

# 固定費つき複数ナップサック問題の近似解法と厳密解法

Heuristic and Exact Algorithms for the Fixed-Charge Multiple Knapsack Problem

防衛大学校情報工学科  
竹岡 貴裕, 山田 武夫

TAKEOKA Takahiro, YAMADA Takeo

{g44041, yamada}@nda.ac.jp

Department of Computer Science, The National Defense Academy  
Yokosuka, Kanagawa 239-8686, Japan

## 1 はじめに

$n$  個の商品  $1, 2, \dots, n$  があり, 商品  $j$  の重量と利得がそれぞれ  $w_j, p_j$  であるとする. これらを容量がそれぞれ  $c_i$  ( $i = 1, 2, \dots, m$ ) の複数のナップサックに詰め込み, 総利得を最大化する問題は複数ナップサック問題 (multiple knapsack problem: MKP) と呼ばれ, 多数の研究が発表されて来ている [5, 6, 11]. 本稿ではこれに対してナップサックにそれぞれ費用が付随していて, 各ナップサックの使用には対応する費用を支払わなければならない状況を考える. これを固定費付き複数ナップサック問題 (fixed-charge multiple knapsack problem: FCMKP) と呼ぶが, これは数式では以下のように定式化される [13, 14].

FCMKP:

$$\max z(\mathbf{x}, \mathbf{y}) := \sum_{i=1}^m \sum_{j=1}^n p_j x_{ij} - \sum_{i=1}^m f_i y_i \quad (1)$$

$$\text{s.t.} \quad \sum_{j=1}^n w_j x_{ij} \leq c_i y_i, \quad \forall i \in M, \quad (2)$$

$$\sum_{i=1}^m x_{ij} \leq 1, \quad \forall j \in N, \quad (3)$$

$$x_{ij}, y_i \in \{0, 1\}, \quad \forall i \in M, \forall j \in N. \quad (4)$$

ここで  $M = \{1, 2, \dots, m\}$ ,  $N = \{1, 2, \dots, n\}$  はそれぞれナップサックと商品の集合を表し,  $x_{ij}$  は商品  $j$  の採否を,  $y_i$  はナップサック  $i$  の使用・不使用を表す 0-1 型の決定変数である.

ナップサックのコストがすべて 0, すなわち  $f_i \equiv 0$  ( $i \in M$ ) のときは明らかに  $y_i \equiv 1$  (すべてのナップサックを使用) が最適で, 問題は通常の MKP に帰す. 標準的な 0-1 ナップサック問題 (knapsack problem: KP) がすでに  $\mathcal{NP}$ -困難 [3] なので, その拡張である MKP, FCMKP はいずれも  $\mathcal{NP}$ -困難であるが, KP, MKP については過去の研究によりかなり大規模な問題が比較的短時間に解けるようになっている [5, 6].

本稿ではラグランジュ緩和, 釘付けテスト, 分枝限定法などの手法 [8] を用いて FCMKP を厳密に解くことを試みるが, 一般性を失うことなく, 以下では商品とナップサックは以下の仮定を満たすものとする.

$A_1$ : 問題データ  $p_j, w_j$  ( $j \in N$ ),  $c_i, f_i$  ( $i \in M$ ) はすべて正の整数.

$A_2$ : 商品は相対利得の大きい順に  $p_1/w_1 \geq p_2/w_2 \geq \dots \geq p_n/w_n$  と整列済み.

$A_3$ : ナップサックは相対容量の大きい順に  $c_1/f_1 \geq c_2/f_2 \geq \dots \geq c_m/f_m$  と整列済み.

## 2 上下界値

### 2.1 ラグランジュ緩和

$\lambda = (\lambda_1, \lambda_2, \dots, \lambda_m) \geq 0$  に対して, FCMKP のラグランジュ関数を

$$L(x, y; \lambda) := \sum_{i=1}^m \sum_{j=1}^n (p_j - \lambda_i w_j) x_{ij} + \sum_{i=1}^m (\lambda_i c_i - f_i) y_i \quad (5)$$

で定義すると, ラグランジュ緩和問題は

LFCMKP( $\lambda$ ):

$$\begin{aligned} \max \quad & L(x, y; \lambda) \\ \text{s.t.} \quad & (3), (4) \end{aligned}$$

となる. この問題の最適関数値  $\bar{z}_L(\lambda)$  を最小とする問題  $\min\{\bar{z}_L(\lambda) \mid \lambda \geq 0\}$  をラグランジュ双対問題という. このとき, 次が成り立つ.

