情報処理学会論文誌 Vol.56 No.11 2083–2091 (Nov. 2015)

複数ナップサック割当て問題の厳密解法

片岡 靖詞[1,a)]

受付日 2015年4月3日, 採録日 2015年8月12日

概要：本研究では，割当て問題と複数ナップサック問題の両方が融合した問題を扱い，これを**複数ナップサック割当て問題**と呼ぶことにする．この問題は複雑な組合せ構造を持っており，最先端の商用ソルバを用いても，ごく小さなサイズの例題でさえ効率良く解くことが難しい．本論文の目的は，この複数ナップサック割当て問題を，中規模サイズまで最適に解くことができる分枝限定法アルゴリズムを開発することである．この目的のために，上界値を得るための緩和問題および下界値を得るための近似解法を提案するが，これらは両方とも $\mathcal{NP}$-困難性を残した問題を用いている．しかしながら，これらの上下界値は実用上問題ないほど高速に求めることができ，しかも分枝限定法を開発するうえで好都合な性質を持っている．計算機実験では，提案する分枝限定法によってこの問題が効率良く解けることを示す．さらに，提案する上下界値は，特に各ナップサックにある程度の数の商品が詰め込まれるときには，しばしば一致するという結果も得られた．

キーワード：組合せ最適化，割当て問題，複数ナップサック問題

An Exact Algorithm for the Multiple Knapsack Assignment Problem

Seiji Kataoka[1,a)]

Received: April 3, 2015, Accepted: August 12, 2015

Abstract: The problem we are going to discuss is a combined problem of both the assignment problem and the multiple knapsack problem, called *the multiple knapsack assignment problem*. Because of the complicated combinatorial structure, leading commercial software cannot solve this problem effectively even if the instance size is small. Hence, the purpose of this paper is to propose a branch-and-bound algorithm that can solve the problem to optimality up to considerable size. For this purpose, we propose a relaxed problem that gives tight upper bounds and an approximate algorithm that gives tight lower bounds. Though these procedures depend on solving $\mathcal{NP}$-hard problems, they can be solved quickly in practice and give nice properties for developing the branch-and-bound algorithm. Computational experiments show the effectiveness of the branch-and-bound algorithm. Furthermore, we can see that the proposed upper and lower bounds often coincide with each other particularly when a number of items are loaded into each knapsack.

Keywords: combinatorial optimization, assignment problem, multiple knapsack problem

1. はじめに

商品の集合があり，各商品には価値と重量が与えられている．通常のナップサック問題は，重量の総和がナップサック制限を超えないように，価値の総和を最大とするような商品の部分集合を求める問題である [7], [8]．本論文で扱う問題にはさらにいくつかの設定や条件があり，各商品はいくつかのグループに分けられており，ナップサックも複数ある．そして異なるグループに属する商品は，同一のナップサックには入れられないものとする．たとえば冷凍食品のグループと暖かい惣菜のグループは，同じ買物袋には入れられないといったものである．この問題はどのナップサックをどのグループに割り当てるかを決める割当て問題（Assignment Problem: AP）[1], [2] という側面と，グループ内のどの商品をどのナップサックに入れるかを決める複数ナップサック問題（Multiple Knapsack Problem: MKP）[7], [8], [9] という側面の，2 つの問題が融

[1] 防衛大学校情報工学科
Department of Computer Science, National Defense Academy, Yokosuka, Kanagawa 239–8686, Japan
[a)] seiji@nda.ac.jp

合した組合せ構造を持っており，**複数ナップサック割当て問題**（Multiple Knapsack Assignment Problem: MKAP）と呼ぶことにする．

MKAP は Kataoka-Yamada [6] が提唱した問題であり，彼らはこの問題の $\mathcal{NP}$-困難性 [4] を示し，上界値を得るための 3 つの方法を示した．これら 3 つの上界値は線形緩和による上界値と一致する．また彼らは MKAP の下界値を求めるヒューリスティックアルゴリズムを提案した．計算機実験において，大規模な例題に対して高速で高精度な上下界値を得ることに成功している．だが，同じ論文において，彼らは商品数が 50 かそこらの小さな例題では，精度の良い上下界値を得ることが困難であることを示している．特に上界値は貧弱であり，時として最適値の 2 倍にもなることがある．貧弱な上界値は分枝限定法において列挙木の枝を見切ることの困難さを引き起こす．このため，最先端の商用ソルバ，たとえば Gurobi [5] を用いても，この程度の小さな例題が 20 分実行しても解けない．

本論文の目的は，MKAP をかなりのサイズまで厳密に解くことである．この目的のために，より強力な上下界値を得るための戦略を示す．通常，上下界値を得るためには，高速に解くことができる問題を選ぶことが常套手段であり，$\mathcal{NP}$-困難な問題を用いることはない．しかし本研究で用いる上界値戦略，つまり緩和問題には，依然として整数条件を残しているものを用いる．また下界値にも $\mathcal{NP}$-困難な問題 — 複数ナップサック問題を用いる．これらは計算量の理論上は困難な問題ではあるが，実用的な時間で高速に解くことが可能である．したがって，もし得られる値が上下界値として精度の高いものであるならば，緩和問題や実行可能解を得るために $\mathcal{NP}$-困難な問題を用いることも，MKAP のような複雑な問題を扱うためには 1 つの選択肢である．

