

一般化ナップサック共有問題の厳密解法 (飯田 耕司教授に捧ぐ)

藤本 晶子* 山田武夫** 片岡 靖詞***
渡辺宏太郎****

(平成 15 年 3 月 14 日受付; 平成 15 年 4 月 23 日受理)

An Exact Algorithm for the Generalized Knapsack Sharing Problem
(Dedicated to Prof. K. IIDA)

By Masako FUJIMOTO*, Takeo YAMADA**, Seiji KATAOKA*** and
Kohtaro WATANABE****

概要: We are concerned with a variation of the knapsack problem as well as of the knapsack sharing problem, where we are given a set of n items and a knapsack of a fixed capacity. As usual, each item is associated with its profit and weight, and the problem is to determine the subset of items to be packed into the knapsack. However, in the problem there are s players and the items are classified into $s+1$ disjoint groups, N_i ($i = 0, 1, \dots, s$). The player k is concerned only with the items in $N_0 \cup N_k$, where N_0 is the set of *common* items. We formulate the generalized knapsack sharing problem as the one to maximize the minimum of the profits over all the players. An algorithm is developed to solve this problem to optimality, and through a series of computational experiments, we evaluate the performance of the developed algorithm.

Key words : knapsack problem, knapsack sharing, combinational optimization, exact algorithm.

1. はじめに

ナップサック問題 (knapsack problem: **KP**)^{6,13} はオペレーションズ・リサーチや情報工学における基本問題の 1 つで, 様々な研究が行われ, 今日では数十万変数の問題がさほど困難なく解かれるようになっているが, 本稿ではこの問題の 1 つの拡張を考察する. すなわち, s 人のプレイヤーが n 個の商品の集合 $N := \{1, 2, \dots, n\}$ のうちのいくつかを容量 C のナップサックに収容するものとする. 各商品 $j \in N$ には重量 w_j と利得 p_j が付与されていて, 商品の集合 N は共通商品群 N_0 と s 個の個別商品群 N_1, N_2, \dots, N_s に分かれている. すなわち,

$$\bigcup_{k=0}^s N_k = N, \quad N_k \cap N_l = \emptyset \quad (k \neq l) \quad (1.1)$$

* 防衛大学校 第 41 期理工学研究科学生 情報数理専攻

** 防衛大学校 情報工学科 教授

*** 防衛大学校 情報工学科 助教授

**** 防衛大学校 情報工学科 助手

x_j を商品 j をナップサックに入れるとき 1, そうでないとき 0 の決定変数とし, 0-1 ベクトル $\mathbf{x} := (x_1, x_2, \dots, x_n)$ を解,

$$\sum_{j \in N} w_j x_j \leq C \quad (1.2)$$

を満たす解を実行可能解と呼ぶ. 解 \mathbf{x} に対し, プレイヤー k の利得はナップサック中の $N_0 \cup N_k$ の商品の利得の総和

$$z^0(\mathbf{x}) + z^k(\mathbf{x}) \quad (1.3)$$

で表されるものとする. ここに,

$$z^k(\mathbf{x}) := \sum_{j \in N_k} p_j x_j \quad (1.4)$$

で, 我々は (1.3) 式の最小値を最大化することを目的とする. この問題を 一般化ナップサック共有問題

(generalized knapsack sharing problem: **GKSP**) と呼ぶと、問題は次のように定式化される。

GKSP:

$$\max. \min_{1 \leq k \leq s} \{z^0(\boldsymbol{x}) + z^k(\boldsymbol{x})\} \quad (1.5)$$

$$\text{s. t.} \quad \sum_{j \in N} w_j x_j \leq C \quad (1.6)$$

$$x_j \in \{0, 1\}, \quad (j \in N) \quad (1.7)$$

以下では、 z^* を GKSP の最適目的関数値、 N_X を個別商品すべての集合と定義する。すなわち

$$N_X := \bigcup_{k=1}^s N_k \quad (1.8)$$

GKSP は、個別商品群が存在しない (すなわち、 $s = 0$ の) 場合、通常の 0-1 ナップサック問題¹³⁾ であり、逆に共有商品群が存在しない (すなわち、 $N_0 = \emptyset$ の) 場合は、ナップサック共有問題 (knapsack sharing problem: **KSP**)^{4, 18, 19)} となる。KP が既に \mathcal{NP} -困難⁵⁾ であるので、それを一般化した GKSP もまた \mathcal{NP} -困難である。本問題のように、max-min 型の目的関数^{3, 20)} を持つ最適化問題は様々な場面で研究されているが、ナップサック問題に関連するものとしては Brown^{1, 2)}、Jacobsen⁸⁾、Kaplan⁹⁾、Luss^{11, 12)}、Porteus¹⁵⁾、Tang¹⁷⁾、Zeitlin²¹⁾ などがある。これらはすべて GKSP の特殊ケースと位置づけられる。

