1 ナップサック問題 (Knapsack Problem)

ナップサック問題は以下のように定義できる。

入力 サイズ $a_1, a_2, \ldots, a_n$, 価値 $c_1, c_2, \ldots, c_n$ の n 個の品物集合 X とナップサックサイズ b（a_i, c_i, b はすべて正の整数で、$a_i \leq b$ を仮定）.

問題 サイズが b 以下の X の品物の部分集合 S で価値が最大となるものを求める（S のサイズは $a(S) = \sum_{i \in S} a_i$ で価値は $c(S) = \sum_{i \in S} c_i$）。

以下の整数計画問題として捉えることもできる。

$$\begin{aligned} &\max\ \boldsymbol{c}^T \boldsymbol{x} \\ &\text{s.t. } \boldsymbol{a}^T \boldsymbol{x} \leq b\ ,\ x_i \in \{0, 1\} \end{aligned}$$

2 Greedy アルゴリズム (Greedy Algorithm)

1. 単位価値の高い順番に並べる（$c_1/a_1 \geq c_2/a_2 \geq \ldots \geq c_n/a_n$ とする）。
2. $i := 1$ から n まで順番に $a_i < b - \sum_{j=1}^{i-1} a_j x_j$ ならば $x := 1$ とし、そうでないならば $x_i := 0$ として、$x_i = 0$ を定める。

解 $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ の価値 $\sum_{i=1}^{n} c_i x_i$ を $C(\boldsymbol{x})$ と書くことにする。Greedy アルゴリズムで得られる解を $\boldsymbol{x}^g = (x_1^g, x_2^g, \ldots, x_n^g)$,$\boldsymbol{x}^{opt}$ を最適解, $C := \max_{1 \leq i \leq n} c_i$ とする。すると、以下が成立する。

$$C(\boldsymbol{x}^g) = \sum_{i=1}^{n} c_i x_i^g \geq C(\boldsymbol{x}^{opt}) - C$$

実際、$\boldsymbol{x}^g$ が最適解ならば $C \geq 0$ であるから、上式が成立することは明らかである。

また、$\boldsymbol{x}^g$ が最適解でないときは、ある k が存在して、次が成立する。

$$1 \geq x_k^{opt} > x_k^g = 0$$

ここで、上を満たす k で最小のものを j と定める。すると任意の $i \leq j - 1$ に対して、次が成立する

$$x_i^g \geq x_i^{opt}$$

よって

$$\begin{aligned} C(\boldsymbol{x}^g) &= \sum_{i=1}^{n} c_i x_i^g \geq \sum_{i=1}^{j} c_i x_i^g = \sum_{i=1}^{j} c_i \left(\left(x_i^{opt} \right) + \left(x_i - x_i^{opt} \right) \right) \\ &\geq \sum_{i=1}^{j} c_i x_i^{opt} + \sum_{i=1}^{j} \frac{c_j}{a_j} a_i \left(x_i - x_i^{opt} \right) \left(\because \frac{c_i}{a_i} \geq \frac{c_j}{a_j} \geq 0 \right) \\ &= \sum_{i=1}^{j} c_i x_i^{opt} + \frac{c_j}{a_j} \left(\sum_{i=1}^{j} a_i x_i^g - \sum_{i=1}^{j} a_i x_i^{opt} \right) \end{aligned}$$

また、j は Greedy アルゴリズムで選ばれなかったので

$$\sum_{i=1}^{j} a_i x_i^g = \sum_{i=1}^{j-1} a_i x_i^g > b - a_j$$

であり、$\boldsymbol{x}^{opt}$ は解なので $\sum_{i=1}^{n} a_i x_i^{opt} \leq b$ より

$$\sum_{i=1}^{j} a_i x_i^{opt} \leq b - \sum_{i=j+1}^{n} a_i x_i^{opt}$$

さらに、任意の $i \geq j+1$ に対して、$0 < c_i/a_i \leq c_j/a_j$ であるから

$$\begin{aligned}
\frac{c_j}{a_j}\left(\sum_{i=1}^{j} a_i x_i^g - \sum_{i=1}^{j} a_i x_i^{opt}\right) &\geq \frac{c_j}{a_j}\left((b-a_j) - \sum_{i=1}^{j} a_i x_i^{opt}\right) \\
&\geq \frac{c_j}{a_j}\left((b-a_j) - \left(b - \sum_{i=j+1}^{n} a_i x_i^{opt}\right)\right) \\
&= \frac{c_j}{a_j}\left(\sum_{i=j+1}^{n} a_i x_i^{opt} - a_j\right) \\
&\geq \sum_{i=j+1}^{n} c_i x_i^{opt} - c_j
\end{aligned}$$

故に

$$\begin{aligned}
C(\boldsymbol{x}^g) &\geq \sum_{i=1}^{j} c_i x_i^{opt} + \frac{c_j}{a_j}\left(\sum_{i=1}^{j} a_i x_i^g - \sum_{i=1}^{j} a_i x_i^{opt}\right) \\
&\geq \sum_{i=1}^{j} c_i x_i^{opt} + \sum_{i=j+1}^{n} c_i x_i^{opt} - c_j \\
&= \sum_{i=1}^{n} c_i x_i^{opt} - c_j \\
&= C(\boldsymbol{x}^{opt}) - c_j \\
&\geq C(\boldsymbol{x}^{opt}) - C
\end{aligned}$$