本論文の構成は以下のとおりである．まず 2 章で MKAP を 0-1 整数計画問題として定式化する．まずは MKAP の難しさを実感するために，小さな例題に対して最先端の商用ソルバ Gurobi 5.0.1 を用いて解く．つづく 3 章では，上界値と下界値を得る戦略について提案する．これらの上下界値に基づいて，MKAP を厳密に解くための分枝限定法アルゴリズムを示す．先ほど述べたように，これらの上下界値を求めるアルゴリズムは $\mathcal{NP}$-困難な問題から構成されている．この選択に至った理由は，その他いくつかの可能な上界値・下界値戦略を試した結果であり，それらを付録にまとめている．4 章では，より大きな例題や様々なパラメータ設定に対して，提案するアルゴリズムの性能を示す．5 章に本論文の成果についてまとめる．

2. MKAP の定式化とその困難さ

商品の集合を N，グループの集合を G とする．グループ k（$k \in G$）に属する商品の集合を N^k（$\bigcup_{k\in G} N^k = N$, $N^{k_1} \cap N^{k_2} = \emptyset$, $k_1 \neq k_2$）とする．また商品 j（$j \in N^k$）の価値と重量をそれぞれ p_j^k，w_j^k とする．さらにナップサックの集合を M とし，ナップサック i の制限を c_i（$i \in M$）とする．MKAP を定式化するために，0-1 決定変数 x_{ij}^k を定め，商品 j（$j \in N^k$）をナップサック i に入れるとき 1，それ以外のとき 0 をとるものとする．さらに 0-1 決定変数 y_i^k を定め，ナップサック i をグループ k に割り当てるとき 1，それ以外のとき 0 をとるものとする．このとき，MKAP は次のように定式化することができる．

$$\text{(MKAP)} \max \sum_{k\in G}\sum_{j\in N^k}\sum_{i\in M} p_j^k x_{ij}^k, \tag{1}$$

$$\text{s.t.} \sum_{j\in N^k} w_j^k x_{ij}^k \le c_i y_i^k, \ \forall i \in M, k \in G, \tag{2}$$

$$\sum_{i\in M} x_{ij}^k \le 1, \qquad \forall j \in N^k, k \in G, \tag{3}$$

$$\sum_{k\in G} y_i^k \le 1, \qquad \forall i \in M, \tag{4}$$

$$x_{ij}^k, y_i^k \in \{0,1\}, \qquad \forall i,\ j,\ k. \tag{5}$$

目的関数 (1) は価値の総量を示す．制約式 (2) はナップサック制限を示し，ナップサック i をグループ k に割り当てるとき，つまり $y_i^k = 1$ のとき，この式は活性化する．制約式 (3) は各商品 j がたかだか 1 つのナップサックに入れられることを示す．同様に制約式 (4) は各ナップサックがたかだか 1 つのグループに割り当てられることを示す．最後に式 (5) は変数の 0-1 制約である．

はじめに，MKAP の小さな例題に対して商用ソルバ Gurobi 5.0.1 を適用した結果について示す．n，g，m をそれぞれアイテム数，グループ数，ナップサック数とする．小さな例題とは $n \in \{40, 60, 80, 100\}$，$g \in \{2, 5\}$，$m \in \{5, 10, 15, 20\}$ とする．商品のグループ分けは，乱数により一様な割合で g 個のグループに割り振る．その他の設定は，Kataoka-Yamada [6] に従った．つまり，商品の価値 p_j^k や重量 w_j^k は，それぞれ独立に [1,1000] の一様整数乱数で与え，ナップサック制限は，約半分の商品が選ばれるように設定した．これらの詳細な設定方法については 4 章で説明する．

表 1 に結果を示す．z^* は最適値を示すが，Gurobi 5.0.1 によって 1,200 秒以内に解けなかった場合は計算を打ち切り，その時点での最良値を用いた．各数値は 10 回試行した平均値である．計算機は DELL Precision T7500（Intel Xeon X5680 (3.3 GHz) × 2）を用い，CPU は計算時間（秒）である．#sol は 10 回の試行のうち 1,200 秒以内に解けた例題の数である．Gurobi で 1,200 秒計算しても解けなかった場合，CPU は 1200 としている．したがって，#sol の値が 10 より小さな場合，もし最適解が出るまで実行を止めなければ，計算時間は表に示されている数値よりも，もっと大きな値になる．

これらの例題は小さいものではあるが，Gurobi で 1,200

表 1 Gurobi を用いた予備実験結果
Table 1 Results of preliminary experiments by Gurobi.

n	g	m	z^*	CPU	#sol
40	2	5	16396.3	2.20	10
		10	16416.5	145.37	10
		15	16256.1	5.56	10
		20	15823.4	1.99	10
	5	5	15327.0	0.14	10
		10	16164.4	2.30	10
		15	16151.4	0.96	10
		20	15709.2	0.49	10
60	2	5	24245.1	125.17	9
		10	24294.6	961.69	2
		15	24244.4	1110.16	1
		20	24196.1	1075.08	2
	5	5	22703.5	0.16	10
		10	24043.8	12.39	10
		15	24134.0	174.82	10
		20	24119.8	111.87	10
80	2	5	32061.5	527.75	6
		10	32121.5	911.77	3
		15	32101.2	1093.37	1
		20	32056.9	1200.00	0
	5	5	30242.0	0.33	10
		10	31926.0	279.35	8
		15	32048.5	918.06	7
		20	31986.8	1200.00	0
100	2	5	39341.1	128.38	9
		10	39368.8	994.42	2
		15	39328.8	1200.00	0
		20	39261.6	1200.00	0
	5	5	37876.3	0.49	10
		10	39168.9	566.65	6
		15	39252.0	1200.00	0
		20	39225.9	1200.00	0

秒実行しても，すべてが解けるとは限らない．Kataoka-Yamada も述べているように，この程度の小さな例題においては，良い実行可能解だけでなく良い上界値を得ることも難しい．また，Gurobi を用いた場合，グループ数が少ないときは，さらに解くことが難しくなっていることも分かる．