本稿では、GKSP を解くための厳密解法を提案し、その有効性を数値実験により確かめる。まず、一般性を失うことなく以下を仮定する。

A₁. 問題のデータ C, p_j, w_j ($j \in N$) はすべて正整数。

A₂. それぞれの商品群 N_i で、商品は相対価値の大きいものから順に番号付けられている。すなわち、 $i = 0, 1, \dots, s$ について $j, j' \in N_i, j < j'$ ならば、 $p_j/w_j \geq p_{j'}/w_{j'}$ 。

A₃. $\sum_{j \in N} w_j > C$ かつ、 $w_j \leq C, j \in N$ 。

2. 問題の分割と上下界値

2.1 問題の分割

GKSP を解くために、以下の 2 つの問題を考える。

KP_k(c):

$$\max. \quad z^k(\boldsymbol{x}) \quad (2.1)$$

$$\text{s. t.} \quad \sum_{j \in N_k} w_j x_j \leq c \quad (2.2)$$

$$x_j \in \{0, 1\}, \quad (j \in N_k) \quad (2.3)$$

c が与えられたとき、 $KP_k(c)$ は商品群 N_k に関する通常の 0-1 ナップサック問題で、数十万までの商品数について、数分程度の計算時間で解くことができる⁷⁾。 $KP_k(c)$ の最適目的関数値を $z_{KP_k}^*(c)$ で表し、特に $z_{KP_0}^*(c)$ を $z_{KP}^*(c)$ と略記する。

また、問題

KSP(c):

$$\max. \quad \min_{1 \leq k \leq s} z^k(\boldsymbol{x}) \quad (2.4)$$

$$\text{s. t.} \quad \sum_{j \in N_X} w_j x_j \leq c \quad (2.5)$$

$$x_j \in \{0, 1\}, \quad (j \in N_X) \quad (2.6)$$

の最適目的関数値を $z_{KSP}^*(c)$ とする。 c が与えられた場合、 $KSP(c)$ はナップサック共有問題で、数万商品までの問題を解くアルゴリズムが開発されている^{4, 18)}。

これらを用いると、問題 GKSP は次の問題に帰着される。

GKSP':

$$\max. \quad z^*(c) := z_{KP}^*(c) + z_{KSP}^*(C - c) \quad (2.7)$$

$$\text{s. t.} \quad 0 \leq c \leq C \quad (2.8)$$

さらに、 $z_{KP_k}^*(c)$ の '逆関数' を

$$c_{KP_k}^*(z) := \min\{c \mid z_{KP_k}^*(c) \geq z\} \quad (2.9)$$

とし、 $z_{KSP}^*(c)$ の逆関数 $c_{KSP}^*(z)$ も同様に定義すると、次が成立する⁶⁾。

定理 2.1

(i) $z_{KP_k}^*(c), z_{KSP}^*(c)$ は、いずれも $[0, \infty)$ において単調非減少、右連続な階段関数である。

(ii) $z_{KSP}^*(c)$ は $z_{KP_k}^*(c)$ ($1 \leq k \leq s$) を横方向に加えあわせて得られる。すなわち、

$$c_{KSP}^*(z) = \sum_{k=1}^s c_{KP_k}^*(z) \quad (2.10)$$

2.2 連続緩和

GKSP の上下界値を得るため、 $KP_k(c)$ 及び $KSP(c)$ において制約条件 $x_j \in \{0, 1\}$ を $0 \leq x_j \leq 1$ に置き換えた連続緩和問題¹⁴⁾ を $\overline{KP}_k(c), \overline{KSP}(c)$ とし、それらの最適目的関数値をそれぞれ $\bar{z}_{KP_k}(c), \bar{z}_{KSP}(c)$ とする。また、これらの逆関数をそれぞれ $\underline{c}_{KP_k}(z), \underline{c}_{KSP}(z)$ と記す。このとき、以下の定理が得られる¹⁰⁾。

