第１１回：１２月１６日（水）

11　第１１回

11.1　ナップサック問題に対する動的計画法（利得版）

0,1-ナップサック問題の問題例をアイテムのサイズ $a_i,\ i = 1, 2, \ldots, n$，アイテムの利得 c_i, $i = 1, 2, \ldots, n$，ナップサックサイズ b とする．今度は 0,1-ナップサック問題でアイテムの利得 c_i が整数値である場合を考える．

部分問題の定義

$c_{\mathrm{SUM}} \triangleq \sum_{i=1,2,\ldots,n} c_i$ と定義する．整数 $j = 1, 2, \ldots, n$, $p = 0, 1, \ldots, c_{\mathrm{SUM}}$ に対し，利得和が p 以上となるように $\{1, 2, \ldots, j\}$ からアイテム集合 S を選ぶときのサイズ和の最小値を $g^*(j, p)$ と定義する．すなわち，

$$g^*(j,p) \triangleq \min\Big\{\sum_{i\in S} a_i \mid S \subseteq \{1,2,\ldots,j\}, \sum_{i\in S} c_i \geq p\Big\}.$$

（利得和を p 以上にする選択がないときは $g^*(j,p) \triangleq \infty$ とする.）

元の問題の最適値は，$g^*(n,p), p = 0, 1, \ldots, c_{\mathrm{SUM}}$ の値から次のように算出する．ナップサックサイズ b に対して，$g^*(n,p) \leq b$ である最大の p^* を見つける．部分問題の定義に照らし合わせると，利得和が p^* 以上となるように必要な最小のサイズ和は b 以下であり，$g^*(n, p^*+1) > b$ であることからナップサックサイズ b では利得和が p^*+1 以上となるような選択はできないことを言っている．よって p^* が元の問題の最適値となる．

実は，上記の部分問題の定義にはナップサックサイズ b の情報は直接，使われていないので，後からナップサックサイズ b を選び直しても最適値が算出できる．

再帰式の導出

境界条件：$p = 0, 1, \ldots, c_{\mathrm{SUM}}$,

$$g^*(1,p) = \begin{cases} 0, & p = 0, \\ a_1, & p \leq c_1, \\ \infty, & p > c_1. \end{cases}$$

$g^*(j,p)$ の最適解 S はアイテム $1, 2, \ldots, j$ の中から選択するが，その最適解 S にアイテム j が使われていない場合と使われている場合に分けて考える．

- **アイテム j が使われていない場合:** 最適解 S はそのままアイテム $1,2,\ldots,j-1$ の中から利得和が p 以上の制約下で選ぶ最適解となっている（最適性の原理）．よって，$g^*(j,p)=g^*(j-1,p)$ が成り立つ．

- **アイテム j が使われている場合:** $S'=S\setminus\{j\}$ もやはりアイテム $1,2,\ldots,j-1$ の中から利得和が $p-c_j$ 以上の制約下で選ぶ最適解となっている．この場合は，a_j のサイズを忘れずに足して，$g^*(j,p)=g^*(j-1,p-c_j)+a_j$ が成り立つ（最適性の原理）．

  ただし $p<c_j$ の場合は $g^*(j,p)=a_j$ ($S=\{j\}$, アイテム j のみを使う) となる．

以上から以下のように再帰式を得ることができる．
漸化式：$j=2,3,\ldots,n$,

$$g^*(j,p)=\begin{cases}\min\{g^*(j-1,p),\ g^*(j-1,p-c_j)+a_j\}, & p\geq c_j\\ \min\{g^*(j-1,p),\ a_j\}, & p<c_j.\end{cases}$$

アルゴリズムの設計

サイズ版の動的計画法アルゴリズムと同様である．境界条件の問題から解き始め，得られた最適値を保存し，順に大きな部分問題の最適値を漸化式に従って決定していく．元の問題の最適値が得られたら，これを与えている部分問題をバックトラックして探索し，最適解を決定する．