3. 上界値・下界値および分枝限定法

3.1 上界値・下界値

複数ナップサック問題（MKP）は MKAP における重要な構成要素である．また，MKP に対しては，Pisinger [9] による優れたアルゴリズムも知られている．Pisinger による MKP のアルゴリズムが効果的である背景には，複数あるナップサック制限の和をとって，単一のナップサック問題として解いたときに選ばれる商品は，高い確率で複数あるナップサックに詰め込むことが可能であるという経験則に基づいている．すなわち，複数あるナップサック制限の総和を，新たにナップサック制限とした通常のナップサック問題における最適値は，MKP の最適値と一致する可能性が高く，もし一致しなくても，その値は MKP の効果的な上界値として利用できる．

同様の性質を MKAP にあてはめることができる．つまり MKAP の緩和問題として，ナップサックをグループに割り当てる問題を基盤とし，同時に各グループに割り当てられたナップサック制限の総和によって，そのグループ内で通常のナップサック問題を解く．いい換えれば，この緩和問題はナップサックの複数性を緩和しているので，複数性緩和問題（Multiplicity-Relaxed Problem: MRP）と呼ぶことにする．MRP は次のように定式化できる．

$$
\begin{aligned}
(\mathrm{MRP})\ \max\quad & \sum_{k\in G}\sum_{j\in N^k} p_j^k x_j^k, \\
\text{s.t.}\quad & \sum_{j\in N^k} w_j^k x_j^k \le \sum_{i\in M} c_i y_i^k, && \forall k\in G, \\
& \sum_{k\in G} y_i^k \le 1, && \forall i\in M, \\
& x_j^k, y_i^k \in \{0,1\}, && \forall i,\ j,\ k.
\end{aligned}
$$

MRP は MKAP の定式化からも導くことができる．すなわち，制約式 (2) に同一の重みを掛けた代理制約緩和を作り，制約式 (3) を取り除く．このとき，いくつかの列が同じものになるが，それらは 1 つだけ残す．このとき，MRP には$\boldsymbol{x}$型の変数があるが，それらにはナップサックに関する添字 i が入っていない．つまり MKAP とくらべて $1/m$ のサイズになっている．MRP は依然として 0-1 計画問題であるが，Gurobi を用いれば，MKAP よりはるかに高速に解くことができる．そして MRP の最適値は MKAP の上界値を与える．以降この上界値を MRUB と呼ぶことにする．

下界値は各グループにおいて MKP を解くことによって得ることができる．これは，上界値を求めるために MRP を解いたとき，$\boldsymbol{y}$型の変数はナップサックとグループの割当てを与えていることを活用する．このときに割り当てられたナップサックを用いて，各グループにおいて MKP を解く．先ほど述べたように，MKP を解くためには Pisigner のアルゴリズムが活用できる．これら MKP の最適値の総和が MRUB と一致したら，これは MKAP の最適解を与える．一致しない場合でも，これら MKP の解は MKAP の実行可能解として下界値を与える．以降この値を MRLB と呼ぶことにする．

1 章で述べたように，通常では上下界値を得るために多項式オーダで解けるような問題を利用する．しかし，本研究では上下界値を得るために MRP あるいは MKP という 2 つの $\mathcal{NP}$-困難な問題を利用している．この疑問に応えるために，付録では他の方法 — たとえば線形緩和や Kataoka-Yamada [6] のヒューリスティック解法など — と

ともに，これらの上下界値の比較結果を示す．これらの結果をふまえて，本論文では MRUB および MRLB を最も期待できる上下界値として採用した．

3.2 分枝限定法

効果的な分枝限定法を開発するためには，適切な分枝変数の選択は重要な戦略の 1 つである．本論文では MRP を緩和問題として用いており，この解には$\boldsymbol{x}$型と$\boldsymbol{y}$型の 0-1 変数がある．このうち解の決定要因として重要なのは$\boldsymbol{y}$型変数であり，その内容に従って$\boldsymbol{x}$型変数の値が決まる．つまり，実行可能解を構成するためには，$\boldsymbol{y}$型変数に対応するナップサックとグループ間の割当てが本質的である．したがって，提案する分枝限定法では，$\boldsymbol{y}$型変数を分枝変数として着目する．具体的には，列挙木のある未分枝頂点において，その先祖ではまだ固定されていない変数のうち，1 の値をとるものを 1 つを選ぶ．

選んだ分枝変数を $y_i^{\hat{k}}$ とするとき，分枝戦略としては 2 分枝ルールを採用し，左子問題では $y_i^{\hat{k}} = 1$ に固定し，右子問題では $y_i^{\hat{k}} = 0$ とする．この分枝戦略では，左子問題は親問題の持つ情報を多く持っているので，緩和問題を解くことなどはスキップすることが可能である．その後の分枝変数の選択においては，着目するグループを順次変えていき，どのグループからもできるだけ均等に分枝変数が選ばれるようにする．

Y_1，Y_0 をそれぞれ 1 に固定する変数と 0 に固定する変数の集合とする．このとき，提案する分枝限定法は，**Algorithm** MKAPBAB(Y_1, Y_0) のように再帰的に記述することができる．メイン関数からは MKAPBAB$(\emptyset, \emptyset)$ を呼ぶ．このアルゴリズムでは，固定された変数のもとでの緩和問題 MRP を MRP(Y_1, Y_0) のように表現する．

