固定費付き複数ナップサック問題の上界値

02702090　防衛大学校　*保田　亮　YASUDA Ryo
01107880　防衛大学校　片岡　靖詞　KATAOKA Seiji

1　はじめに

n 個のアイテムと，m 個のナップサックを考える．各アイテム j には価値 p_j と重量 w_j が，また各ナップサック i には重量制限 b^i と固定費 f^i が与えられている．このとき**固定費付き複数ナップサック問題**（Fixed charge Multiple Knapsack Problem: FMKP）は (P) のように定式化できる．

$$(P)\quad \max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_j^i - \sum_{i=1}^{m} f^i y^i \qquad (1)$$

$$\text{s.t.}\quad \sum_{j=1}^{n} w_j x_j^i \le b^i y^i, \quad i=1,\ldots,m \qquad (2)$$

$$\sum_{i=1}^{m} x_j^i \le 1, \qquad j=1,\ldots,n \qquad (3)$$

$$x_j^i, y^i \in \{0,1\} \qquad (4)$$

ここで，x_j^i はナップサック i にアイテム j を入れるとき 1，それ以外で 0 をとる変数，y^i はナップサック i が使用されるとき 1，それ以外で 0 をとる変数である．

FMKP は複数ナップサック問題（MKP）の発展型であるが，ナップサック使用時に固定費が付くことにより，MKP で有効とされてきた (2) 式の代理制約緩和による上界値算法 [2,3] の活用が困難になる．

一方，1995 年以降，Danzig-Wolf の分解原理を整数計画問題に適用した分枝-費用法 (branch-and-price method) の研究 [1] が，一般化割当問題 [4] や固定費付きの問題 [5] などで数々の成功を収めている．本研究では MKP が一般化割当問題の特殊例であることに着目し，**列生成を用いた FMKP の上界値**について検討する．

2　上界値

MKP の緩和問題としてよく用いられている手法には，**線形緩和**: C，**代理制約緩和**: S，**ラグランジュ緩和**: L などがある (アルファベットは各緩和法の略号．最適値には z を用いる)．この節では，これらの緩和問題を FMKP に適用するときの可否について考察する．

2.1　線形緩和 $C(P)$

アイテムが効率順 $(p_i/w_i \ge p_j/w_j, i<j)$ に並んでいるとき，線形緩和のナップサック関数は，この順に重量・価値を累加した図 1 の上側のような凹関数を意味する．また，FMKP では使用するナップサックに対する固定費があるため，これも効率順 $(f^i/b^i \le f^j/b^j, i<j)$ に並べ，コストとして負値側に同様に描くと，これも図 1 の下側のような凹関数になる．FMKP の線形緩和の最適値は，これら 2 つの関数の和（図 1 太線の凹関数）の最大値を求めることに他ならない．

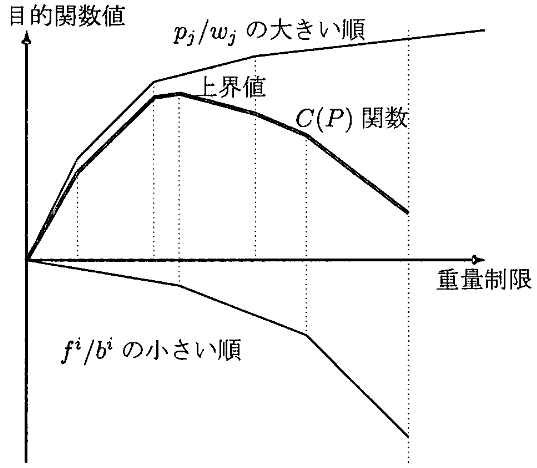

図 1: FMKP の線形緩和ナップサック関数

2.2　代理制約緩和 $S(P)$

代理制約緩和は，ナップサック制約 (2) の各式に正の乗数を掛けて，線形結合したものである．MKP では乗数は全て同じ値が最良と言われているが，同様のことが FMKP でも成立するので，ここでは乗数を 1 とする．

$$S(P)\quad \max \quad \sum_{i=1}^{m}\sum_{j=1}^{n} p_j x_j^i - \sum_{i=1}^{m} f^i y^i$$

$$\text{s.t.}\quad \sum_{i=1}^{m}\sum_{j=1}^{n} w_j x_j^i \le \sum_{i=1}^{m} b^i y^i$$

$$(3), (4)$$

MKP では代理制約緩和が有効であると報告されているが [2,3]，FMKP ではナップサックの重量制限に組合せ的な要因が入るため，最適に解くことが困難であり，緩和問題として活用するには難がある．