次の例題を用いて説明する．

$$\begin{aligned}c_1&=2,\quad c_2=3,\\ a_1&=4,\quad a_2=7,\\ b&=8.\end{aligned}$$

サイズ版の動的計画法アルゴリズムと同様に，部分問題 $g^*(j,p)$ は 2 個のパラメータを持つので for ループを二重に持つ反復アルゴリズムの形で書くことができるが，計算の進行は，j,p を成分とする行列（表）のセルを埋めていく様子を見ると理解しやすい．スライドを使って説明する．

♠「講義資料と課題」2〜7 ページ

2 ページ: $c_{\mathrm{SUM}}=\sum_{i=1,2,\ldots,n}c_i=c_1+c_2=5$ であるので，$p=0,1,\ldots,5$ の範囲で部分問題が定義される．スライド 2 ページは境界条件に従って表の一段目を埋めた様子である．

3 ページ: スライド 3 ページで示すように，$p\geq c_2=3$ の場合も $p<c_2=3$ の場合も，再帰式の第一項は，上の段の値のコピーを取ることに相当する．

4 ページ: 一方再帰式の第二項に関して，

- $p\geq c_2=3$ の場合は上の段の値を $c_2=3$ だけ右へシフトさせたのち，$a_2=7$ を上乗せすることに相当する．

- $p < c_2 = 3$ の場合は $a_2 = 7$ を取る．

5 ページ: そして，第一項と第二項のうち大きな利得値を選ぶ．これで，$g^*(n,p), p = 0, 1, \ldots, c_{\mathrm{SUM}}$ が得られる．

6 ページ: ここでナップサックサイズ b がどのように与えられていても対応する最適値は 6 ページのように決定できる．いま，$b = 8$ としているので，最適値は 3 である．

7 ページ: 次に，最適解を求める．これには表を埋めていく際に，再帰式において第一項と第二項のうちどちらが最大値を決めていたかを記録しておく．すると，最適値に対応する項 $g^*(2,3) = 7$ から順に第一項と第二項の二択のうち最大のほうへさかのぼっていくことで，サイズ和が $g^*(2,3) = 7$ を達成しているアイテムの集合が得られる．この問題例の場合は，２番目のアイテムを使うことが分かり，次に 1 番目のアイテムは使わないことが判明する．最適解は $z = (z_1 = 0, z_2 = 1)$ となる．

計算量の上界の解析

部分問題の個数は nc_{SUM} であり，これに比例する領域があれば計算が実行できるので，領域計算量は $O(nc_{\mathrm{SUM}})$ である．部分問題をひとつ解くためには，$g^*(j,p) = \min\{g^*(j-1,p),\ g^*(j-1, p-c_j) + a_j\}$ を計算する必要があるが，いずれも 2 次元配列にデータを格納しておけば $O(1)$ 時間で計算できる．部分問題をひとつ $O(1)$ 時間で解けるので，部分問題の個数をかけて，$O(nc_{\mathrm{SUM}})$ がアルゴリズム全体の時間計算量の上界となる．この上界は正確である．これも疑多項式時間である．

【課題１１－１】スライドで実演した計算例を参考にして，次の 0,1-ナップサック問題の例題に対して動的計画法アルゴリズム（利得版）を適用したときの計算の過程を表にせよ．最適値だけでなくバックトラックにより最適解を求める過程も説明をつけること．

$$
\begin{aligned}
&c_1 = 2, \quad c_2 = 3, \quad c_3 = 5,\\
&a_1 = 4, \quad a_2 = 7, \quad a_3 = 12,\\
&b = 20.
\end{aligned}
$$

11.2 ナップサック問題に対する近似スキーム

NP-困難な問題であるナップサック問題に対して分枝限定法，整数値サイズ（および整数値利得）の場合にはサイズ版（および利得版）の動的計画法アルゴリズムを設計してきた．アイテム数 n やサイズ，利得の大きさが大きいときには多大な計算時間がかかる．そこで最適解ではなく，目的関数値が最適値に近い実行可能解を求めることも重要となる．