定理 1

- (i) 任意の  $\lambda \geq 0$  に対して  $\bar{z}_L(\lambda)$  は FCMKP の上界値を与える.
- (ii)  $\bar{z}_L(\lambda)$  は  $\lambda$  について区分的に線形な凸関数である.
- (iii) ラグランジュ双対問題は  $\lambda_1^\dagger = \lambda_2^\dagger = \dots = \lambda_m^\dagger (= \lambda^\dagger)$  であるような最適解  $\lambda^\dagger = (\lambda^\dagger, \lambda^\dagger, \dots, \lambda^\dagger)$  を持つ.

証明: (i), (ii) ラグランジュ緩和の一般論 [2, 8] より容易に導かれる.

(iii)  $k := \arg \min\{\lambda_i\}$  とする. すなわち

$$\lambda_k^\dagger \leq \lambda_i^\dagger, \quad \forall i \in M. \quad (6)$$

$\phi(\cdot)$  を  $\cdot$  の真偽により 1 または 0 をとる真理値関数とすると, LFCMKP( $\lambda^\dagger$ ) の解は

$$\begin{aligned} x_{ij}(\lambda^\dagger) &= \phi(i = k, p_j - \lambda_k^\dagger w_j > 0), \\ y_i(\lambda^\dagger) &= \phi((\lambda_i^\dagger c_i - f_i) > 0) \end{aligned}$$

で与えられ, 最適関数値は

$$\bar{z}_L(\lambda^\dagger) = \sum_{j=1}^n (p_j - \lambda_k^\dagger w_j)^+ + \sum_{i=1}^m (\lambda_i^\dagger c_i - f_i)^+ \quad (7)$$

となる。ここで、 $(\alpha)^+ := \max\{\alpha, 0\}$  で、 $(\lambda_i^+ c_i - f_i)^+$  は  $\lambda_i^+$  について単調非減少関数なので、(6) の条件下で (7) を最小とする  $\lambda^+$  は

$$\lambda_i^+ \equiv \lambda_k^+, \forall i \in M$$

を満たすと考えてよい。 ■

定理 1 より、ラグランジュ緩和問題 LFCMKP( $\lambda$ ) において  $\lambda_i \equiv \lambda$  ( $i \in M$ ) の場合のみを考えれば良いので、問題は

**LFCMKP( $\lambda$ ):**

$$\max \sum_{j=1}^n (p_j - \lambda w_j) x_j + \sum_{i=1}^m (\lambda c_i - f_i) y_i \quad (8)$$

$$\text{s.t. } x_j, y_i \in \{0, 1\}, \forall i \in M, \forall j \in N. \quad (9)$$

と書きかえられる。ここに

$$x_j := \sum_{i=1}^m x_{ij}$$

である。  $\lambda \geq 0$  に対して、LFCMKP( $\lambda$ ) の最適値は

$$\bar{z}_L(\lambda) = \sum_{j=1}^n (p_j - \lambda w_j)^+ + \sum_{i=1}^m (\lambda c_i - f_i)^+ \quad (10)$$

となる。(10) の勾配は ( $\lambda$  で微分可能ならば)

$$d\bar{z}_L(\lambda)/d\lambda = \sum_{\lambda c_i - f_i > 0} c_i - \sum_{p_j - \lambda w_j > 0} w_j$$

なので、 $\bar{z}_L(\lambda)$  を最小とする  $\lambda^+$  は 2 分探索法で容易に求められる。このようにして得られる FCMKP の上界値を  $\bar{z}_L$  と記す。

## 2.2 下界値

Pisinger[10] は、通常の 0-1 ナップサック問題 KP について以下の高速近似解法を提示している。  $p_j, w_j$  ( $j \in N$ ) を商品  $j$  の利得および重量、  $c$  をナップサック容量とし、商品は仮定  $A_2$  に従って番号づけられているとする。  $\bar{p}_j$  ( $\bar{w}_j$ ) で商品  $j$  までの累積利得 (重量) を表すものとし、臨界商品を  $b = \min\{j \mid \bar{w}_j > c\}$  で定義して、前 (後) 向きグリーディ解をそれぞれ下のように定める。

