A Brief Introduction to the Approximation Algorithm

FPTAS for solving the 0-1 Knapsack Problem

Emil Huang
huangh@suda.edu.cn

近似算法和近似比

近似算法

- 当NP类问题的输入极大时，要求该问题的最优解，算法的执行时间将会是指数级别的。这时，我们很难找到问题的最优解(即使能够找到，花费的代价也是极大的)。因此我们通常会选择在多项式时间内再到问题的一个近似解。

近似比(Approximation Ratio)

设一个最优化问题的最优值为C^*，该问题的近似算法求得了近似最优值C，有近似比

$$\eta = \max\left\{\frac{C}{C^*}, \frac{C^*}{C}\right\}$$ （最大化、最小化问题均适用）

一个能求得最优解的近似算法得到的近似比为1

最大化问题中，如果$C \le C^*$, 则最优解C^*比近似解C大$\left(\frac{C^*}{C} - 1\right)$倍

0-1背包问题的定义

0-1背包问题

- 给定n种物品和一背包。物品i的重量是w_i，其价值为v_i，背包的容量为C。应如何选择装入背包的物品，使得装入背包中物品的总价值最大?

$$\max \sum_{i=0}^{n} v_i x_i$$

$$\begin{cases} \sum_{i=0}^{n} w_i x_i \le C \\ x_i \in \{0,1\}, 1 \le i \le n \end{cases}$$

0-1背包问题是一个NP完全问题

0-1背包问题的DP求解

- 递归式

$$m(i,j) = \begin{cases} \max\{m(i+1,j), m(i+1,j-w_i)+v_i\} & j \ge w_i \\ m(i+1,j) & 0 \le j \le w_i \end{cases}$$

- 算法复杂度分析：

  从$m(i,j)$的递归式容易看出，算法需要$O(n \cdot C)$计算时间，C表示背包容量。当$C > 2^n$时，算法需要$\Omega(n \cdot 2^n)$计算时间。

对于0-1背包问题，当问题的输入变得无穷大时，我们往往需要付出很大的代价才能得到问题的最优解。对此，我们希望设计一个近似算法能够保证该问题在多项式时间内被解决(即一个求解时间短很多的近似算法)。

Definition of the approximation scheme

近似方案 (Approximation Scheme)

- 设π是目标函数为f_π的$NP\text{-}hard$最优化问题，如果关于输入(I,ε)，I为π的实例，ε为误差参数，算法$\mathcal{A}$输出解S使得

  1) 若π是最小化问题，$f_\pi(I,S) \le (1+\varepsilon)\cdot OPT$

  目标函数在输入为I的实例下输出解S小于$(1+\varepsilon)\cdot$OPT

  2) 若π是最大化问题，$f_\pi(I,S) \ge (1-\varepsilon)\cdot OPT$

我们称算法$\mathcal{A}$是π的近似方案。

PTAS和FPTAS的定义

PTAS (Polynomial-Time Approximation Scheme)

- 若对于每一个固定的$\varepsilon > 0$，算法$\mathcal{A}$的运行时间以实例I的规模的多项式为上界，则称$\mathcal{A}$是一个多项式时间近似方案（简称PTAS）。

FPTAS (Fully Polynomial-Time Approximation Scheme)

- 在PTAS的基础上，我们进一步要求算法$\mathcal{A}$，即算法$\mathcal{A}$的运行时间以实例I的规模和$1/\varepsilon$的多项式为上界，则称$\mathcal{A}$是一个完全多项式时间近似方案（简称FPTAS）。

FPTAS被认为是最值得研究的近似算法，

仅有极少的$NP\text{-}hard$问题存在FPTAS。

伪多项式时间算法

- 从给出近似方案的定义到确定FPTAS的过程中，我们的定义始终基于了假设，即出现在实例 I 中的数都以二进制的形式存放的，同时我们用$|I|$表示在该假定下实例I的规模。我们用I_u表示把出现的所有数以一进制形式存放的实例I，定义实例I的一进制规模为记录I_u所需的一进制位数，记作$|I_u|$。
- 如果π的算法在实例I上的运行时间以$|I|$的多项式为上界，则称算法是有效的。如果π的算法在实例I上的运行时间以$|I_u|$的多项式为上界，则称算法是伪多项式时间算法。

7

一个0-1背包的伪多项式时间算法

- 问题

给定物体集合$S=\{a_1,\ldots,a_n\}$，每个物体有指定的大小$size(a_i)\in\mathbb{Z}^+$和指定的收益$profit(a_i)\in\mathbb{Z}^+$，以及背包容量$B\in\mathbb{Z}^+$，找一个子集，该子集的总大小以$B$为上界并且它的总收益最大。

- 算法思路

定义P是收益最大的物体的收益，$P=\max\limits_{a_i\in S} profit(a_i)$。

则nP是任何解能够获得的收益的一个平凡上界。

对于每个$i\in\{1,\ldots,n\}$和$p\in\{1,\ldots,nP\}$，设$S_{i,p}$表示$\{a_1,\ldots,a_i\}$的一个子集，该子集的总收益正好是p，并且它的总大小Size达到最小。设$A(i,p)$表示集合$S_{i,p}$的大小(如果这样的集合不存在，则$A(i,p)=\infty$)。显然，对于每一个$p\in\{1,\ldots,nP\}$，$A(1,p)$是已知的(递归的终止条件)。

8

一个0-1背包的伪多项式时间算法

我们可给出如下的递归式

$$A(i+1,p)=\begin{cases}\min\{A(i,p),size(a_{i+1})+A(i,p-profit(a_{i+1}))\} & if\ \ profit(a_{i+1})\le p\\ A(i,p) & otherwize\end{cases}$$