2.3　ラグランジュ緩和 $L(P, \boldsymbol{\lambda})$

制約式 (3) に非負の乗数 $(\lambda_1, \ldots, \lambda_n)$ を掛けて目的関数に組み込んだものをラグランジュ緩和という．

$$L(P,\boldsymbol{\lambda})\quad \max \quad \sum_{i=1}^{m}\sum_{j=1}^{n}(p_j-\lambda_j)x_j^i - \sum_{i=1}^{m} f^i y^i + \sum_{j=1}^{n}\lambda_j$$

$$\text{s.t.}\quad (2), (4)$$

これは m 個の独立なナップサック問題に分解でき，ナップサック i の最適解を x_j^{*i} とすると，$L(P,\boldsymbol{\lambda})$ の最適値は (5) 式で示される．$(z)^+$ は $z<0$ のとき 0 である．

$$\sum_{i=1}^{m}\left(\sum_{j=1}^{n}(p_j-\lambda_j)x_j^{*i} - f^i\right)^+ + \sum_{j=1}^{n}\lambda_j \qquad (5)$$

μ_0^s	μ_0^1	μ_1^1	$\cdots$	μ_k^1	$\cdots$	$\mu_{K^1}^1$	μ_0^2	μ_1^2	$\cdots$	μ_k^2	$\cdots$	$\mu_{K^2}^2$	$\cdots\mu_j^s\cdots$		rhs
1	0	$-\boldsymbol{p}\mathcal{X}_1^1 + f^1$	$\cdots$	$-\boldsymbol{p}\mathcal{X}_k^1 + f^1$	$\cdots$	$-\boldsymbol{p}\mathcal{X}_{K^1}^1 + f^1$	0	$-\boldsymbol{p}\mathcal{X}_1^2 + f^2$	$\cdots$	$-\boldsymbol{p}\mathcal{X}_k^2 + f^2$	$\cdots$	$-\boldsymbol{p}\mathcal{X}_{K^2}^2 + f^2$	$\cdots 0 \cdots$	=	0
0	**0**	$\mathcal{X}_1^1$	$\cdots$	$\mathcal{X}_k^1$	$\cdots$	$\mathcal{X}_{K^1}^1$	**0**	$\mathcal{X}_1^2$	$\cdots$	$\mathcal{X}_k^2$	$\cdots$	$\mathcal{X}_{K^2}^2$	$\cdots \boldsymbol{e}_j \cdots$	=	**1**
0	1	1	$\cdots$	1	$\cdots$	1	0	0	$\cdots$	0	$\cdots$	0	$\cdots 0 \cdots$	=	1
0	0	0	$\cdots$	0	$\cdots$	0	1	1	$\cdots$	1	$\cdots$	1	$\cdots 0 \cdots$	=	1

図 2: $m = 2$ のときの単体表（イメージ）

ラグランジュ緩和において，$z(L(P,\boldsymbol{\lambda}))$ の値を最小にするラグランジュ乗数 $\boldsymbol{\lambda}$ を決定するためには，劣勾配法がよく用いられるが，収束までには時間を要する．また，MKP では同じアイテムが複数使われることを許すため，ラグランジュ緩和が効果的だという報告はない．

紹介した 3 つの緩和問題はどれも，仮に実行可能解が得られたとしても，FMKP の実行可能解を導くことはなく，有効な下界値（近似値）の情報にも乏しい．

3　列生成による上界値 $G(P)$

$\mathcal{X}_k^i$ を i 番目のナップサック制約 (2) を満足する（つまり $\boldsymbol{p}\mathcal{X}_k^i \le b^i$）0-1 列ベクトル $(k = 0,\ldots,K^i)$ とする．K^i は一般的に非常に大きな数になる．(P) の $\boldsymbol{x}^i$ は，このベクトル $\mathcal{X}_k^i$ の凸結合で，$\boldsymbol{x}^i = \sum_{k=1}^{K^i} \mu_k^i \mathcal{X}_k^i \left(\sum_{k=1}^{K^i} \mu_k^i = 1, \mu_k^i \ge 0\right)$ のように表現でき，これを (P) に代入すると $G(P)$ のように再定式化（マスター問題）できる．