3.1 節で示したように，MRP(Y_1, Y_0) を解いたのち，下界値 MRLB を得るために，$\boldsymbol{y}$型変数による割当てに従って定義される複数ナップサック問題 MKP$(\boldsymbol{y})$ を g 回解かなければならない．このために，Pisinger の MKP アルゴリズム `mulknap()` をサブルーチンとして呼ぶ．MRLB は，このとき g 回の MKP$(\boldsymbol{y})$ を解いた最適値の総和であり，gMKP$(\boldsymbol{y})$ のように示す．

Algorithm MKAPBAB(Y_1, Y_0)

Input: Y_1, Y_0
Return: optimal solution (Optimal value is BestLB)
MRUB = MRP(Y_1, Y_0) // which is solved by Gurobi.
if MRP(Y_1, Y_0) is infeasible **then return**
MRLB = gMKP$(\boldsymbol{y})$ // which is solved by Pisinger's `mulknap()`.
if BestLB < MRLB **then**
  BestLB = MRLB
  Preserve the $\boldsymbol{y}$ and the optimal g MKPs solutions.
if BestLB ≥ MRUB or MRLB = MRUB **then return**
Choose a branching variable $y_i^{\hat{k}} \in \{y_i^k | y_i^k = 1, y_i^k \notin Y_1 \cup Y_0\}$
if no such a variable **then return**
MKAPBAB$(Y_1 \cup \{y_i^{\hat{k}}\}, Y_0)$
MKAPBAB$(Y_1, Y_0 \cup \{y_i^{\hat{k}}\})$
return

4. 計算機実験

提案する分枝限定法の性能評価をするために計算機実験を行う．特に商品の数，価値と重量の相関，ナップサック制限についてみていく．アルゴリズムは，ANSI C 言語で記述し，商用ソルバである Gurobi5.0.1 [5] と Pisinger [9] による MKP を解くサブルーチン `mulknap()` を組み込む．計算機は DELL Precision T7500（Intel Xeon X5680（3.3 GHz）× 2）を用いる．

各商品の重量 w_j^k は，[1,1000] の一様整数乱数を用いて与え，その値に応じて価値 p_j^k を以下に示す 3 つの方法によって相関を与える．

- UNCOR（無相関）：p_j^k も $[1, 1000]$ の一様整数乱数
- WEAK（弱相関）：$p_j^k := 0.6w_j^k + \theta$．$\theta$ は $[1, 400]$ の一様整数乱数
- STRONG（強相関）：$p_j^k := w_j^k + 200$

商品のグループ分けは，ほぼ同じ数になるように均等な割合で分配する．ナップサック制限は $c_i := \lfloor \rho(\sum_{k \in G} \sum_{j \in N^k} w_j^k)\xi_i \rfloor$（$i \in M$）のように定める．$\xi_i$ は [0,1) の一様乱数で与えた後，$\sum_{i \in M} \xi_i = 1$ となるように調整した値である．また，ρ はナップサック制限のパラメータであり，$\rho = 0.5$ とした場合は約半分の商品がナップサックに収められる解になることを意味する．

実験結果を**表 2**，**表 3** および**表 4** に示す．表内の各数値は，同じパラメータ設定で 10 回試行した平均値である．最も左の列には，一番注目しているパラメータを示し，表 2 では商品数 n，表 3 では価値と重量の相関，表 4 ではナップサック制限パラメータ ρ である．グループ数 g，ナップサック数 m に続き，CPU は計算時間（秒）であるが，最大 1,200 秒で打ち切っている．そのために#sol に 10 問の試行中 1,200 秒で解けなかった例題数を示している．#bra は分枝限定法における分枝数であり，この値が 0 ということは，分枝限定法に入る前に上界値と下界値が一致し，最適解が得られたことを意味する．この一致は比較的頻繁に起こり，それを示すために#LB=UB には 10 回の試行中，上下界値が一致して解けた例題数を示している．z^* は最適値であり，残る 4 つの列は上下界値と最適値との絶対誤差を示しており，本研究で用いた MRUB，MRLB のほかに比較対象として，線形緩和による上界値 LPUB，Kataoka-Yamada [6] のヒューリスティック解法による下界値 KYUB を併記した．

表 2 問題のサイズに関する結果：UNCOR，$\rho = 0.5$
Table 2 The results for various sizes instances: UNCOR, $\rho = 0.5$.