总大小以B为上界的各物体能够达到的最大收益为$\max\{p|A(n,p)\le B\}$。（p可能是指数级的）

上述递归可在$\underline{O(n^2P)}$时间内计算处结果

↓

n个物品，每个物品对应1～nP，取值不同的子问题

9

0-1背包问题的FPTAS

■设计思路

若各物品收益均是比较小的数，即它们以n的多项式为上界，那么基于上述递推式所设计的DP算法是一个常规的多项式时间算法，因为它的运算时间以$|I|$的多项式为上界，但是，当p是指数级，就需要对物品价值进行放缩，即

忽略各物品价值的最后若干有效位(依赖于误差参数ε)

以便将修改后的收益看作是以n和$^1/_\varepsilon$的多项式为上界的数。

所以，我们可以在以n和$^1/_\varepsilon$的多项式为上界的时间内找到收益至少为$(1-\varepsilon)\cdot OPT$的解。

精确度换取时间

10

算法 0-1背包问题的FPTAS

(实现方法 -- 缩放和取整)

STEP 1：给定$\varepsilon>0$，设K $=\frac{\varepsilon P}{n}$.

STEP 2：对每个物品a_i，定义$profit'(a_i)=\left\lfloor\frac{profit(a_i)}{K}\right\rfloor$.

STEP 3：用这些值作为物品新的收益(价值)，利用前述动态规划算法找到收益最大的集合，记为S'.

STEP 4：输出S'.

11

定理1：算法的时间复杂度为$O\left(n^2\cdot\left\lfloor\frac{n}{\varepsilon}\right\rfloor\right)$

（ 如$\varepsilon=0.1$，能够保证DP算法的结果$\ge 0.9\cdot OPT$，而算法的复杂度为$O(10n^3)$ ）

证明：代入新的物品价值后，DP所需要的时间

$$O(n^2\cdot P)=O\left(n^2\cdot\left\lfloor\frac{P}{K}\right\rfloor\right)=O\left(n^2\cdot\left\lfloor\frac{P}{\frac{\varepsilon P}{n}}\right\rfloor\right)=O\left(n^2\left\lfloor\frac{n}{\varepsilon}\right\rfloor\right)$$

12

定理2：算法输出的集合S'满足

$$profit(S') \geq (1-\varepsilon)\cdot OPT$$

证明：用O来表示未缩放之前的最优物品集合(对应最优解OPT)。对于任意物品a，我们在STEP 2中作了向下舍入操作。因此

$$\begin{cases} K\cdot profit'(a) \leq profit(a) \\ K\cdot profit'(a) > profit(a) - K \\ \qquad \left(\Leftarrow \quad profit'(a) > \dfrac{profit(a)}{K} - 1\right) \end{cases}$$

有

$$profit(a) - K < K\cdot profit'(a) \leq profit(a)$$

13

由 $profit(a) - K < K\cdot profit'(a)$

得 $\forall i$，$profit(a) - K\cdot profit'(a) < K$

因此 $profit(O) - K\cdot profit'(O) < nK$

又因为DP算法所得解为最优解(S'是在新价值体系下至少与O一样好的集合)，因此

$$profit(S') \geq K\cdot profit'(O) \geq profit(O) - nK$$
$$= OPT - \varepsilon p$$
$$\because \quad OPT \geqq P$$
$$\therefore \quad profit(S') \geqq (1-\varepsilon)\cdot OPT$$

14

求解0-1背包问题的贪心算法

输入：正整数$W, w_1, v_1, w_2, v_2, \ldots, w_n, v_n$。

1) 依据$^{v_i}/_{w_i}$的大小，依单调递减序排列所有物品。

不妨设$^{v_1}/_{w_1} \geq {}^{v_2}/_{w_2} \geq \cdots \geq {}^{v_n}/_{w_n}$

2) 若$\sum_{i=0}^{n} w_i \leq W$，则输出：$C_G \leftarrow \sum_{i=0}^{n} v_i$

否则求出最大的k，使其满足：

$$\sum_{i=1}^{k} w_i \leq W < \sum_{i=1}^{k+1} w_i$$

输出：$C_G \leftarrow max\{v_{k+1}, \sum_{i=1}^{k} v_i\}$

15

定理：对于背包问题的一个实例，设opt表示最优解目标函数值，C_G是由上述算法生成的近似解目标函数值，则$opt \leq 2C_G$(近似比为2)

证明：

1) 若$\sum_{i=0}^{n} w_i \leq W$，则$C_G = opt$

2) 若$\sum_{i=0}^{n} w_i > W$，设k是算法得到的整数

可证$\sum_{i=1}^{k} v_i \leq opt < \sum_{i=1}^{k+1} v_i$

16

最优值opt的上界怎么算？

先放前k个物品，假定第$(k+1)$个物品可分割，将背包装满，用其他物品代替只会降低背包物品的价值/重量比值

OPT少于$\sum_{i=1}^{k+1} v_i$，最优值opt上界求解：

$$opt \leq \hat{c} = \sum_{i=1}^{k} v_i + \frac{v_{k+1}}{w_{k+1}}\left(W - \sum_{i=1}^{k} w_i\right)$$
$$< \sum_{i=1}^{k} v_i + \frac{v_{k+1}}{w_{k+1}}\cdot w_{k+1} = \sum_{i=1}^{k+1} v_i$$

$$C_G = \max\left\{v_{k+1}, \sum_{i=1}^{k} v_i\right\} \geq \frac{1}{2}\sum_{i=1}^{k+1} v_i > \frac{opt}{2}$$
$$\because \max\{a,b\} \geq \frac{1}{2}(a+b)$$

17