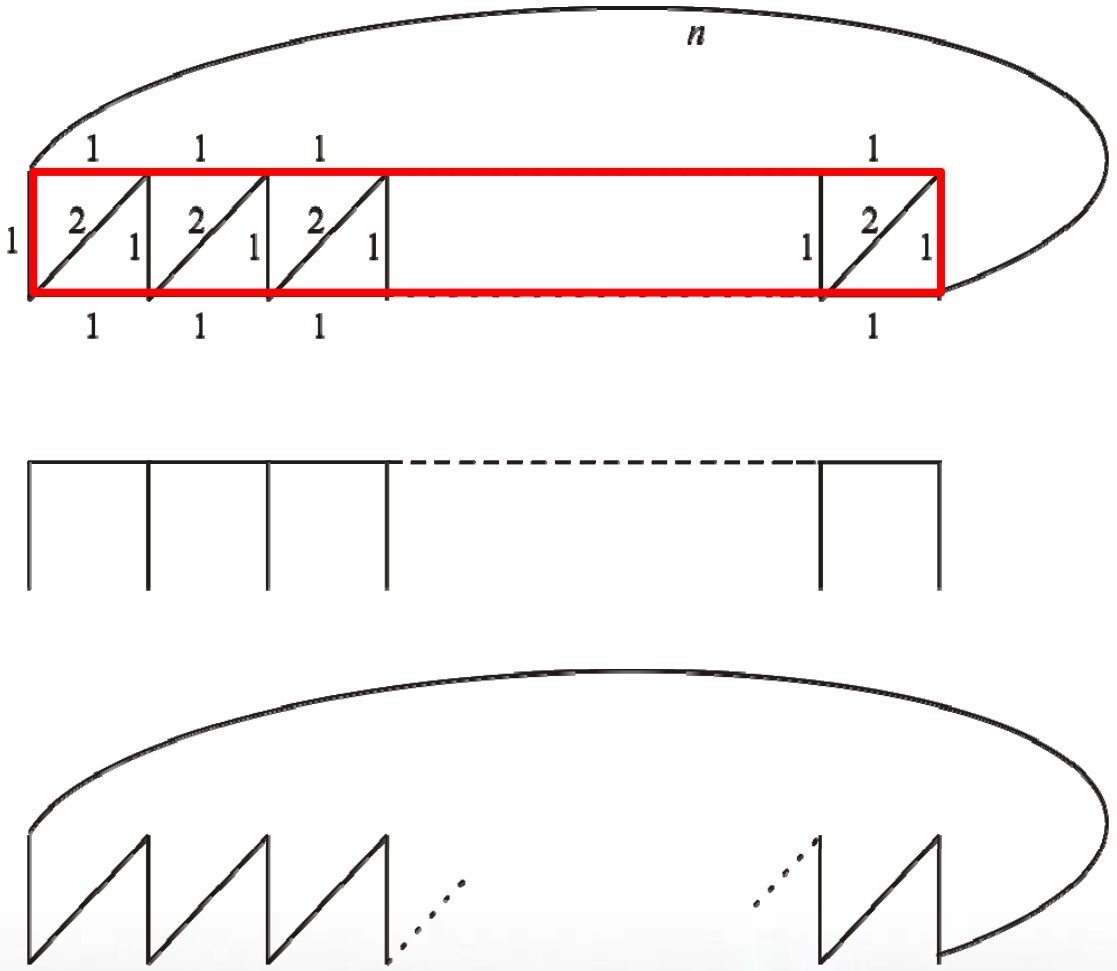


紧实例

$\mathbf{OPT}(I) = 2n$

$\mathbf{MST}(I) = 4n-2$

$$= \left(2 - \frac{1}{n}\right)\mathbf{OPT}(I)$$

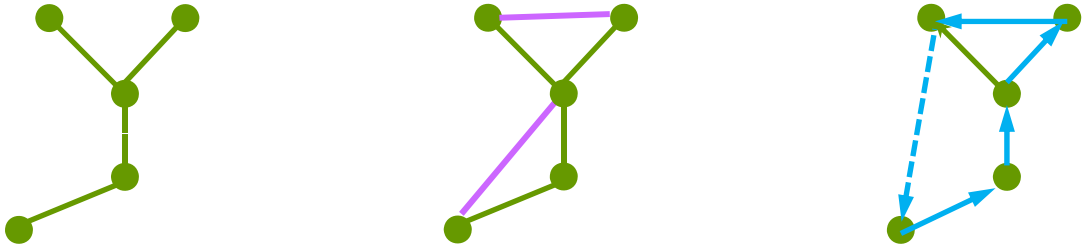


8.3.3 最小权匹配法

最小权匹配法 MM:

首先求图的一棵最小生成树T.