$$z_f = \max_{j=b, \dots, n} \{\bar{p}_{b-1} + p_j : \bar{w}_{b-1} + w_j \leq c\},$$

$$z_b = \max_{j=1, \dots, b-1} \{\bar{p}_b - p_j : \bar{w}_b - w_j \leq c\}.$$

Pisinger は  $z_f$  と  $z_b$  の大きいほうを KP の下界値とすることを提唱している。

本稿ではこれを FCMKP に拡張する。すなわちナップサックの番号順に逐次上の近似解法を適用し、商品をナップサックに詰めていく。この際、ナップサックに収容された商品の総利得がそのナップサックのコスト以下であれば当該ナップサックの使用を取り消し、そこに詰められた商品はすべて解放して次のナップサックに移る。これを最後のナップサックまで繰り返して得られる解を拡張 Pisinger 解と呼び、対応する下界値を  $z$  と記す。

### 3 釘付けテスト

(10) 式を最小とする‘最適’ラグランジュ乗数  $\lambda^\dagger$  と対応する上界値  $\bar{z} := \bar{z}_L(\lambda^\dagger)$ , および前節の方法による下界値  $\underline{z}$  が計算済みであるとする。また, 商品とナップサックのしきい値を以下によりそれぞれ定義する。

$$\theta_j := p_j - \lambda^\dagger w_j, \quad \eta_i := \lambda^\dagger c_i - f_i \quad (11)$$

次に  $\delta$  を 0 または 1 として, FCMKP に制約式  $y_k = \delta$  を付加した問題を  $P(y_k = \delta)$  とする。さらに,  $\lambda^\dagger$  によるそのラグランジュ緩和を次のように導入する。

$\bar{P}(y_k = \delta)$ :

$$\begin{aligned} \max \quad & \sum_{j=1}^n \theta_j x_j + \sum_{i=1}^m \eta_i y_i \\ \text{s.t.} \quad & (9), y_k = \delta \end{aligned} \quad (12)$$

この問題の最適値は明らかに

$$\bar{z}(y_k = \delta) := \sum_{j \in N} \theta_j^\dagger + \sum_{i \neq k} \eta_i^\dagger + \eta_k \delta \quad (13)$$

で, これより直ちに次が従う。

**定理 2** (ナップサックの釘付け) すべての  $k \in M$  に対して

$$(i) \quad \bar{z} - \underline{z} < \eta_k \Rightarrow y_k^* = 1,$$

$$(ii) \quad \bar{z} - \underline{z} < -\eta_k \Rightarrow y_k^* = 0.$$

**証明**: (i) (10), (13) より  $\eta_k \geq 0 \Rightarrow \bar{z}(y_k = 0) = \bar{z} - \eta_k$  であることに注意すると,  $0 \leq \bar{z} - \underline{z} < \eta_k$  より  $\bar{z}(y_k = 0) = \bar{z} - \eta_k < \underline{z}$  を得る。これは最適解では  $y_k = 0$  ではありえないことを意味しているので, (i) が証明された。(ii) も同様に証明される。 ■

同様に, 商品についても次の釘付け定理が成立する。

**定理 3** (商品の釘付け) すべての  $j \in N$  について,

$$(i) \quad \bar{z} - \underline{z} < -\theta_j \Rightarrow x_j^* = 0,$$

$$(ii) \quad \bar{z} - \underline{z} < \theta_j \Rightarrow x_j^* = 1.$$

**注 1** ナップサック問題に対する釘付けテストは [1, 4, 7] などにみられる。本節は最初にラグランジュ緩和を介在させることによって, 同様の手法を FCMKP にも適用可能としたものである。

### 4 厳密解法

FCMKP を解く分枝限定法アルゴリズムの基本的な考え方は, 釘付け後に使用・不使用が未確定のナップサックについて, つまり変数  $y_i$  について, 逐次  $y_i = 1$  または  $y_i = 0$  という条件を付加した‘子問題’に

分割していくことである。これを記述するために以下の問題を定義する。 $F_0$  と  $F_1$  は  $M$  の互いに排反な部分集合 ( $F_0, F_1 \subseteq M, F_0 \cap F_1 = \emptyset$ ) で

$$K_1 \subseteq F_1.$$

を満たすものとする。これらはそれぞれ 0 および 1 に固定されたナップサックの集合を表す。固定されていないナップサックの集合を  $U := M \setminus (F_0 \cup F_1)$  とし、次の部分問題を考える。