最適値の k 倍以内の目的関数値をもつ実行可能解を **k-近似解** (k-approximation solution) という．近似率 k については，最適値と近似値のどちらを分母にするかは定義の仕方による．

与えらえた実数 $0 < \varepsilon < 1$ に対して，$(1-\varepsilon)$-近似解（最大利得の $(1-\varepsilon)$ 倍以内の利得を持つ実行可能解）を $n, 1/\varepsilon$ の多項式で求めるアルゴリズムを設計する．ε を近似精度として近似率をいくらでも 1 に近づけられるアルゴリズムを**近似スキーム** (approximation scheme) という．

0,1-ナップサック問題の例題をアイテムのサイズの集合，利得の集合，ナップサックサイズの組として $I = (\{a_1, a_2, \ldots, a_n\}, \{c_1, c_2, \ldots, c_n\}, b)$ と記し，この問題例の最適値（最大利得）を $\mathrm{opt}(I)$ と記す．ここで，サイズや利得は概念上は整数値である必要はない（概念上とは，通常の計算機でデータを有限桁の数字で表現する場合には有理数であり，これは整数値として扱うことができる）．

問題例 I の $(1-\varepsilon)$-近似解を求めるために，勉強した動的計画法アルゴリズムを使うことを考える．分枝限定法はサイズや利得が整数値でなくても作動するが，最適値からの近似率を評価することが難しい．問題例 I にサイズ版あるいは利得版の動的計画法アルゴリズムを適用し，計算量を減らし，$(1-\varepsilon)$-近似解を求めるために，問題例 I をまず加工することを考える．もし，サイズ a_i のデータを加工してしまうと，解の実行可能性が保持できなくなる恐れがあるので，利得 c_i のデータを加工する．簡単に言えば，各利得 c_i をそれに比例する整数値 $\widetilde{c}_i$ に置き換えれば，利得版の動的計画法アルゴリズムで加工した問題例 $\widetilde{I}$ を解くことができる．加工後の整数値 $\widetilde{c}_i$ が小さいほどアルゴリズムの計算量 $O(n\sum_{i=1}^{n}\widetilde{c}_i)$ は小さくなる．もちろん，利得 c_i を整数値 $\widetilde{c}_i$ に置き換えるときに行う整数値への丸めにより，情報が落ちるので，加工後の問題例 $\widetilde{I}$ の最適解 $\widetilde{z}$ は元の問題例 I の最適解 z^* とは限らない．$\widetilde{z}$ が問題例 I の $(1-\varepsilon)$-近似解であることを保証するためには，整数値への丸め具合と近似率 $\frac{\sum_{i=1}^{n}\widetilde{c}_i\widetilde{z}_i}{\sum_{i=1}^{n}c_i z_i^*}$ との関係を導く必要がある．この解析はアルゴリズムの説明の後に行う．

ナップサック問題に対する近似スキーム

1. 与えらえた問題例 $I = (\{a_1, a_2, \ldots, a_n\}, \{c_1, c_2, \ldots, c_n\}, b)$ に対して，$a_j \le b, j = 1, 2, \ldots, n$ を仮定する（$a_j > b$ であるアイテムは使えないので除去しておく）．
   $c_{\max} := \max\{c_j \mid j = 1, 2, \ldots, n\}$ とする．

2. 問題例 I から各利得 c_j を次の整数値 $\widetilde{c}_j$ に変更した問題例 $\widetilde{I} = (\{a_1, a_2, \ldots, a_n\}, \{\widetilde{c}_1, \widetilde{c}_2, \ldots, \widetilde{c}_n\}, b)$ を用意する．
$$\widetilde{c}_j := \left\lfloor \frac{n}{\varepsilon \cdot c_{\max}} c_j \right\rfloor,\ j = 1, 2, \ldots, n. \tag{7}$$