$$G(P) \quad \max \quad \sum_{i=1}^{m}\sum_{k=1}^{K^i}(\boldsymbol{p}\mathcal{X}_k^i - f^i)\mu_k^i \tag{6}$$

$$\text{s.t.} \quad \sum_{i=1}^{m}\sum_{k=1}^{K^i}\mathcal{X}_k^i\mu_k^i \le \boldsymbol{1} \tag{7}$$

$$\sum_{k=1}^{K^i}\mu_k^i = 1 \quad i = 1,\ldots,m \tag{8}$$

$$\mu_k^i \in \{0,1\} \tag{9}$$

初期実行可能基底が理解しやすいように，$m = 2$ のときの単体表を図 2 に示す．

基底に入れることができる非基底列を組織的に選択するためには，改訂単体法を行う逆行列の目的行が $(1,\beta_1,\ldots,\beta_j,\ldots,\beta_n,\alpha^1,\ldots,\alpha^i,\ldots,\alpha^m)$ のような行ベクトルであるとすれば，次のようなナップサック問題 (Col^i) を解き，$z^i > 0$ になるような $\mathcal{X}^i$ を求めればよい．

$$(Col^i) \quad z^i = \max \quad (\boldsymbol{p} - \boldsymbol{\beta})\mathcal{X}^i - (\alpha^i + f^i) \tag{10}$$

$$\text{s.t.} \quad \boldsymbol{w}\mathcal{X}^i \le b^i \tag{11}$$

$$\mathcal{X}^i \in \{0,1\}^n \tag{12}$$

表 1 は，上界値（$z(C(P))$ を 1.0 とする）と整数解の出現率（%）を，価値・重量の相関度と m の値を変化させた結果である．問題の規模は $n = 50$, $m = 15, 25, 35$ とし，p_j と w_j は共に [1,100] の乱数（無相関）および $w_j = p_j + 10$（強相関），b^i は $\left[0, \frac{n}{m+i}\sum w_j\right]$ の乱数．f^i はナップサックに入る平均的な総価値の 0.8 倍とした．表の各数値は 100 回実験を行った平均値である．

表 1: m の変化による上界値と整数解の出現率

m	15		25		35	
	上界	整数	上界	整数	上界	整数
無相関	0.9956	1%	0.9684	11%	0.9241	35%
強相関	0.9935	0%	0.9518	6%	0.8649	25%

表 1 より，無相関・強相関ともに列生成による上界値が線形緩和による上界値よりも優れた値を得ており，ナップサック数が増えるにつれてもよい値を得ることが分かる．整数解の出現率もナップサック数が増えるにつれ高くなっている．全般に強相関の方が上界値でよい値を得るが，整数解の出現率は無相関の方が高い．

表 2 は $m = 25$ のとき，ナップサックに入る平均的な総価値の 0.5, 0.8, 1.0 倍に固定費を変化させたときの結果である．

表 2: 固定費 f^i の変化による上界値と整数解の出現率

$\frac{\sum f^i}{\boldsymbol{px}}$	0.5		0.8		1.0	
	上界	整数	上界	整数	上界	整数
無相関	0.9626	18%	0.9685	11%	0.9748	9%
強相関	0.9491	14%	0.9518	6%	0.9693	3%

表 2 より，固定費の減少にしたがって，使用するナップサックの数も多くなるため，表 1 の結果と同様，列生成による上界値がよくなり，整数解の割合も増大することが分かる．相関に関しても，強相関に対してよい上界値を得るが，整数解の出現率は無相関の方が高い．

一般化割当問題の列生成法では，生成する列の非ゼロ要素が少ない方が効果的だと報告されているが [4]，この現象は使用するナップサック数が多いほど効果的だという結果と合致する．また，列生成による上界値算法は，問題の規模に比して使用メモリが少なく，$n = 1000$ 程度の問題でも約 40MB のメモリで容易に実行できる．

参考文献

[1] C.Barnhart *et al.*: Branch-and-price — column generation for solving huge integer programs. *Ope. Res.*, **46**(1998)316-329.

[2] H.Kellerer *et al.*: *Knapsack Problems*, (Springer 2004).

[3] S.Martello and P.Toth: *Knapsack Problems* algorithms and computer inplementations, (John Wilay & Sons, 1989).

[4] M.Savelsbergh: A branch-and-price algorithm for the generalized assignment problem. *Ope. Res.*, **45**(1997)831-841.

[5] L.A.Wolsey, *Integer Programming*, (John Wilay & Sons, 1998).