n	g	m	CPU	#sol	#bra	#LB=UB	z^*	LPUB$-z^*$	MRUB$-z^*$	z^*-KYLB	z^*-MRLB
250	2	5	0.05	10	0.0	10	100448.8	656.53	0.0	314.2	0.0
		10	0.05	10	0.0	10	101066.1	37.41	0.0	45.6	0.0
		15	0.07	10	0.0	10	101077.8	23.13	0.0	62.4	0.0
		20	0.20	10	7.2	9	101076.2	23.29	0.1	56.7	0.0
	5	5	0.74	10	0.0	10	94337.2	6768.13	0.0	8849.8	0.0
		10	2.22	10	5.4	9	100773.1	330.41	2.1	613.6	4.0
		15	360.72	9	48.4	8	101002.5	98.43	1.1	357.4	7.3
		20	697.54	5	6.7	7	101048.8	50.69	17.2	360.8	16.3
500	2	5	0.07	10	0.0	10	202157.5	109.90	0.0	86.6	0.0
		10	0.07	10	0.0	10	202238.5	27.03	0.0	34.1	0.0
		15	0.07	10	0.0	10	202240.5	23.22	0.0	41.3	0.0
		20	0.10	10	0.0	10	202242.6	19.76	0.0	34.6	0.0
	5	5	0.68	10	0.0	10	190590.3	11677.10	0.0	3156.1	0.0
		10	4.07	10	0.0	10	201966.7	298.83	0.0	338.1	0.0
		15	199.22	10	0.0	10	202176.2	87.52	0.0	194.0	0.0
		20	944.52	3	0.0	10	202215.0	47.36	0.0	214.1	0.0
750	2	5	0.14	10	0.0	10	303212.8	307.91	0.0	218.1	0.0
		10	0.20	10	2.0	9	303491.3	27.28	0.0	19.9	1.6
		15	0.07	10	0.0	10	303489.9	27.03	0.0	24.8	0.0
		20	0.10	10	0.0	10	303490.0	25.09	0.0	19.4	0.0
	5	5	8.13	10	0.0	10	290207.9	13312.81	0.0	12061.2	0.0
		10	12.78	10	6.6	9	303121.1	397.48	0.4	701.1	4.2
		15	133.84	10	0.0	10	303428.1	88.83	0.0	209.5	0.0
		20	1110.29	1	0.0	10	303462.3	52.79	0.0	194.7	0.0
1000	2	5	0.08	10	0.0	10	406364.5	47.14	0.0	179.7	0.0
		10	0.09	10	0.0	9	406382.3	27.40	0.0	15.3	0.5
		15	0.07	10	0.0	9	406386.9	20.82	0.0	12.3	0.1
		20	0.08	10	0.0	6	406381.7	23.16	1.1	3.9	1.9
	5	5	28.17	10	0.0	10	387736.2	18675.44	0.0	16292.7	0.0
		10	7.04	10	0.0	10	406168.7	241.00	0.0	667.0	0.0
		15	102.50	10	0.0	10	406332.6	75.12	0.0	183.1	0.0
		20	806.23	4	0.0	10	406355.6	49.26	0.0	134.7	0.0

計算時間は 1,200 秒で打ち切っているが，Gurobi を用いて MRP を解く過程，および `mulknap()` を用いて MKP を解く過程には，打ち切り制限を設けていない．これは，本論文で示す上下界値の強さを示すためである．したがって，MRP および MKP を解くときに 1,200 秒以上要したが，その結果として上下界値が一致した場合，CPU は 1,200 秒に設定し，#sol はカウントしないが，#LB=UB の方はカウントしている．

4.1 商品の数について

表 2 は商品数 n を変えた実験結果である．n の候補は $\{250, 500, 750, 1000\}$ とし，グループ数 g は $\{2, 5\}$ から，ナップサック数 m は $\{5, 10, 15, 20\}$ から選んだ．価値と重量は無相関を，ナップサック制限パラメータ ρ は 0.5 とした．

この結果より，商品数が増加しても最適解を求めるまでの計算時間はほとんど衰えていないことが分かる．特にグループ数が 2 のときは，実験で用いたすべての例題において瞬時に最適解を得ている．この現象は Gurobi を用いて解いた結果（表 1）とは正反対である．この実行効率の良さは，採用した上界値 MRUB と下界値 MRLB の効果に拠るところが大きく，多くの例題で一致している．しかし，ナップサック数 m が増えてくると，#sol<#LB=UB となる場合も散見される．これは上下界値を求めるためだけで 1,200 秒を超えてしまったことを意味しているが，そのような場合でも求められた上下界値は多くの試行で一致しており，最適性を保証している．

一方，LPUB と KYLP は，最適値と比べるとかなりの誤差がある．これほどの誤差があると，これらの上下界値を分枝限定法に使ったとしても，列挙木を効果的に刈り取ることは難しいであろう．MRP や MKP を解くことは，理論的には $\mathcal{NP}$-困難性の壁はあるが，実際には高速に解け

表 3 重量と価値の相関に関する結果：$n = 500$, $\rho = 0.5$
Table 3 The results for cor-relations: $n = 500$, $\rho = 0.5$.

Cor-relation	g	m	CPU	#sol	#bra	#LB=UB	z^*	LPUB$-z^*$	MRUB$-z^*$	z^*-KYLB	z^*-MRLB
UNCOR	2	5	0.07	10	0.0	10	202157.5	109.90	0.0	86.6	0.0
		10	0.07	10	0.0	10	202238.5	27.03	0.0	34.1	0.0
		15	0.07	10	0.0	10	202240.5	23.22	0.0	41.3	0.0
		20	0.10	10	0.0	10	202242.6	19.76	0.0	34.6	0.0
	5	5	0.68	10	0.0	10	190590.3	11677.10	0.0	3156.1	0.0
		10	4.07	10	0.0	10	201966.7	298.83	0.0	338.1	0.0
		15	199.22	10	0.0	10	202176.2	87.52	0.0	194.0	0.0
		20	944.52	3	0.0	10	202215.0	47.36	0.0	214.1	0.0
WEAK	2	5	0.12	10	0.0	10	156620.3	37.34	0.0	53.9	0.0
		10	0.15	10	0.0	10	156637.2	17.92	0.0	12.4	0.0
		15	0.10	10	0.0	10	156636.8	16.08	0.0	16.6	0.0
		20	0.14	10	0.0	10	156638.5	12.83	0.0	16.5	0.0
	5	5	134.45	10	0.0	10	151940.5	4717.14	0.0	4640.2	0.0
		10	59.71	10	4.0	9	156496.4	158.72	1.9	203.8	0.1
		15	585.20	9	0.3	9	156586.3	66.58	6.6	128.2	0.0
		20	1200.00	0	0.0	10	156610.6	40.73	0.0	112.5	0.0
STRONG	2	5	601.45	5	0.0	10	195878.2	333.62	0.0	200.0	0.0
		10	0.07	10	0.0	10	196115.2	93.54	0.0	119.4	0.0
		15	0.09	10	0.0	10	196113.0	92.27	0.0	80.0	0.0
		20	0.08	10	0.0	9	196111.5	91.85	0.0	140.2	0.1
	5	5	1200.00	0	0.0	10	187104.6	9107.22	0.0	3793.8	0.0
		10	841.31	3	0.0	10	195952.9	255.84	0.0	397.3	0.0
		15	489.90	6	0.0	10	196044.0	161.27	0.0	371.2	0.0
		20	513.83	6	0.0	10	196049.4	153.95	0.0	382.8	0.0

表 4 ナップサック制限パラメータに関する結果 ρ：$n = 500$，UNCOR
Table 4 The results for knapsack capacity parameter ρ: $n = 500$, UNCOR.