3. 問題例 $\widetilde{I}$ の最適解 $\widetilde{z}_j \in \{0, 1\}, j = 1, 2, \ldots, n$ をナップサック問題に対する動的計画法（利得版）で求める．この解 $\widetilde{z}_j, j = 1, 2, \ldots, n$ を元の問題例 I の $(1-\varepsilon)$-近似解として出力する．

これから上記近似スキームの正当性を証明する．

(i) 実行可能性について：$\widetilde{z}_j,\ j=1,2,\ldots,n$ が元の問題例 I において実行可能でなければ意味がないが，問題例 $\widetilde{I}$ を作るときは，$\{a_1,a_2,\ldots,a_n\}$ や b を変更していないので，I と $\widetilde{I}$ は同じ実行可能領域を持ち，$\widetilde{z}_j$ は問題例 I でも実行可能解である．

(ii) 近似率について：$\widetilde{z}_j$ が元の問題例 I の $(1-\varepsilon)$-近似解であること，すなわち，$\mathrm{opt}(\widetilde{I}) \geq (1-\varepsilon)\mathrm{opt}(I)$ を示す．どのアイテム i も $a_i \leq b$ を満たすので，$\mathrm{opt}(I) \geq c_{\max}$ である．
式 (7) において切り下げを外すと以下が成り立つ．

$$\frac{n}{\varepsilon \cdot c_{\max}} c_j \geq \widetilde{c}_j, \qquad j=1,2,\ldots,n, \tag{8}$$

$$\widetilde{c}_j > \frac{n}{\varepsilon \cdot c_{\max}} c_j - 1, \qquad j=1,2,\ldots,n. \tag{9}$$

これらより，問題例 I の最適解を $z_j^* \in \{0,1\},\ j=1,2,\ldots,n$ とすると $\mathrm{opt}(\widetilde{I})$ の下界を以下のように解析できる．

$$\begin{aligned}
\mathrm{opt}(\widetilde{I}) = & \sum_{1\leq j\leq n} c_j \widetilde{z}_j \quad (\text{opt の定義より}) \\
& \geq \frac{\varepsilon \cdot c_{\max}}{n} \sum_{1\leq j\leq n} \widetilde{c}_j \widetilde{z}_j \quad ((8)\text{ より}) \\
& \geq \frac{\varepsilon \cdot c_{\max}}{n} \sum_{1\leq j\leq n} \widetilde{c}_j z_j^* \quad (z^* \text{ が } \widetilde{I} \text{ の実行可能解の一つであることより}) \\
& \geq \sum_{1\leq j\leq n} c_j z_j^* - n \cdot \frac{\varepsilon \cdot c_{\max}}{n} \quad ((9)\text{ より}) \\
& = \mathrm{opt}(I) - \varepsilon \cdot c_{\max} \geq \mathrm{opt}(I) - \varepsilon \cdot \mathrm{opt}(I) \quad (\mathrm{opt}(I) \geq c_{\max}\text{ より}) \\
& = (1-\varepsilon)\mathrm{opt}(I).
\end{aligned}$$

よって，$\widetilde{z}_j$ は元の問題例 I の $(1-\varepsilon)$-近似解である．

(iii) 計算時間について： 動的計画法（利得版）は問題例 $\widetilde{I}$ に対して $O(n\sum_{1\leq j\leq n}\widetilde{c}_i)$ 時間で作動する．ここで，$n\sum_{1\leq j\leq n}\widetilde{c}_i \leq n\sum_{1\leq j\leq n}(\frac{n\cdot c_j}{\varepsilon\cdot c_{\max}}) \leq \frac{n^3}{\varepsilon}$ であるので，計算量は $O(\frac{n^3}{\varepsilon})$ である．

この近似スキームは $n, 1/\varepsilon$ の多項式時間で作動するので，多項式時間近似スキーム (polynomial-time approximation scheme: PTAS) と呼ばれている．

第 1 1 回の講義は以上である．