$P(F_0, F_1)$ :

$$\begin{aligned} \max \quad & (1) \\ \text{s.t.} \quad & (2), (3), (4), \\ & y_i = 0, \forall i \in F_0, \end{aligned} \tag{14}$$

$$y_i = 1, \forall i \in F_1. \tag{15}$$

第 2 節の  $\lambda^+$  を用いて上の問題をラグランジュ緩和すると

$\bar{P}(F_0, F_1)$ :

$$\begin{aligned} \max \quad & (12) \\ \text{s.t.} \quad & (9), (14), (15) \end{aligned}$$

となる。

これらの問題の最適関数値をそれぞれ  $z^*(F_0, F_1)$ ,  $\bar{z}(F_0, F_1)$  と表記すると、明らかに  $\bar{z}(F_0, F_1)$  は  $z^*(F_0, F_1)$  の上界値となる。すなわち、 $z^*(F_0, F_1) \leq \bar{z}(F_0, F_1)$  で、後者は明らかに

$$\bar{z}(F_0, F_1) = \sum_{j=1}^n \theta_j^+ + \sum_{i \in F_1} \eta_i + \sum_{i \in U} \eta_i^+. \tag{16}$$

である。すると、もし現時点での下界値  $\underline{z}$  が  $\bar{z}(F_0, F_1) < \underline{z}$  を満たしているならば、この部分問題  $P(F_0, F_1)$  を終端することが出来る。

枝払いのための条件としては、上の他に実行可能性と優越性が挙げられる。最初に、子問題  $P(F_0, F_1)$  でのナップサック総容量を  $\bar{c}(F_0, F_1) := \sum_{i \in M \setminus F_0} c_i$  とし、ナップサックに収容することが確定している商品の総重量を  $W_{\text{fix}} := \sum_{j \in I_1} w_j$  とすると、 $W_{\text{fix}} > \bar{c}(F_0, F_1)$  の場合には子問題  $P(F_0, F_1)$  は実行不可能であり、よってこの子問題は終端されることになる。

次に、 $c_i \geq c_{i'}$ ,  $f_i \leq f_{i'}$  かつ  $(c_i, f_i) \neq (c_{i'}, f_{i'})$  のとき、ナップサック  $i$  は  $i'$  に優越するという。子問題  $P(F_0, F_1)$  において、 $i$  が  $i'$  に優越している場合、ナップサック  $i$  を使用せずに  $i'$  を使用することは不合理である。よって、ここで  $i \in F_0$ ,  $i' \in F_1$  の場合、子問題を終端する。 $P(F_0, F_1)$  がこのようなナップサックの対を含む場合、この子問題は不合理であるという。

もし  $P(F_0, F_1)$  がこれらによって終端されず、 $U$  が空集合でない場合には  $U$  に含まれるナップサック  $i$  を選び、 $P(F_0, F_1)$  の子問題  $P(F_0 \cup \{i\}, F_1)$  と  $P(F_0, F_1 \cup \{i\})$  を新たに生成する。 $U = \emptyset$  の場合は  $P(F_0, F_1)$  は末端子問題で、この場合には  $P(F_0, F_1)$  は MKP なので Pisinger のプログラム `multknapp`[11] を用いて最適値  $z^*(F_0, F_1)$  を得ることが出来る。これが現時点での最良値  $\underline{z}$  より大であれば  $\underline{z} \leftarrow z^*(F_0, F_1)$  により下界値を更新する。

以上をまとめ、FCMKP を解くには第 2 節で求めた下界値  $\underline{z}$  と釘付けテストの結果を持って次のアルゴリズムを呼べばよい。ただし、集合は最初  $F_0 := \emptyset$  で、 $F_1$  は 1 と釘付けされたナップサックの集合である。

アルゴリズム B&B( $F_0, F_1$ )

Step 1:  $U := M \setminus (F_0 \cup F_1)$  とする.  $U = \emptyset$  ならば, Step 5 へ.

Step 2:  $W_{\max} > \bar{c}(F_0, F_1)$  または  $P(F_0, F_1)$  が不合理である場合,  $P(F, R)$  を終端して Step 6 へ.

Step 3:  $\bar{z}(F_0, F_1)$  を計算し,  $\bar{z}(F_0, F_1) < z$  ならば  $P(F, R)$  を終端して Step 6 へ.