定理 2.2

- (i) $\bar{z}_{KP_k}(c), \bar{z}_{KSP}(c)$ はいずれも $[0, \infty)$ において区分的に線形, 単調非減少な凹関数である.
- (ii) $\bar{z}_{KSP}(c)$ は $\bar{z}_{KP_k}(c)$ ($1 \leq k \leq s$) を横方向に加え合わせて得られる. すなわち,

$$\underline{z}_{KSP}(z) = \sum_{k=1}^s \underline{z}_{KP_k}(z) \quad (2.11)$$

図 2.1 は $s = 2, n = 30, |N_k| = 10$ ($k = 0, 1, 2$), $C = 6000$ で, p_j, w_j を $[1, 1000]$ 間の一様乱数とした場合の計算例で, 関数 $\bar{z}_{KP_i}(C - c)$ ($i = 1, 2$) を細い実線で, $\bar{z}_{KP_0}(c)$ と $\bar{z}_{KSP}(C - c)$ を太い破線で示し, それらの和

$$\bar{z}(c) := \bar{z}_{KP}(c) + \bar{z}_{KSP}(C - c) \quad (2.12)$$

を太い実線で表している.

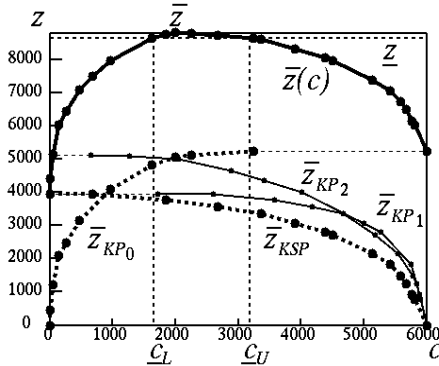


図 2.1 上下界値
Fig. 2.1 Upper and lower bounds.

2.3 上下界値

(2.12) 式で定義した $\bar{z}(c)$ は区分的に線形な凹関数で, $c = \bar{c}$ で最大値 $\bar{z} = \bar{z}(\bar{c})$ をとるとすると, \bar{z} は明らかに GKSP の一つの上界値となる. 図 2.1 では $\bar{c} = 1999, \bar{z} = 8818$ である.

上の \bar{c} において, $KP_0(\bar{c}), KSP(C - \bar{c})$ を厳密に解くと, それらは GKSP の実行可能解となり, そのときの目的関数値

$$\underline{z} := z_{KP_0}^*(\bar{c}) + z_{KSP}^*(C - \bar{c}) \quad (2.13)$$

は明らかに GKSP の一つの下界値を与える. 上の例ではこれは $\underline{z} = 8666$ である.

3. 厳密解法

凹関数 $\bar{z}(c)$ は, 一般に \bar{z} より小さい $z = \underline{z}$ と 2 つの点で交わるが, このときの横座標を $\underline{c}_L, \underline{c}_U$ とする ($\underline{c}_L \leq \underline{c}_U$). これらは 2 分探索法により容易に求められる. すると, GKSP の目的関数値 \underline{z} の解がすでに得られているので, 区間 $[\underline{c}_L, \underline{c}_U]$ 内の c についてのみ $z^*(c) = z_{KP}^*(c) + z_{KSP}^*(C - c)$ の最大値を求めれば十分である. 更に, $z_{KP}^*(c)$ は単調非減少で右連続な階段関数なので, $z_{KP}^*(c)$ の不連続点についてのみ, $z^*(c)$ を計算すればよい. すなわち, 一般的な解法は次のようになる.

解法 0 :
 $[\underline{c}_L, \underline{c}_U]$ 間の $z_{KP}^*(c)$ の各不連続点 c で, $KP(c), KSP(C - c)$ を厳密に解き, $z^*(c)$ を最大とする $c = c^*$ を求める.

図 2.1 の例については, $\underline{c}_L = 1651.9, \underline{c}_U = 3194.3$ で, この間の $z_{KP}^*(c)$ を $\bar{z}(c)$ とともに図 3.1 に示す. 区間 $[\underline{c}_L, \underline{c}_U]$ にふくまれる $z_{KP}^*(c)$ の (4 個の) 不連続点で, $z_{KSP}^*(C - c)$ を解き, $z^*(c)$ を求めると, 図 3.1 の破線のようになり, $c^* = 2261$ で最大値 $z^* = 8719$ が得られる.

