2000年度日本オペレーションズ・リサーチ学会
春季研究発表会

マックスミン型多重ナップサック問題の解法

02202770 防衛大学校情報工学科 末澤 浩道 SUEZAWA Hiromichi
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 (knapsack problem: KP) は，組合わせ最適化分野の基本問題の1つで，過去に多くの研究がなされて [1] いるが，本論文ではこれを拡張して**マックスミン型多重ナップサック問題**(max-min multiple KP: M^{3}KP) を定式化し，その解法を検討する．

2 問題の設定

M^{3}KP は以下のように定式化される．

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\min_{1\le i\le m}\sum_{j=1}^{n} p_j x_{ij} & \\
\text{subject to} & \displaystyle\sum_{j=1}^{n} w_j x_{ij} \le c_i, & i=1,2,\cdots,m \\
& \displaystyle\sum_{i=1}^{m} x_{ij} \le 1, & j=1,2,\cdots,n \\
& x_{ij}\in\{0,1\} &
\end{array}
$$

ここで以下の仮定を置く．

仮定 1 p_j，w_j，c_i は全て正の整数である．

仮定 2 商品は相対利得 p_j/w_j の順に番号づけられている．すなわち $p_1/w_1 \ge p_2/w_2 \ge \cdots \ge p_n/w_n$ ．

仮定 3 ナップサックは容量 c_i の順に番号づけられている．すなわち $c_1 \ge c_2 \ge \cdots \ge c_n$.

仮定 4 任意の商品 j について $c_1 \ge w_j$ ．

M^{3}KP は特殊ケースとして KP を含んでいるので，$\mathcal{NP}$-困難である．

3 上界値

M^{3}KP を連続緩和すると線型計画問題が得られる．これより得られる M^{3}KP の上界値を**上界値 1** と呼ぶ．

次に，M^{3}KP の最初の制約式を

$$\sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_{ij} \le \sum_{i=1}^{m} c_i$$

に緩和し，目的関数も

$$\min_{1\le i\le m}\sum_{j=1}^{n} p_j x_{ij} \le \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_{ij}$$

より，右辺に置換し，さらに変数を連続緩和すると

$$
\begin{array}{lll}
\text{maximize} & \displaystyle\sum_{j=1}^{n}\frac{p_j}{m} y_j & \\
\text{subject to} & \displaystyle\sum_{j=1}^{n} w_j y_j & \le \displaystyle\sum_{i=1}^{m} c_i \\
& 0 \le y_j \le 1 &
\end{array}
$$

という（連続型）KP が得られる．これを解いて得られる上界値を**上界値 2** という．これは，M^{3}KP の最初の制約式をラグランジュ緩和して得られる上界値と一致する．

4 近似解法

近似解法は (1) グリーディ法, (2) 2-opt 法, (3) ER 法の3つの方法から成る．このうち (1) は最初の実行可能解を得るための方法で，(2)，(3) は得られた解をさらに改善する方法である

4.1 グリーディ法

M^{3}KP の一つの実行可能解を得るには，商品を逐次収容可能なナップサックに入れて行けば良いが，本稿ではある時点でのナップサック i の総利得，総重量を z_i，W_iとしたとき，評価値 $z_i+\alpha(c_i-W_i)$ が最も小さいナップサックに次の商品を入れる，という方針を採用する．ここに α は現在の利得と残容量のトレード・オフを表すパラメータで，ここでは $\alpha=1.0$ とした．

4.2 2-opt 法

M^{3}KP の実行可能解 $x=(x_{ij})$ に対し，2個以内の商品のナップサック (KS) への割り当てを変更して解を改善することを考える．この場合，以下の4種類の変更が考えられる．なお，どの KS にも入っていない商品を**フリー**な商品という．

(i) KS_i中の商品 j を $\mathrm{KS}_{i'}$へ移す.

(ii) KS_i中の商品 j と $\mathrm{KS}_{i'}$中の商品 j'を入れ替える.

(iii) KS_i中の商品 j を $\mathrm{KS}_{i'}$へ移し, $\mathrm{KS}_{i'}$中の商品 j'をフリーとする.

(iv) フリーな商品 j を KS_i入れ, KS_i中の商品 j'を $\mathrm{KS}_{i'}$へ移す.