ρ	g	m	CPU	#sol	#bra	#LB=UB	z^*	LPUB$-z^*$	MRUB$-z^*$	z^*-KYLB	z^*-MRLB
0.25	2	5	0.10	10	0.0	10	142899.6	89.00	0.0	53.5	0.0
		10	0.11	10	0.0	10	142952.3	32.30	0.0	54.4	0.0
		15	0.11	10	0.0	10	142958.7	23.38	0.0	39.8	0.0
		20	1.77	10	43.2	9	142958.6	20.59	0.0	37.6	5.4
	5	5	4.12	10	0.0	10	137347.1	5641.50	0.0	6293.6	0.0
		10	5.47	10	0.0	10	142744.8	239.80	0.0	324.7	0.0
		15	488.99	8	0.0	10	142896.8	85.28	0.0	297.9	0.0
		20	1104.98	1	27.7	9	142914.2	64.99	10.7	228.0	3.0
0.50	2	5	0.07	10	0.0	10	202157.5	109.90	0.0	86.6	0.0
		10	0.07	10	0.0	10	202238.5	27.03	0.0	34.1	0.0
		15	0.07	10	0.0	10	202240.5	23.22	0.0	41.3	0.0
		20	0.10	10	0.0	10	202242.6	19.76	0.0	34.6	0.0
	5	5	0.68	10	0.0	10	190590.3	11677.10	0.0	3156.1	0.0
		10	4.07	10	0.0	10	201966.7	298.83	0.0	338.1	0.0
		15	199.22	10	0.0	10	202176.2	87.52	0.0	194.0	0.0
		20	944.52	3	0.0	10	202215.0	47.36	0.0	214.1	0.0
0.75	2	5	0.05	10	0.0	10	237280.0	68.96	0.0	181.6	0.0
		10	0.05	10	0.0	10	237325.9	22.29	0.0	73.7	0.0
		15	0.04	10	0.0	10	237326.8	20.60	0.0	22.3	0.0
		20	0.04	10	0.0	10	237330.1	16.36	0.0	24.3	0.0
	5	5	0.32	10	0.0	10	216504.3	20844.66	0.0	7506.0	0.0
		10	2.10	10	0.0	10	236927.4	420.79	0.0	485.1	0.0
		15	58.96	10	5.2	9	237282.9	64.50	2.2	269.5	4.8
		20	707.92	7	0.0	10	237313.2	33.26	0.0	261.1	0.0

る場合が多く，非常に精度の良い上下界値を出してくるので，MKAP のような複雑な組合せ最適化問題を解くためには意味のある選択であろう．

ただ，商品の数に関しては，$n = 250$ のときに上下界値が一致しない場合が目立つ．これは単一のナップサック問題を解いて選ばれた商品が，複数のナップサックに詰め込められないことを意味している．この現象は商品数が少ないときの方が頻繁に起こりやすく，Pisinger も同様のことを考察している．実際に，表 1 で用いた小さい例題では，Gurobi では解けても，本アルゴリズムでは解けない場合もいくつかあった．

4.2 価値と重量の相関について

表 3 は，価値と重量の相関の影響をまとめたものである．この実験では商品数は 500，ナップサック制限パラメータ ρ は 0.5 に固定して行った．

多くのナップサックタイプの問題では，価値と重量の相関が強いほど解き難くなることが多く，MKAP でも同様の傾向が観察できる．しかし，MRUB と MRLB による上下界値は，強相関の場合でも相変わらず優れた精度の値を算出している．このことは，計算にとって手間のかかるのは，MKAP の分枝限定法ではなく，MRP あるいは MKP を解く部分にあることを意味している．また，Pisinger の MKP のアルゴリズムは，ナップサックの数が多くなると解き難くなると報告されており，無相関や弱相関の場合でも，ナップサック数が多いところに解き難い例題が偏在していることも分かる．

4.3 ナップサック制限パラメータについて

表 4 は，ナップサック制限パラメータ ρ の振舞いについてみたものである．ここでは，商品数 500 で無相関の場合に固定して調べた．

多くのナップサックタイプの問題では，おおよそ半分の商品が選ばれるような中くらいのナップサック制限を持つ例題が難しいといわれる．しかし，MKAP の場合，ナップサック制限に関しては明確な特徴が観察されなかった．ナップサックに関しては，制限よりもその数の方に影響力がある．また Pisinger は，MKP のアルゴリズムは多くの商品が 1 つのナップサックに詰め込まれる状況において効率的だと述べている．このことを勘案すれば，ナップサック数が多く，さらにグループ数も多いとき，1 つのナップサックに入る商品数も少なくなる傾向になり，計算時間も急速に悪くなっていくことが観察される．さらに，このような場合には，上下界値が一致しなくなる可能性も高くなり，アイテム数が少なくても提案するアルゴリズムでは解けなくなる場合も出てくる．