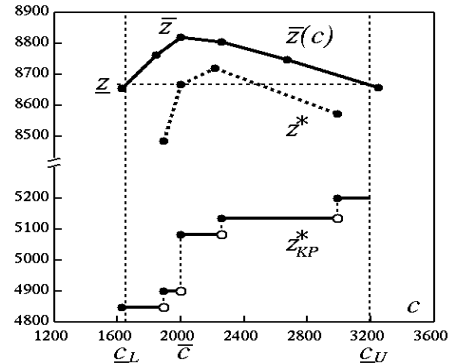


図 3.1 関数 $\bar{z}(c)$ と $z^*(c)$
Fig. 3.1 Functions $\bar{z}(c)$ and $z^*(c)$

ところで, 上の解法を実行するためには, $[\underline{c}_L, \underline{c}_U]$ 間の $z_{KP}^*(c)$ の不連続点を全て列挙する必要があるが, このために, 次の逆問題を考える.

IKP(z):

$$\min. \quad c(x) := \sum_{j \in N_0} w_j x_j \quad (3.1)$$

$$\text{s. t.} \quad \sum_{j \in N_0} p_j x_j \geq z \quad (3.2)$$

$$x_j \in \{0, 1\} \quad (3.3)$$

この問題は, $y_j := 1 - x_j$ と変数変換をすることにより, 通常の 0-1 ナップサック問題と同じアルゴリズムで解くことができる¹⁸⁾.

(3.3) を $0 \leq x_j \leq 1$ と連続緩和した問題を $\text{IKP}(z)$ とすると, 以下が成立する.

定理 3.1

$\text{IKP}(z), \text{IKP}(z)$ の最適目的関数値はそれぞれ $c_{\text{KP}_0}^*(z), \underline{c}_{\text{KP}_0}(z)$ である.

$z_{\text{KP}}^*(c)$ の不連続点を全列挙するアルゴリズムは次のようになる.

不連続点全列挙アルゴリズム:

Step 1. $c := \underline{c}_U$ とする.

Step 2. $c \leq \underline{c}_L$ なら終了. そうでなければ $\text{KP}_0(c)$ を解き, $z := z_{\text{KP}}^*(c)$ を求める.

Step 3. $\text{IKP}(z)$ を解き, 最適目的関数値 $\underline{c} := c_{\text{KP}_0}^*(z)$ を得る. (\underline{c}, z) は不連続点なので, これを出力する.

Step 4. $c := \underline{c} - 1$ として Step 2 へ戻る

図 3.2 は上のアルゴリズムの挙動を示したもので, 図中の # i は step i を意味する.

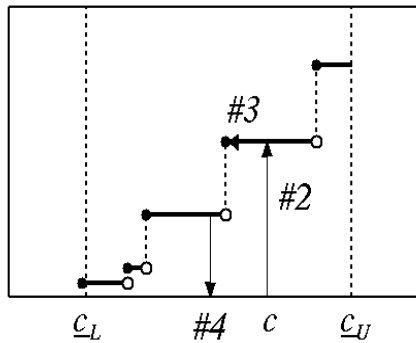


図 3.2 不連続点列挙アルゴリズム

Fig. 3.2 Enumeration of discontinuity points.

実際には, Step 2 で $\text{KP}_0(c)$ を解くときに, 例えば Horowitz-Sahni のアルゴリズム⁷⁾ を用いると, 不連続点 (\underline{c}, z) が得られることが多い. このような場合には, 上のアルゴリズムを若干工夫すると, Step 3 の $\text{IKP}(z)$ を解く部分を省略して, より高速な不連続点の全列挙アルゴリズムを構築することができる.

ところで, 解法 0 では全ての不連続点で $\text{KSP}(C - c)$ を厳密に解いたが, この部分も下のように簡略化できる.

解法 1 :

$[\underline{c}_L, \underline{c}_U]$ 内の $z_{\text{KP}}^*(c)$ の各不連続点 c で, 次を実行する.

- Step 1.** $\text{KP}(c)$ を解き, $z_{\text{KP}}^*(c)$ を得る.
- Step 2.** $z_{\text{KSP}}^*(C - c) \leq \underline{z} - z_{\text{KP}}^*(c)$ かどうかを調べる. Yes なら Step 4 へ, そうでなければ Step 3 へ進む.
- Step 3.** \underline{z} を $\underline{z} \leftarrow z_{\text{KP}}^*(c) + z_{\text{KSP}}^*(C - c)$ により更新する.
- Step 4.** 次の不連続点へ進む.

上の Step 2 の判定条件は

$$z^\dagger := \underline{z} - z_{\text{KP}}^*(c) \quad (3.4)$$

とおくと

$$z_{\text{KSP}}^*(C - c) \leq z^\dagger \quad (3.5)$$

となるが, これは

$$c_{\text{KSP}}^*(z^\dagger) \geq C - c \quad (3.6)$$

と同値である. (3.5) 式を判定するには, $\text{KSP}(C - c)$ を厳密に解き, $z_{\text{KSP}}^*(C - c)$ を求めれば良いが, これを何度も反復するのは時間がかかるので, 以下ではもっと簡便にこれを判定することを考える.

