2003年日本オペレーションズ・リサーチ学会
秋季研究発表会

ナップサック関数の一計算法について

02502630　防衛大学校情報工学科　*藤本 晶子　FUJIMOTO Masako
01700900　防衛大学校情報工学科　山田 武夫　YAMADA Takeo

1　はじめに

ナップサック問題

$$\text{Maximize} \quad \sum_{j=1}^{n} p_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \le c$$

$$x_j \in \{0,1\}, \quad (j = 1, 2, \cdots, n)$$

において，データはすべて正の整数とする．この問題の最適目的関数値を c の関数と考えたものを，**ナップサック関数** [2] といい，$z(c)$ と記す．$z(c)$ は右連続で単調非減少な階段関数なので，その不連続点をすべて求めれば $z(c)$ を計算したことになる．本稿では，区間 $[\underline{c}, \bar{c}]$ で $z(c)$ の不連続点を全列挙するアルゴリズムについて考察する．

2　従来の方法

区間 $[0, \bar{c}]$ の場合には，以下の方法が可能である．

2.1　動的計画法

$z_k(c)$ を，商品 1〜k についてのナップサック関数とすると，

$$z_k(c) = \begin{cases} \max\{z_{k-1}(c), z_{k-1}(c - w_k) + p_k\} & (c \ge w_k \text{の場合}) \\ z_{k-1}(c) & (\text{その他の場合}) \end{cases} \quad (1)$$

が成立する．$z_0(c) \equiv 0$ より初めて，$k = n$ まで計算すると，$z(c) = z_n(c)$ が求めるナップサック関数である．

2.2　Nemhauser, Ullmann の方法 [1]

(1) 式を図示したものが図 1 である．$z_k(\cdot)$ は破線で示した $z_{k-1}(\cdot)$ と，それを (w_k, p_k) だけ平行移動した実線の階段関数（それぞれ，$\mathcal{O}, \mathcal{S}$ で表示）を“縦方向にマージ”して得られる．これらの不連続点が集合 $\mathcal{O}, \mathcal{S}$ に w-座標の大きいものから順に格納されているとすると，この部分のアルゴリズムは次のようになる．

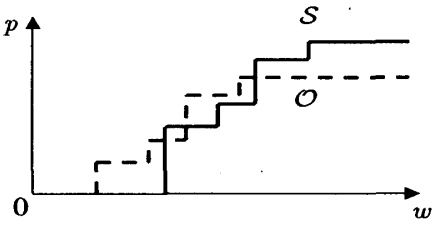

図 1: NU 法

アルゴリズム V_Merge(k)

入力：不連続点の集合 $\mathcal{O}, \mathcal{S}$
出力：$z_k(\cdot)$ の不連続点の集合 $\mathcal{N}$

Step 1. 最初の $O \in \mathcal{O}$，$S \in \mathcal{S}$ をとり出し，$\mathcal{N} \leftarrow \emptyset$ とする．

Step 2. $O_w = S_w = 0$ なら $\mathcal{N} \leftarrow \mathcal{N} \cup \{S\}$ として終了．

Step 3. $O_w \le S_w$ なら Step 4 へ，そうでなければ Step 5 へ．

Step 4. $S_p > O_p$ なら $\mathcal{N} \leftarrow \mathcal{N} \cup \{S\}$ とする．次の $S \in \mathcal{S}$ をとり出し，Step 2 へ．

Step 5. $O_p > S_p$ なら $\mathcal{N} \leftarrow \mathcal{N} \cup \{O\}$ とする．次の $O \in \mathcal{O}$ をとり出し，Step 2 へ．

NU 法は，V_Merge(k) を $k = 1$ から $k = n$ まで反復するもので，k ステップ目の $\mathcal{N}$ が，$k+1$ ステップ目の $\mathcal{O}$ となる．

2.3　問題点

前述の方法はいずれも $\underline{c} = 0$ の場合を対象とするもので，$\underline{c}$ が大きく，区間幅 $\bar{c} - \underline{c}$ が小さい場合には，メモリー所要量が大きく，あまり効率が良くない．

3　下方探索法

任意の整数 $c \ge 0$ に対し，w-座標が c を超えない最大の不連続点を $D(c)$ と記すと，次のアルゴリズム（downward search : DS 法）によって，$[\underline{c}, \bar{c}]$ の不連続点を全列挙できる．