以上より

$$C(\boldsymbol{x}^g) \geq C(\boldsymbol{x}^{opt}) - C$$

であることが分かった。

ここで、解として Greedy アルゴリズムの解と最大価値のもの C を 1 つ選ぶアルゴリズムで得られた解のうち価値の高いほうを採用するアルゴリズムの近似値比を考える。この近似解を $\boldsymbol{x}^a = (x_1^a, \ldots, x_n^a)$ とする。その価値は

$$C(\boldsymbol{x}^a) = \sum_{i=1}^{n} c_i x_i^a \geq \frac{C(\boldsymbol{x}^g) + C}{2} \geq \frac{1}{2} C(\boldsymbol{x}^{opt})$$

を満たす。すなわち、このアルゴリズムは近似値比 0.5 の近似アルゴリズムである。

ここで、Greedy アルゴリズムの計算時間を考えると単位価値の高い順番に並べることに一番時間がかかり $O(n \log n)$ かかる.

次に、この Greedy アルゴリズムを利用した PTAS を見てみよう。

3 PTAS

3.1 PTAS とは

Polynomial Time Approximation Scheme

略。

3.2 ナップサック問題に対する PTAS

集合で考えると楽なので以下集合で議論する。$C(S) := \sum_{i \in S} c_i$ とする。

1. X の空でない部分集合で要素数 k 以下の S を列挙する。$\left(S \in 2^X, |S| \leq k\right)$
2. 各部分集合 S について、残りの集合 $X \backslash S$ に対して、ナップサックサイズ $\tilde{b}_S := b - \sum_{i \in S} a_i$ として Greedy アルゴリズムを適用（$\tilde{b}_S < 0$ の場合その S は以降考えない）。
   その解を G_S とする。
3. $C(G_S) + C(S)$ が最大となる S を探し、解を $A := S \cup G_S$ とする。

このアルゴリズムの計算時間について考える。
X の空でない部分集合で要素数 k 以下のものの数は

$$\sum_{m=1}^{k} {}_nC_m \leq \sum_{m=1}^{k} n^k = O(kn^{k+1})$$

である。この部分集合それぞれにに対して Greedy アルゴリズムを適用する。前処理としてあらかじめ単位価値によって品物をソートしておけば、Greedy アルゴリズムは $O(n)$ で実行可能であるから、全体の計算時間は $O(kn^k)$ である。

つぎにこのアルゴリズムの解 A の最適解 O に対する近似率について考える。
次が成立する。

$$C(O) \leq \left(1 + \frac{1}{k}\right) C(A)$$

まず、$|O| \leq k$ の場合は、部分集合 S のいずれかが最適解であるから O と A は一致して $C(O) = C(A)$ が成立する。そこで、$|O| > k$ の場合について考える。

$H = \{i_1, \ldots, i_k\}$ を最適解に含まれる品物を大きい順に k 取り出してきた集合とする。つまり、

$$c_{i_1} \geq c_{i_2} \geq \ldots \geq c_{i_k} \geq \ldots \geq c_{i_{|O|}} \ , \ i_l \in O$$

ここで、$|H| = k$ であるから、部分集合 S のいずれかは H と一致する。そこで、$X \backslash H$ に対する Greedy アルゴリズムについて考える。（これより得られる解が上の近似率を与える。）

$R = O \backslash H = \{j_1, j_2, \ldots, j_{|O|-k}\}$ を $O \backslash H$ のなかを単位価値の大きい順に並べたものとする。つまり、

$$\frac{c_{j_1}}{a_{j_1}} \geq \frac{c_{j_2}}{a_{j_2}} \geq \ldots \geq \frac{c_{j_{|R|}}}{a_{j_{|R|}}}$$

さて、$X \backslash H$ に対する Greedy アルゴリズムを実行中に R に含まれる要素が Greedy アルゴリズムの解に最初に含まれなくなったときを考える。これは含まれなかった要素を j_m, ナップサックの残りのサイズを b_e とすると $a_{j_m} > b_e$ が成立するからである。

また、この時点で Greedy アルゴリズムの解に含まれ、最適解には含まれない要素のサイズの合計 Δ について考えると、先の b_e を用いて

$$\sum_{i \in H} a_i + \sum_{\ell=1}^{m-1} a_{j_\ell} + \Delta + b_e = b$$

が成立する。さらに、Greedy アルゴリズムの性質から、この時点で Greedy アルゴリズムの解に含まれる要素の単位あたりの価値は少なくとも、c_{j_m}/a_{j_m} である。