4.3 ER 法

簡単な数値実験を行ったところ, 2-opt 法はかなり時間がかかり, 得られる解の質にもやや不満が残るケースが多かった. そこで, 本稿ではナップサックから余分と思われる商品を抜き出し, 利得の低い KS の内容を改善して行く, **抜き取り-再充填法**(Extract-Refill, ER 法) を提案する. この方法の考え方は次のとおりである.

x を $\mathrm{M}^3\mathrm{KP}$ の実行可能解とし, このときの各 KS の利得を $z_i(x)$ とする. これを大きさの順に並べ替え, $z^{(1)}(x) \geq z^{(2)}(x) \geq \cdots \geq z^{(m)}(x)$ とし, 対応する KS を $\mathrm{KS}^{(1)}$, $\mathrm{KS}^{(2)}$, $\cdots$, $\mathrm{KS}^{(m)}$ と記す. $z(x) := z^{(m)}(x)$ が解 x の評価値である. ER 法では $\mathrm{KS}^{(m)}$を除き, $\mathrm{KS}^{(1)}, \mathrm{KS}^{(2)}, \cdots, \mathrm{KS}^{(m-1)}$から順に, 出来るだけ p_j/w_jの大きい商品を抜き出してフリーとし, フリーな商品のプールを拡大して行く. この際各 KS の利得が $z(x) + \delta$ 以下にならないようにする. ここで, δ は $\delta \geq 1$ のパラメータである.

最後に, $\mathrm{KS}^{(m)}$に含まれる商品とフリーな商品を併せ, これらを $\mathrm{KS}^{(m)}$へ入れるナップサック問題を解くと, x より改善された解が得られることが期待される. 以上を反復するのが ER 法である.

5 数値実験

上下界値の算出アルゴリズムを評価するために若干の数値実験を行った. p_j, w_jが独立で $[1, 1000]$ の一様乱数に従う場合と, w_j は上と同じで, p_j を $p_j = w_j + \beta$とした場合を試みた. ここで, β は $[1, 200]$ の一様乱数で, この場合は p_j と w_j が一定の相関をもつことになる. ナップサックの容量は $c_1 = \sum_{j=1}^{n} w_j/m$, $c_i = 0.75c_{i-1} (i \geq 2)$ で, 商品数 $n = 200 \sim 4000$, ナップサック数 $m = 2, 4, 8$ とした. 実験の詳細は当日述べるが, 次の所見を得た.

- ER 法は 2-opt 法よりかなり早い.
- 解の質は, $m = 2, 4$ では 2-opt の方が若干良く, $m = 8$ では ER 法の方が良い. 誤差は商品数が少ない場合は数%で, n が大きくなると誤差は小さくなる.
- 上界値 1 (連続緩和) は上界値 2 より常に良い値を出すが, 計算時間がかかる.
- 上界値 2 はほとんど瞬時に計算されるが, m が大きくなると, 値は劣化する.
- 相関がある場合, 解の質は全ての n, m で ER 法が良かった.
- 相関がある場合, 2-opt 法は独立である場合よりも時間を要した.

6 厳密解法

$\mathrm{M}^3\mathrm{KP}$ の厳密解法はかなり困難である. 現在, 分枝限定法と動的計画法を検討しているが, 次のような問題点を克服する必要がある.

6.1 分枝限定法

前節までの上下界値を用いて $\mathrm{M}^3\mathrm{KP}$ を解く分枝限定法のプログラムを作成したが, 極めて小さい問題しか解けていない. また, $\mathrm{M}^3\mathrm{KP}$ を直接商用コード XPRESS-MP にかけても同様であった.

この原因としては, $\mathrm{M}^3\mathrm{KP}$ の連続緩和解が極度に退化していて, 分枝変数を固定しても上界値がなかなか下がらず, いたずらに子問題が増えていく状況が観察された.

6.2 動的計画法

2 ナップサックの場合を考え各ステップでの状態を (W_1, W_2, P_1, P_2) とする. ここに, W_i, P_i は KS_i の総重量と総利得である. これにより, 動的計画法のアルゴリズムが構成出来るが, 一般には状態数が爆発的に大きくなるので, この「次元ののろい」を克服する必要がある.

参考文献

[1] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley, 1990