记 T 的所有奇度顶点在原图中的导出子图为 H, H 有偶数个顶点, 求 H 的最小匹配 M. 把 M 加入 T 得到一个欧拉图, 求这个欧拉图的欧拉回路; 最后, 沿着这条欧拉回路, 跳过已走过的顶点, 抄近路得到一条哈密顿回路.



求任意图最小权匹配的算法是多项式时间的, 因此 MM 是多项式时间的.

最小权匹配法的性能

定理8.5 对货郎问题的所有满足三角不等式的实例 I,

$$\mathrm{MM}(I) < \frac{3}{2}\mathrm{OPT}(I)$$

证 由于满足三角不等式, 导出子图 H 中的最短哈密顿回路C的长度不超过原图中最短哈密顿回路的长度 $\mathrm{OPT}(I)$. 沿着C隔一条边取一条边, 得到 H 的一个匹配. 总可以使这个匹配的权不超过 C 长的一半. 因此, H 的最小匹配 M 的权不超过 $\mathrm{OPT}(I)/2$, 求得的欧拉回路的长小于 $(3/2)\mathrm{OPT}(I)$. 抄近路不会增加长度, 得证

$$\mathrm{MM}(I) < (3/2)\mathrm{OPT}(I).$$

MM是3/2 -近似算法

北京大学

货郎问题的难度

定理8.6 货郎问题(不要求满足三角不等式) 是不可近似的, 除非 **P = NP.**

证 假设不然, 设**A**是货郎问题的近似算法, 其近似比 $r \le K$, K是常数. 任给图 $G = <V,E>$, 如下构造货郎问题的实例 I_G:

城市集V, $\forall u,v \in V$,

若 $(u,v) \in E$, 则令 $d(u,v)=1$; 否则令 $d(u,v)=Kn$, 其中 $|V| = n$. 若G 有哈密顿回路, 则

$$\mathrm{OPT}(I_G) = n, \quad \mathrm{A}(I_G) \le r\,\mathrm{OPT}(I_G) \le Kn$$

否则

$$\mathrm{OPT}(I_G) > Kn, \quad \mathrm{A}(I_G) \ge \mathrm{OPT}(I_G) > Kn$$

所以, G 有哈密顿回路当且仅当 $\mathrm{A}(I_G) \le Kn$

证明

于是, 下述算法可以判断图 G 是否有哈密顿回路:

首先构造货郎问题的实例 I_G, 然后对I_G 运用算法 A. 若 $A(I_G) \le Kn$, 则输出“Yes”; 否则输出“No”.

由于K是固定的常数, 构造 I_G 可在 $O(n^2)$ 时间内完成且 $|I_G| = O(n^2)$. A是多项式时间的, A 对 I_G 可在 n 的多项式时间内完成计算, 所以上述算法是 HC 的多项式时间算法. 而 HC 是 NP 完全的, 推得 P=NP.

8.4 背包问题

0-1背包问题的优化形式:

任给 n 件物品和一个背包，物品 i 的重量为 w_i，价值为 v_i，$1 \le i \le n$，背包的重量限制为 B，其中 w_i，v_i 以及 B 都是正整数.

把哪些物品装入背包才能在不超过重量限制的条件下使得价值最大? 即, 求子集 $S^* \subseteq \{1,2,\ldots,n\}$ 使得

$$\sum_{i \in S^*} v_i = \max\left\{ \sum_{i \in S} v_i \mid \sum_{i \in S} w_i \le B, S \subseteq \{1,2.\cdots,n\} \right\}.$$

8.4.1 一个简单的贪心算法

贪心算法G-KK

1. 按单位重量的价值从大到小排列物品. 设

$$v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n.$$

2. 顺序检查每一件物品, 只要能装得下就将它装入背包, 设装入背包的总价值为 V.