定理 2.1, 2.2 より, 次が成立することに注意する.

定理 3.2

$$c_{\text{KSP}}^*(z) \geq \sum_{k=1}^s [c_{\text{KP}_k}(z)] \geq [c_{\text{KSP}}(z)] \quad (3.7)$$

以上より, (3.5) 式をチェックするには次の方法が考えられる.

- (i) $\text{KSP}(C - c)$ を解き, 直接 (3.5) 式を確かめる.
- (ii) $\text{IKP}_k(z^\dagger)$ を解き, (3.6) 式を確かめる.
- (iii) 次式を確かめる.

$$\sum_{k=1}^s [c_{\text{KP}_k}(z^\dagger)] \geq C - c \quad (3.8)$$

- (iv) 次式を確かめる.

$$[c_{\text{KSP}}(z^\dagger)] \geq C - c \quad (3.9)$$

定理 3.2 より、これらの間には、(3.9) \Rightarrow (3.8) \Rightarrow (3.6) の関係がある。(3.8), (3.9) は 2 分探索法で高速にチェックできるので、計算時間は (iv) \sim (i) の順に長くなる。そこで、チェック・レベル (CL) を次のように定める。

- CL = 0: (i) により (3.5) 式を判定する。
- CL = 1: (ii) により (3.6) 式を判定する。
- CL = 2: (iv) \Rightarrow (ii) の順にチェックする。
- CL = 3: (iv) \Rightarrow (iii) \Rightarrow (ii) の順にチェックする。

4. 数値実験

前節のアルゴリズムを ANSI C 言語で実装し、IBM RS/6000 SP44 Model 270 (CPU: POWER3-II SMP 2 way, 375MHz) 上で、 n, s, C を様々に変えて、一連の数値実験を行った。以下では、全商品中で共通商品群の占める比率を $\lambda := |N_0|/n$ で表し、個別商品群はすべて同数で、 $|N_i| = (1 - \lambda)n/s$ とした。また、本節では w_j と p_j は以下の相関タイプによりランダムに発生させた。

- 無相関 (UNCOR)

w_j : [1, 1000] の一様乱数

p_j : [1, 1000] の一様乱数 (w_j と独立)

4.1 チェック・レベル

図 4.1 は、 $n = 2048, C = 200n$ で、 $s = 2, 4, 8, \lambda = 1/2, 1/4, 1/8$ としたときの計算時間 (CPU sec.) を示す。各点はランダムな 10 回の試行の平均値である。

どのケースについても、チェック・レベルが 0~3 の順に計算時間が短くなった。よって、以下ではチェック・レベルは 3 に固定する。

4.2 既存ソフトとの比較

GKSP は線形 0-1 計画問題であるので、市販の数値計画ソフトを用いて解くことが可能である。ここでは、国内で広範に用いられている NUOPT¹⁶⁾ と、本稿のアルゴリズムを比較する。

表 4.1 は $n = 512, 1024, 2048, s = 2, 4, 8, \lambda = 1/2, 1/4, 1/8$ の場合について、計算時間と生成子問題数 (NUOPT), KP を解いた回数 (#KP: 本稿の方法) を示している。各行は、ランダムに発生した 10 個の例題についての平均で、NUOPT と本稿の方法は計算が終了した場合については、すべて同じ結果を出力した。NUOPT の項で、— としているのはメモリー

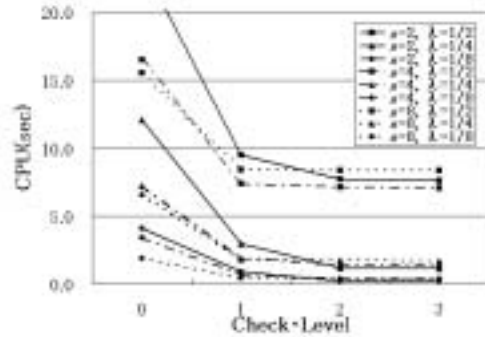


図 4.1 チェック・レベルと計算時間

Fig. 4.1 Check level and CPU time in seconds.