最終的に得られる Greedy アルゴリズムの解 G は、この時点の解よりも要素が減っていることはないので、以下の不等式が成立する。

$$C(G) \geq \sum_{\ell=1}^{m-1} c_{j_\ell} + \Delta \frac{c_{j_m}}{a_{j_m}}$$

ここで、最適解の価値 $C(O)$ を考えると

$$\begin{aligned}
C(O) &= \sum_{i \in H} c_i + \sum_{j \in R} c_j \\
&= C(H) + \sum_{\ell=1}^{m-1} c_{j_\ell} + \sum_{\ell=m}^{|R|} c_{j_\ell} \\
&\leq C(H) + \left(C(G) - \Delta \frac{c_{j_m}}{a_{j_m}} \right) + \sum_{\ell=m}^{|R|} c_\ell \\
&\leq C(H) + \left(C(G) - \Delta \frac{c_{j_m}}{a_{j_m}} \right) + \left(b - \sum_{i \in H} a_i - \sum_{\ell=1}^{m-1} a_{j_\ell} \right) \frac{c_{j_m}}{a_{j_m}} \\
&= C(H) + C(G) + b_e \frac{c_{j_m}}{a_{j_m}} \\
&\leq C(H) + C(G) + c_{j_m} \\
&= C(H \cup G) + c_{j_m}
\end{aligned}$$

H の選び方から

$$(k+1) c_{j_m} \leq C(O)$$

よって

$$C(O) \leq C(H \cup G) + \frac{1}{k+1} C(O) \leq C(A) + \frac{1}{k+1} C(O)$$

したがって

$$C(O) \leq \left(1 + \frac{1}{k} \right) C(A)$$

これより

$$C(A) \geq \frac{1}{1 + 1/k} C(O) > \left(1 - \frac{1}{k} \right) C(O)$$

PTAS を得るためには $\varepsilon = 1/k$ とすればよく、このとき計算時間は $O((1/\varepsilon) n^{1/\varepsilon})$ である。

4 動的計画法 (Dynamic Programming)

$d(p, q)$ を品物集合 $\{1, 2, \ldots, p\}$ の部分集合でちょうど価値が q となるもののサイズの最小値とする。ただし、価値 q を達成する部分集合が存在しない場合 $d(p, q) = \infty$ とする。

次の漸化式が成り立つ。

$$d(p+1, q) = \begin{cases} \min\{d(p, q), d(p, q - c_{p+1}) + a_{p+1}\} & q \geq c_{p+1} \\ d(p, q) & q < c_{p+1} \end{cases}$$

ここで、$d(1, q)$ は容易に計算できる。また、ナップサック問題の最適値は nC で抑えられる。ゆえに、$d(p, q)\,(p = 1, 2, \ldots, n, q = 0, 1, \ldots, nC)$ は動的計画法に基づいて計算でき、その計算時間は $O(n^2 C)$ である。また、出来上がった $d(p, q)$ を順に見ていくことでナップサック問題に対する最適解が得られる。

これは入力長 $\log C$ に対する擬多項式時間アルゴリズムである。もし、C を n の多項式程度まで小さくできれば、それは多項式時間アルゴリズムである。この点に着目してスケーリング (scaling) を用いて、ナップサックに対する FPTAS を構成することが可能である。

5 FPTAS

5.1 FPTAS とは

Fully Polynomial Time Approximation Scheme
略。

5.2 ナップサック問題に対する FPTAS

1. 正の数 ε に対して、$K := \varepsilon C/n$ とする。
2. 各品物に対して、その価値を $c'_i := \lfloor c_i/K \rfloor$ と小さくする
3. こうして、価値を小さくした状態でのナップサック問題の最適解 D を動的計画法により求める。
4. 上の解 D と最初の価値で最大価値の品物を 1 つ選ぶ解のうち価値の高いほうを近似解 A とする。

上のアルゴリズムが FPATS であることを示そう。計算時間に関しては動的計画法の部分が最も時間がかかり $O(n^2\lfloor C/K \rfloor) = O(n^2\lfloor n/\varepsilon \rfloor) = O(n^3/\varepsilon)$ で、多項式時間アルゴリズムである。

次に近似率を考える。$c'_i := \lfloor c_i/K \rfloor$ より、

$$0 \leq c_i - Kc'_i < K$$

が成立する。ここで、$C'(S) = \sum_{i \in S} c'_i$ として、小さくした価値のもとでの S の価値を表すことにすれば、任意の X の部分集合 S について

$$0 \leq C(S) - C'(S) < nK$$

これは、当然最適解 O、小さくした価値のもとでの動的計画法の解 D についても成立する。また、D は小さくした価値のもとでは最適解であるから、

$$C'(O) \geq C'(D)$$

さらに、アルゴリズムより $C(A) \geq C$ であるから、

$$\begin{aligned} C(A) &\geq C(D) \geq KC'(D) \\ &\geq KC'(O) \\ &> C(O) - nK \\ &= C(O) - n\frac{\varepsilon C}{n} \\ &= C(O) - \varepsilon C \\ &\geq C(O) - \varepsilon C(A) \end{aligned}$$

以上より

$$C(A) \geq \frac{1}{1+\varepsilon} C(O) > (1-\varepsilon)C(O)$$