3. 求 $v_k = \max\{ v_i \mid i = 1, 2, \ldots, n \}$.

若$v_k > V$, 则将背包内的物品换成物品 k.

实例 (w_i,v_i): (3,7), (4,9), (5,9), (2,2); B=6.

G-KK给出的解是装入(3,7)和(2,2), 总价值为9. 若把第3件物品改为(5,10), 则装入第3件, 总价值为10.

这两个实例的最优解都是装入(4,9)和(2,2), 总价值为11.

北京大学

G-KK的性能

定理8.7 对0-1背包问题的任何实例 I, 有

$$\mathrm{OPT}(I) < 2\,\mathrm{G\text{-}KK}(I).$$

证 设物品 l 是第一件未装入背包的物品, 由于物品按单位重量的价值从大到小排列, 故有

$$\begin{aligned}\mathrm{OPT}(I) &< \mathrm{G\text{-}KK}(I) + v_l \\ &\le \mathrm{G\text{-}KK}(I) + v_{\max} \\ &\le 2\,\mathrm{G\text{-}KK}(I).\end{aligned}$$

G-KK是2-近似算法.

8.4.2 多项式时间近似方案

算法 PTAS 输入$\varepsilon>0$和实例 I.

1. 令 $m=\lceil 1/\varepsilon \rceil$.
2. 按单位重量的价值从大到小排列物品. 设

$$v_1/w_1 \geq v_2/w_2 \geq \ldots \geq v_n/w_n.$$

3. 对每一个 $t=1,2,\ldots,m$ 和 t 件物品, 检查这 t 件物品的重量之和. 若它们的重量之和不超过B, 则接着用G-KK把剩余的物品装入背包.
4. 比较得到的所有装法, 取其中价值最大的作为近似解.

PTAS是一簇算法. 对每一个固定的 $\varepsilon > 0$, PTAS是一个算法, 记作 PTAS_ε .

PTAS的性能

定理8.8 对每一个$\varepsilon>0$和0-1背包问题的实例I,

$$\mathrm{OPT}(I) < (1+\varepsilon)\,\mathrm{PTAS}_{\varepsilon}(I),$$

且 $\mathrm{PTAS}_{\varepsilon}$的时间复杂度为$O(n^{1/\varepsilon+2})$.

证 设最优解为S^*. 若 $|S^*| \le m$, 则算法必得到 S^*. 设$|S^*| > m$. 考虑计算中以 S^*中 m 件价值最大的物品为基础, 用G-KK得到的结果 S. 设物品 l 是 S^* 中第一件不在 S 中的物品, 在此之前 G-KK装入的不属于 S^* 的物品 (肯定有这样的物品, 否则应该装入物品 l) 的单位重量的价值都不小于 v_l/w_l, 当然也不小于 S^* 中所有没有装入的物品的单位重量的价值, 故有 $\mathrm{OPT}(I) < \sum_{i\in S} v_i + v_l$. 又, S 包括 S^* 中 m 件价值最大的物品, 它们的价值都不小于 v_l, 故又有 $v_l \le \sum_{i\in S} v_i/m$.

北京大学

多项式时间近似方案

$$\mathrm{OPT}(I) < \sum_{i\in S} v_i + v_l \le \sum_{i\in S} v_i + \sum_{i\in S} v_i/m$$
$$\le (1+1/m)\mathrm{PTAS}_\varepsilon(I) \le (1+\varepsilon)\mathrm{PTAS}_\varepsilon(I)$$

时间复杂度. 从 n 件物品中取 t 件(t =1,2,…,m), 所有可能取法的个数为

$$c_n^1 + c_n^2 + \cdots + c_n^m \le m \cdot \frac{n^m}{m!} \le n^m .$$

对每一种取法, G-KK的运行时间 为$O(n)$, 故算法的时间复杂度为 $O(n^{m+1}) = O(n^{1/\varepsilon+2})$.

多项式时间近似方案: 以ε>0和问题的实例作为输入 I, 对每一个固定的 $\varepsilon > 0$, 算法是 1+ε - 近似的.