5. 結論

本研究では，複数ナップサック割当て問題（MKAP）という，割当て問題と複数ナップサック問題という 2 つの問題を構造に含む複雑な組合せ最適化問題を扱った．複雑な組合せ構造を持つために，Gurobi のような最先端の商用ソルバを用いても，小さな問題でさえも解くことが困難である．

MKAP を厳密に解くため，本研究ではナップサックの複数性を緩和した問題を解いて上界値を得る戦略を採用した．また下界値を求めるために複数ナップサック問題を活用した．提案する手法では上下界値を得るために $\mathcal{NP}$-困難な問題を解かなければならない．従来では，上下界値を得るために $\mathcal{NP}$-困難な問題を解くような戦略は敬遠されたであろう．しかし，今日では $\mathcal{NP}$-困難な問題であっても，実用上問題ないほど高速に解ける場合も少なくない．そして，複雑な問題である MKAP の場合，これらの問題を最適に解くことは，選択肢の 1 つとして有用である．実際に，これらの問題を解くことで得られる上下界値は，様々なタイプの例題に対しても精度が良く，上下界値が一致することもしばしば観察された．その結果，これまで解くことができなかったようなサイズの例題でも，本研究では最適に MKAP を解くことに成功した．特にグループ数が比較的少ないときには，開発したアルゴリズムは非常に効率良く機能する．この傾向は，Gurobi を用いた場合とは正反対であるが，提案アルゴリズムの特性を物語っている．

参考文献

[1] Ahuja, R.K., Magnanti, T.L. and Orlin, J.B.: *Network Flows – Theory, Algorithms, and Applications*, Prentice Hall, Englewood Criffs (1993).

[2] Burkard, R.E., Dell'Amico, M. and Martello, S.: *Assignment Problems*, SIAM, Philadelphia (2009).

[3] Desaulniers, G., Desrosiers, J. and Solomon, M.M.: *Column Generation*, Springer (2005).

[4] Garey, M.R. and Johnson, D.S.: *Computers and Intractability – A Guide to the Theory of NP-Completeness*, Freeman and Company, San Francisco (1979).

[5] Gurobi Optimizer (online), available from ⟨http://www.gurobi.com⟩.

[6] Kataoka, S. and Yamada, T.: Upper and Lower Bounding Procedures for the Multiple Knapsack Assignment Problem, *European Journal of Operational Research*, Vol.237, pp.440–447 (2014).

[7] Kellerer, H., Pferschy, U. and Pisinger, D.: *Knapsack Problems*, Springer, Berlin (2004).

[8] Martello, S. and Toth, P.: *Knapsack Problems – Algorithms and Computer Implementations*, John Wiley & Sons, Chichester (1990).

[9] Pisinger, D.: An Exact Algorithm for Large Multiple Knapsack Problems, *European Journal of Operational Research*, Vol.114, pp.528–541 (1999).

付　　録

A.1 その他の上下界値

付録では，提案する上下界値に至る前に試した MKAP に適用可能ないくつかの緩和問題（上界値）や近似解法（下界値）をまとめる．これらの結果をふまえて，本研究では提案する上下界値を選択した．

最初に試した緩和は，MKAP の定式化に対する線形緩和であり，最も標準的に使われる緩和である．計算時間は高速であり，表 1 に用いた程度の例題であれば 0.01 秒も要さない．

次に試した緩和問題は，いくつかの制約を追加したものである．Gurobi の標準的なメッセージをみると，Gurobi が分枝限定法に入るためにどのような前処理をしているかがおおかた予想がつく．制約を追加して線形計画法を解くことにより，上界値は次第に強化される．またそのときの計算時間は無視できるほど小さい．このような上界値を得るために，Gurobi の分枝頂点数の上限を 1 に設定する．このように設定することで，Gurobi は分枝限定法に入った瞬間に強化された上界値を返してくる．

3 つめに試した緩和は，列生成法によるものである．列生成法は，多くの複雑な組合せ最適化問題に適用でき，精度の良い上界値を得るだけでなく，分枝価格法に持ち込んで最適に解くことにも成功している [3]．まず，MKAP のマスタ問題について説明する．マスタ問題として，各列が $(a_1, \cdots, a_j, \cdots, a_n, b_1, \cdots, b_i, \cdots, b_m)^\top$（$j \in N, i \in M$）という構造を持つ問題を考える．また $\alpha_j^{\hat{k}}$（$j \in N^{\hat{k}}$）をグループ $\hat{k}$ においてナップサック制限 c_i のナップサック問題の実行可能解とする．このとき，対応する列は，次のように構成する：$a_j = \alpha_j^{\hat{k}}$（$j \in N^{\hat{k}}$），$a_j = 0$（$j \in N^k$，$k \neq \hat{k}$），

表 A·1 上下界値の相対誤差
Table A·1 Relative errors of upper and lower bounds.