不足で計算できなかったものや、50000 秒以内の CPU 時間で計算が終了しなかったものを示している。

表で見ると、本稿の方法は NUOPT に比べ圧倒的に優勢である。

4.3 利得と重量が無相関の場合

表 4.2 (a) ~ (c) に、 $C = 200n$ とした場合の結果をまとめる。各行は 10 回のランダムな試行の平均値で、# 不連続、#KSP, #KP はそれぞれ、 $[\underline{z}_L, \underline{z}_U]$ 内の $z_{KP}^*(c)$ の不連続点の数と、KSP, KP を解いた回数を示す。

図 4.2 は、表 4.2 の計算時間を問題サイズ n の関数として両対数スケールで示したもので、ほぼ直線状に増加していることが分かる。

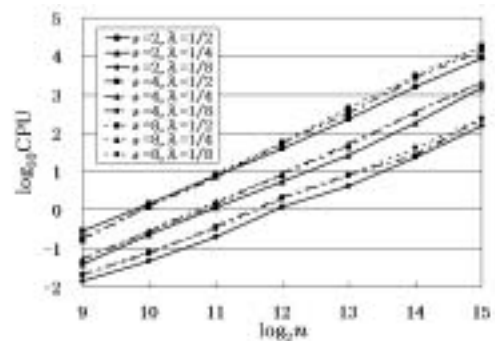


図 4.2 計算時間 (無相関の場合)

Fig. 4.2 CPU time in seconds (UNCOR).

同様に、図 4.3~4.5 はそれぞれ、最適目的関数値 (z^*), KP を解いた回数 (#KP), および $z_{KP}^*(c)$ の不連続点の数 (# 不連続点) を示している。これらから、

1. z^* は n にほぼ比例して増加する。
2. 計算時間は大略 $n^{2.5}$ に比例する。すなわち、 n が 2 倍になると計算時間は約 6 倍となる。

表 4.1 NUOPT との比較

Table 4.1 Comparison against NUOPT.

n	s	λ	NUOPT		本稿の方法	
			子問題	CPU(sec)	‡KP	CPU(sec)
512	2	1/2	5459.9	6.73	602.1	0.279
		1/4	4610.5	5.99	137.6	0.039
		1/8	6368.7	7.61	58.8	0.014
	4	1/2	14831.5	15.72	926.8	0.180
		1/4	60279.6	61.63	354.3	0.054
		1/8	84327.7	88.25	143.4	0.021
	8	1/2	61466.2	61.02	1693.8	0.158
		1/4	17656865.6	20629.03	746.7	0.051
		1/8	27391101.4	34499.09	291.0	0.020
1024	2	1/2	7201.9	18.12	1098.6	1.428
		1/4	14272.7	29.77	287.0	0.229
		1/8	22130.1	42.20	83.6	0.047
	4	1/2	58564.3	97.12	1926.8	1.202
		1/4	648994.6	1067.66	575.8	0.266
		1/8	1198330.9	1992.45	185.9	0.073
	8	1/2	17719810.8	28242.71	4100.8	1.306
		1/4	---	---	1428.1	0.279
		1/8	---	---	449.6	0.083
2048	2	1/2	26287.4	106.16	1730.7	7.607
		1/4	49228.4	161.28	496.7	1.151
		1/8	32320.9	122.10	146.6	0.202
	4	1/2	811542.2	2231.93	3259.6	7.078
		1/4	2171312.0	5976.73	963.4	1.395
		1/8	9231342.4	26728.33	318.1	0.363
	8	1/2	---	---	7361.1	8.334
		1/4	---	---	2670.2	1.689
		1/8	---	---	628.1	0.324

表 4.2 (a) 数値実験結果 (無相関, $s = 2$)

Table 4.2 (a) The result of experiments (UNCOR, $s = 2$).

n	λ	最適値	‡不連続	‡KSP	‡KP	CPU(sec)
512	1/2	143207.6	263.7	3.5	602.1	0.279
	1/4	118747.4	76.2	1.6	137.6	0.039
	1/8	105372.9	36.5	1.1	58.8	0.014
1024	1/2	287017.7	504.4	3.6	1098.6	1.428
	1/4	238262.9	179.0	1.6	287.0	0.229
	1/8	212265.9	55.4	1.2	83.6	0.047
2048	1/2	580009.8	895.8	3.3	1730.7	7.607
	1/4	478850.5	348.4	3.0	496.7	1.151
	1/8	426933.0	105.8	1.4	146.6	0.202
4096	1/2	1160495.6	1223.7	3.7	2441.8	38.745
	1/4	959227.7	559.6	1.9	719.0	5.326
	1/8	852130.8	259.0	1.8	330.6	1.222
8192	1/2	2320109.1	1896.9	4.0	3535.8	228.598
	1/4	1917876.8	771.8	1.6	1038.7	26.788
	1/8	1704748.9	389.6	1.5	462.5	4.111
16384	1/2	4635270.9	2556.6	2.8	4307.3	1559.285
	1/4	3835638.7	1347.8	1.6	1808.5	183.417
	1/8	3409682.5	651.4	1.3	809.9	23.320
32768	1/2	9278281.5	3526.9	3.1	5632.5	9027.214
	1/4	7671686.7	2042.0	2.0	2994.9	1532.052
	1/8	6826241.1	1025.7	1.6	1380.6	155.206