アルゴリズム DS

Step 1. $c \leftarrow \bar{c}$

Step 2. $P \leftarrow D(c)$ を求め，これを出力する．

Step 3. $P_w \leq \underline{c}$ ならば終了．

Step 4. $c \leftarrow P_w - 1$ として Step 2 へ．

このアルゴリズムの動作を図 2 に示す．

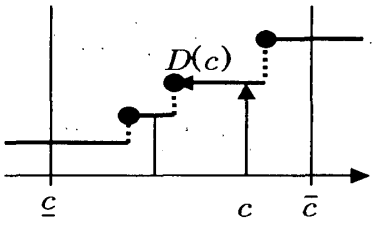

図 2: DS 法

アルゴリズム DS において，最も問題となるのは容量 c が指定されたときに不連続点 $D(c)$ を計算する部分であるが，これは，以下のように実現でき，まず，ナップサック問題を解いて $p = z(c)$ を求め，次いで逆ナップサック問題

$$\text{Minimize} \quad \sum_{j=1}^{n} w_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} p_j x_j \geq p$$

$$x_j \in \{0,1\}, \quad (j = 1, 2, \cdots, n)$$

を解いて，最適値 $w = w(p)$ を得ると，(w, p) が $D(c)$ を与える．逆ナップサック問題は，$y_j = 1 - x_j$ と変数変換すると普通のナップサック問題となるので，この方法では 1 つの不連続点についてナップサック問題を 2 回解くことが必要となる．

しかし，n が大きい場合，ナップサック問題を解くにはある程度時間がかかるので，この回数をさらに削減することが望ましい．このために，次の実験的事実を利用する．すなわち，容量 c に対してナップサック問題を解く際に，Horowitz-Sahni 法を用いると，90% 以上の確率で $D(c)$ が得られる．そこで，容量 c に対して HS 法を解いて得られる点を $H(c)$ とすると，$D(c)$ を求める部分は次のようになる．

$D(c)$ を求めるアルゴリズム

Step 1. $H(c)$ が計算済みでなければ，これを計算する．$P \leftarrow H(c)$.

Step 2. $c' \leftarrow P_w - 1$, $Q \leftarrow H(c')$.

Step 3. $Q_p < P_p$ ならば P を $D(c)$ として出力して終了．そうでなければ，$P \leftarrow Q$ として Step 2 へ．

この方法で，Step 3 の不等式が常に成立するならば，1 つの不連続点ごとに HS を解く回数は 1 回なので，一般に，ナップサック問題を解く回数は不連続点数 $+\alpha$ 程度となる．

4 オンライン・アルゴリズム

商品 $1, 2, \cdots, n$ が 1 度にすべて与えられるのでなく，1 個ずつ逐次出現する場合を考える．このときに，常に最新のナップサック関数 $z_n(c)$ を計算するのがオンライン・アルゴリズムである．NU 法はこの意味でオンライン・アルゴリズムである．DS 法はオンライン・アルゴリズムではないが，NU 法と組み合わせた，次の融合法（Mix 法）はオンライン・アルゴリズムである．

Mix 法（第 k 段）

Step 1. 最初の $O \in \mathcal{O}$, $S \in \mathcal{S}$ をとり出し，$\mathcal{N} \leftarrow \emptyset$ とする．

Step 2. $O_w \leq \underline{c}$, $S_w \leq \underline{c}$ なら，O, S のうち優越するほうを $\mathcal{N}$ に加えて終了．

Step 3. $O_w \leq S_w$ なら Step 4 へ，そうでなければ Step 5 へ．

Step 4. $S_p > O_p$ なら，$\mathcal{N} \leftarrow \mathcal{N} \cup \{S\}$ とする．$S \neq \emptyset$ なら，次の S をとり出し，そうでなければ $S \leftarrow \text{Next}(S)$ として Step 2 へ．

Step 5. $O_p > S_p$ なら，$\mathcal{N} \leftarrow \mathcal{N} \cup \{O\}$ とする．次の $O \leftarrow \mathcal{O}$ をとり出し，Step 2 へ．

上で，O が S に優越するとは，$O_p > S_p$ であるか，$O_p = S_p$, $O_w \leq S_w$ であることをいい，$\text{Next}(S)$ は点 S に対して

$$O^S = \arg\min\{O_w | O_p \geq S_p\}, \quad c' = O^S_w - 1$$

と置いて，$D(c')$ を対応させる関数である．

5 数値実験

数値実験等の詳細については，紙数の関係上当日報告させて頂く．

参考文献

[1] Nemhauser, G. L., Ullmann, Z., "Discreate dynamic programming and capital allocation", *Management Science* Vol. 15 No. 9 (1969), pp.494-505.

[2] 林芳男, 0-1 ナップサック問題の数理とアルゴリズム, 近畿大学商学部 (2000).