Step 4: ナップサック  $i \in U$  を選び子問題  $P(F_0 \cup \{i\}, F_1)$  と  $P(F_0, F_1 \cup \{i\})$  を生成する. これらの問題を再帰的に呼び出して実行し, Step 6 へ.

Step 5: (mulknap[11] を用いて) この MKP を解いて最適値  $z^*(F_0, F_1)$  を求める. もし  $z^*(F_0, F_1) > z$  ならば  $z \leftarrow z^*(F_0, F_1)$  により下界値を更新する.

Step 6: 親問題へ戻る.

このアルゴリズムを実装する際には, Step 4 で分枝に用いるナップサック  $i$  の選び方, すなわち分枝戦略と, 子問題からなる 2 分木の走査方法を指定する必要がある. 後者については, 幅優先探索なども考えられるが, 本研究ではメモリーの必要量が少なくすむ深さ優先探索を採用した. 一方, 分枝戦略については次の 2 つを考察した.

分枝戦略 1: ナップサック  $i \in U$  を番号順 (すなわち,  $c_i/f_i$  の大きいものから順) に取り出す. 子問題の走査は  $P(F_0 \cup \{i\}, F_1) \rightarrow P(F_0, F_1 \cup \{i\})$  の順とする.

分枝戦略 2: ナップサック  $i \in U$  をしきい値  $|\eta_i|$  の小さいものから順に取り出す. 子問題の走査は  $\eta_i > 0$  のときは  $P(F_0 \cup \{i\}, F_1) \rightarrow P(F_0, F_1 \cup \{i\})$  の順, そうでないときは  $P(F_0, F_1 \cup \{i\}) \rightarrow P(F_0 \cup \{i\}, F_1)$  の順とする.

## 5 数値実験

前節の B&B アルゴリズムを数値実験によって評価する. アルゴリズムは C 言語で実装し, IBM RS/6000 Model 270 ワークステーション (CPU: POWER3-II SMP 2way, 375MHz) 上で実験を行なった.

### 5.1 実験計画

例題のサイズは  $n = 20 \sim 16000$  および  $m = 5 \sim 50$  で, 下の様式に従ってランダムに生成する.

(a) 商品

- 重量  $w_j$ :  $[1, 1000]$  上の整数一様分布
- 利得  $p_j$ 
  - 無相関 (UNCOR):  $[1, 1000]$  上の整数一様分布, ( $w_j$ ) とは独立.
  - 弱相関 (WEAK):  $[w_j, w_j + 200]$  上の整数一様分布.
  - 強相関 (STRONG):  $p_j := w_j + 20$

(b) ナップサック

- 容量  $c_i = \lfloor 500n \cdot \delta \cdot \xi_i \rfloor$ . ここで  $\xi_i$  ( $i \in M$ ) は  $\{(\xi_1, \dots, \xi_m) \mid \sum_{i=1}^m \xi_i = 1, \xi_i \geq 0\}$  上の一様分布に従い,  $\delta$  は 0.25, 0.50, 0.75 のいずれかとする.

- コスト  $f_i = \rho_i c_i$ , ただし  $\rho_i$  は  $[0.5, 1.5]$  上で (連続) 一様分布.

ここで  $\delta$  はナップサックに収容可能な商品の割合を制御するパラメータで, 商品の平均重量が約 500 なので,  $\delta = 0.50$  の場合は全商品のおよそ半分がナップサックに収容可能ということを示している.

## 5.2 IP ソルバーとの比較

表 1 は  $n = 20 \sim 60$  および  $m = 5$  の小規模な例題について, 商用ソフト NUOPT Ver. 3.3.0 [9] および XPRESS-IVE Ver. 1-16-20 [12] を用いた場合と, B&B アルゴリズムによる計算結果を比較したものである. 各相関タイプと  $n$  の値ごとに 30 例題をランダムに生成し, CPU 時間 600 秒以内に厳密に解けた回数 (#solved) と, (解けた場合についての) 平均計算時間が秒で示されている.

この表より, 商用ソフトではかなり小さい問題でも求解が困難なケースが出現するが, B&B アルゴリズムではこの程度の問題はほとんど瞬時に解いていることが分かる.

表 1: IP ソルバーとの比較 ( $m = 5$ ).