			ErrU (%)				ErrL (%)		
n	g	m	LPUB	CPUB	CGUB	MRUB	KYLB	CGLB	MRLB
40	2	5	1.24	1.22	1.10	**0.11**	0.90	3.27	**0.08**
		10	1.11	1.06	0.73	**0.52**	1.35	**0.35**	1.35
		15	2.09	1.69	**0.60**	1.51	3.47	**0.04**	2.33
		20	4.87	2.06	**0.15**	4.29	5.39	**0.09**	4.84
	5	5	8.31	7.80	0.50	**0.08**	11.28	**0.22**	0.29
		10	2.69	2.46	**0.91**	1.07	7.79	**0.03**	3.43
		15	2.75	2.01	**0.35**	2.05	10.67	**0.13**	9.92
		20	5.63	2.19	**0.27**	5.04	14.09	**0.05**	14.21
60	2	5	0.63	0.63	0.62	**0.00**	0.50	9.63	**0.00**
		10	0.42	0.41	0.40	**0.18**	0.36	2.13	**0.08**
		15	0.62	0.59	0.49	**0.39**	0.70	**0.06**	0.61
		20	0.82	0.72	**0.41**	0.59	1.03	**−0.03**	1.18
	5	5	7.47	7.33	0.09	**0.00**	13.32	0.01	**0.00**
		10	1.47	1.46	0.80	**0.48**	4.62	**0.94**	1.54
		15	1.08	1.02	**0.53**	0.61	5.34	**0.22**	4.08
		20	1.13	0.98	**0.30**	0.89	5.89	**0.18**	6.01
80	2	5	0.52	0.52	0.52	**0.02**	0.30	15.61	**0.00**
		10	0.32	0.31	0.31	**0.13**	0.32	9.48	**−0.04**
		15	0.38	0.37	**0.35**	1.14	0.27	0.77	**−0.98**
		20	0.52	0.49	0.44	**0.38**	0.32	**−0.22**	0.01
	5	5	6.57	6.50	**0.00**	0.13	7.59	**0.00**	**0.00**
		10	0.94	0.92	0.72	**0.12**	3.47	2.07	**0.63**
		15	0.55	0.52	0.41	**0.29**	3.52	**0.43**	1.79
		20	0.74	0.70	**0.49**	0.56	2.51	**−0.05**	2.83
100	2	5	0.33	0.33	0.33	**0.00**	0.13	19.21	**0.00**
		10	0.25	0.25	0.25	**0.11**	0.14	16.32	**−0.06**
		15	0.35	0.34	0.34	**0.23**	0.04	5.29	**−0.12**
		20	0.51	0.50	0.50	**0.40**	−0.08	0.60	**−0.18**
	5	5	4.21	4.20	0.03	**0.00**	7.96	0.06	**0.00**
		10	0.76	0.76	0.65	**0.05**	4.54	3.59	**0.20**
		15	0.54	0.54	0.49	**0.32**	1.35	1.29	**0.75**
		20	0.61	0.59	0.53	**0.48**	1.49	**−0.08**	1.29

$b_i = 1$ $(i = \hat{i})$，$b_i = 0$ $(i \neq \hat{i})$．これらの列をとるか否かを決める 0-1 変数を用意し，各行とも符号および右辺定数は 1 以下とする．また目的関数の係数は，$\sum_{j \in N^{\hat{k}}} p_j^{\hat{k}} \alpha_j^{\hat{k}}$ とする．

このような列をすべて列挙することは困難であるが，いくつかの列の集合は容易に準備できる．この列集合におけるマスタ問題が線形計画法として最適に解けているとき，新たに追加すべき列は次のように生成する．現時点でのマスタ問題において s_j^k を N^k に属する商品 j の双対変数，t_i をナップサック i に関する制約の双対変数とする．このとき，新しく追加すべき列は，グループ k，ナップサック i に対して次のナップサック問題 KP_i^k を解いて生成できる．

$$\max \left\{ \sum_{j \in N^k} (p_j^k - s_j^k) \alpha_j^k \middle| \sum_{j \in N^k} w_j^k \alpha_j^k \leq c_i, \alpha_j^k \in \{0, 1\} \right\}.$$

もし，KP_i^k の目的関数値が t_i より大きな実行可能解が見つかれば，対応する列がマスタ問題に追加される．すべてのグループとナップサックのペアにおいて，そのような解が見つからなければ，マスタ問題は線形計画法として最適に解けたことを意味し，上界値が得られる．下界値は，最終的なマスタ問題を 0-1 計画法として解けば MKAP の実行可能解が得られる．

表 A·1 にこれらの上界値・下界値の評価を示す．用いた例題は表 1 と同じものであり，各上界値の略号として，LPUB は線形緩和，CPUB は Gurobi の前処理で制約を追加して得たもの，CGUB は列生成法で得たものである．そして MRUB が本論文の 3 章で説明したものである．また下界値の略号として，KYLB は Kataoka-Yamada [6] によるヒューリスティック解法，CGLB は列生成法において追加する列がなくなったときのマスタ問題を 0-1 計画法で解いたもの，MRLB が本論文の 3 章で説明したものである．

これらの上下界値を表 1 における最適値（求められなかった場合は最良値）z^* との相対誤差，つまり ErrU は $100(\mathrm{xxUB} - z^*)/z^*$ であり，ErrL は $100(z^* - \mathrm{xxLB})/z^*$ で評価した．Gurobi は 1200 秒で実行を打ち切っているため，そのときの z^* は最良値になっている．したがって，z^* は xxLB よりも小さくなっている場合もある．そのために表 A·1 の ErrL には，マイナスの値が表示されることもある．

表 A·1 において太字は，4 つの上界値，3 つの下界値の中で最も良い値が出たところである．商品数 n が少ないときは列生成法による上下界値は最も優れているが，n が大きくなると MRUB あるいは MRLB が多くの場合で一番良い値を出すようになる．この結果をふまえて，本研究では上界値として MRUB を，下界値として MRLB を採用した．

片岡 靖詞 （正会員）

防衛大学校情報工学科准教授．1985 年早稲田大学理工学部工業経営学科卒業，1987 年同大学大学院修士課程修了，1990 年同大学院博士課程満期退学．1990 年防衛大学校情報工学科．1993 年博士（工学）．各種の組合せ最適化アルゴリズム開発に従事．日本オペレーションズ・リサーチ学会，INFORMS 各会員．