表 4.2 (b) 数値実験結果 (無相関, $s = 4$)

Table 4.2 (b) The result of experiments (UNCOR, $s = 4$).

n	λ	最適値	# 不連続	#KSP	#KP	CPU(sec)
512	1/2	129161.2	194.5	4.2	926.8	0.180
	1/4	89622.4	79.6	2.9	354.3	0.054
	1/8	67943.7	32.5	1.7	143.4	0.021
1024	1/2	257614.7	419.1	4.7	1926.8	1.202
	1/4	179284.2	161.2	2.1	575.8	0.266
	1/8	136604.4	58.9	1.1	185.9	0.073
2048	1/2	520353.5	726.7	5.1	3259.6	7.078
	1/4	359129.1	268.8	2.8	963.4	1.395
	1/8	274233.0	109.6	1.6	318.1	0.363
4096	1/2	1042916.8	1382.1	5.0	6287.3	53.839
	1/4	720578.1	560.3	2.6	1782.2	8.908
	1/8	546848.4	244.7	1.6	646.0	2.144
8192	1/2	2085952.3	1949.2	5.3	8538.4	329.111
	1/4	1442061.1	914.1	2.9	2710.5	51.042
	1/8	1095595.0	386.2	2.0	822.2	8.035
16384	1/2	4164292.5	2911.3	3.6	12590.0	2682.876
	1/4	2885712.1	1356.4	1.9	3748.1	338.651
	1/8	2192246.5	549.4	1.6	830.3	26.047
32768	1/2	8335031.6	3442.4	2.6	12449.2	13600.522
	1/4	5765504.1	1703.5	1.8	3911.1	2095.001
	1/8	4389332.8	926.7	1.1	1538.7	235.187

表 4.2 (c) 数値実験結果 (無相関, $s = 8$)

Table 4.2 (c) The result of experiments (UNCOR, $s = 8$).

n	λ	最適値	# 不連続	#KSP	#KP	CPU(sec)
512	1/2	124412.4	175.2	4.5	1693.8	0.158
	1/4	76073.3	74.9	2.6	746.7	0.051
	1/8	49696.0	26.4	2.1	291.0	0.020
1024	1/2	247913.5	452.2	5.3	4100.8	1.306
	1/4	152083.2	155.9	3.2	1428.1	0.279
	1/8	99697.9	49.5	1.8	449.6	0.083
2048	1/2	500175.7	821.2	6.7	7361.1	8.334
	1/4	303629.9	316.2	3.4	2670.2	1.689
	1/8	199779.0	82.4	2.0	628.1	0.324
4096	1/2	1003363.9	1407.4	6.2	12375.6	56.343
	1/4	610181.6	467.2	3.2	3740.4	8.146
	1/8	398089.8	181.1	1.7	1308.7	1.992
8192	1/2	2007457.8	2270.9	6.6	20148.0	444.922
	1/4	1221637.3	685.4	2.3	5294.0	45.467
	1/8	798697.7	259.4	1.6	1579.8	7.792
16384	1/2	4006383.7	2933.1	5.6	25640.2	3130.824
	1/4	2445181.6	1018.0	1.7	7728.6	337.385
	1/8	1598601.3	434.7	1.5	2399.4	40.899
32768	1/2	8020818.2	3640.1	4.7	31579.0	18134.677
	1/4	4881680.3	1223.9	1.8	8325.3	2045.644
	1/8	3200604.0	611.9	1.5	2553.9	225.649

3. n, s を固定して, λ を $1/2, 1/4, 1/8$ と変えると, z^* , 計算時間, #KP, #不連続点とも大幅に減少する.

4. n, λ を固定して, s を $2, 4, 8$ と変えると, z^* は減少し, #KP と計算時間は増加するが, #不連続点 はさほど変化しない.