Type	n	NUOPT		XPRESS-IVE		B&B	
		#solved	CPU	#solved	CPU	#solved	CPU
UNCOR	20	30	.65	30	0.10	30	0.01
	30	29	39.85	30	7.76	30	0.00
	40	17	81.50	23	36.68	30	0.01
	50	14	77.02	22	49.67	30	0.00
	60	5	115.12	12	69.29	30	0.00
WEAK	20	29	14.23	30	9.19	30	0.05
	30	15	132.96	20	85.38	30	0.05
	40	8	169.23	13	117.67	30	0.00
	50	3	243.19	5	151.89	30	0.01
	60	0	-	4	70.79	30	0.00
STRONG	20	30	28.38	30	2.84	30	0.02
	30	16	98.58	19	66.21	30	0.03
	40	16	26.27	20	23.26	30	0.01
	50	9	5.07	17	39.17	30	0.00
	60	2	28.91	14	17.45	30	0.01

## 5.3 厳密解

商品数  $n = 1000 \sim 32000$ , ナップサック数  $m = 10 \sim 50$  で, 様々な相関タイプの例題について, 上下界値と釘付けを行い, 分枝限定法による厳密解の計算を試みた. 上下界値と釘付けの計算時間はどのケースも 0.1 秒以下で, ほとんど無視できる程度である. この結果, STRONG で  $m$  が大きく,  $n$  が比較的小さいいくつかの場合を除いて, ナップサックは 90% もしくはそれ以上が 0 または 1 に固定され, UNCOR と WEAK では商品も 1/3 程度が 0 に固定されることが多かった.

B&B は, 少数の例外を除いて数秒以内に FCMKP の厳密解を得ており, この間 MULKNAP を呼び出す回数も通常は数回のみである. 戦略の比較では通常戦略 2 が戦略 1 を上回り, STRONG で  $m$  が大きい場合には特にこの傾向が顕著である.

## 6 むすび

固定費つき複数ナップサック問題を定式化し, これを厳密に解くアルゴリズムを与えた. これは, ラグランジュ緩和と釘付けテスト, および分枝限定法を組み合わせたもので, 末端子問題は複数ナップサック問題 (MKP) であることを利用して既存の MULKNAP プログラムを呼び出して解いている. これに

よって,  $n = 32000$  商品,  $m = 50$  ナップサック程度の問題でも通常の計算機を用いて 10 秒程度で解くことに成功した.

## 参考文献

- [1] R.S. Dembo, P.L. Hammer: "A reduction algorithm for knapsack problems", *Methods of Operations Research*, **36**:49-60, 1980.
- [2] M. Fisher: "The Lagrangian relaxation method for solving integer programming problems," *Management Science* **50**:1861-1871, 2004.
- [3] M.R. Garey, D.S. Johnson: *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, San Francisco, 1979.
- [4] G.P. Ingargiola, F.F. Korsh: "Reduction algorithms for zero-one single knapsack problems," *Management Science*, **20**:460-463, 1973.
- [5] H. Kellerer, U. Pferschy, D. Pisinger: *Knapsack Problems*, Springer, Berlin, 2004.
- [6] S. Martello, P. Toth: *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, Chichester, 1990.
- [7] R.M. Nauss: *Parametric Integer Programming*, Univ. Missouri Press, Columbia, MI, 1979.
- [8] G.L. Nemhauser, L.A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.
- [9] NUOPT Ver. 3.3.0, Mathematical Systems Incorporated, 2002. (<http://www.msi.co.jp/nuopt>)
- [10] D. Pisinger: "An expanding-core algorithm for the exact 0-1 knapsack problem," *European Journal of Operational Research*, **87**:175-187, 1995.
- [11] D. Pisinger: "An exact algorithm for large multiple knapsack problems," *European Journal of Operational Research*, **114**:528-541, 1999. (Source code 'mulknep' available at <http://www.diku.dk/~pisinger/codes.html>)
- [12] XPRESS-IVE Ver. 1.16.20, Dash Associates, 2006. (<http://www.dashoptimization.com>)
- [13] 保田亮, 片岡靖詞: 固定費付き複数ナップサック問題の上界値, OR 学会秋季研究発表会 (東北大学, 2004), 214-215.
- [14] 保田亮: 固定費付き複数ナップサック問題—分枝費用法によるアプローチ—, 修士論文, 防衛大学校理工学研究科情報数理専攻, 平成 17 年 3 月.