ことが分かる。

ここで、 λ を小さくすると、部分問題として解かれる 0-1 ナップサック問題のサイズが全体的に小さくなることによると考えられる。また、 λ を固定して s を増やした場合、共通商品数が不変であることから 4 のようになると思われる。

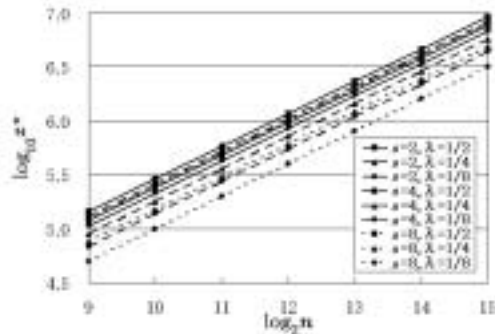


図 4.3 最適目的関数値 (z^*)
Fig. 4.3 Optimal objective value z^* .

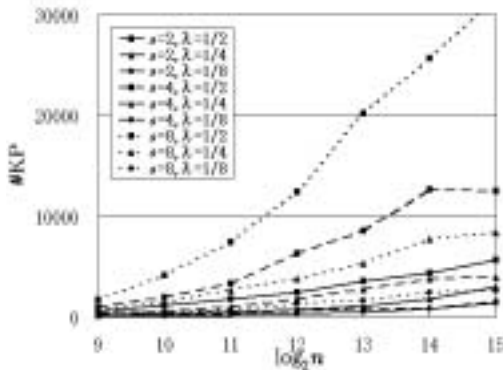


図 4.4 ナップサック問題を解く回数 (#KP)
Fig. 4.4 The number of KPs solved.

4.4 λ の影響

共有商品群の比率 λ を変えたときの計算時間を図 4.6 に示す。ここで、 $n = 2048, 4096$, $s = 2, 4, 8$, $C = 200n$ で、 $\lambda = 1$ は単純なナップサック問題 (KP) を、 $\lambda = 0$ はナップサック共有問題 (KSP) を意味する。各点は 10 回のランダムな計測の平均値である。

図 4.6 から、計算時間は λ の関数と考えると、 $\lambda = 0, 1$ の場合に小さく、 λ が 0.7 から 0.9 あたりにピークを持つ、単峰性の形をなしている。 $\lambda = 1$ に近い側にピークがあり、 $\lambda \leq 0.2$ では計算時間が小さいのが特徴である。

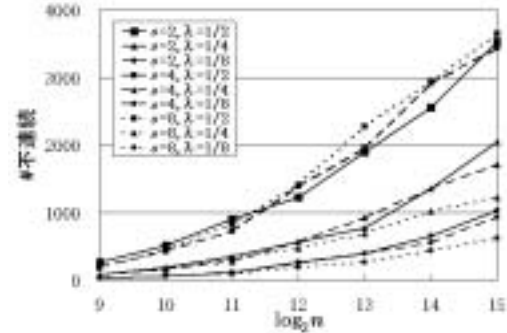


図 4.5 不連続点数 (#不連続)
Fig. 4.5 The number of discontinuity points.

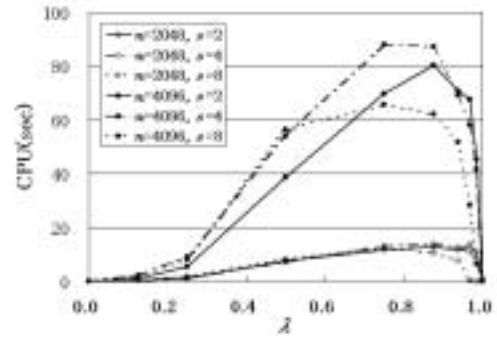


図 4.6 λ と計算時間の関係
Fig. 4.6 λ vs. CPU time in seconds.

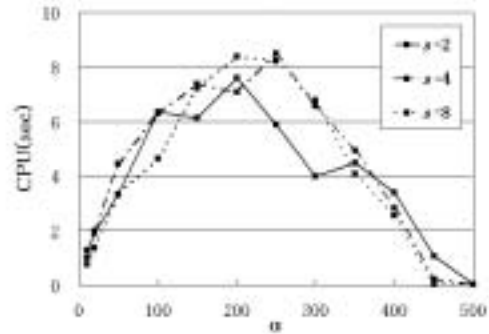


図 4.7 容量 C と計算時間の関係
Fig. 4.7 Capacity vs. CPU time in seconds.

4.5 容量の影響

図 4.7 は $n = 2048$, $\lambda = 1/2$, $s = 2, 4, 8$ で, 容量 C を $C = \alpha n$ としたときの計算時間を $0 \leq \alpha \leq 500$ の関数として表示したものである. 商品の重量 w_j が $[1, 1000]$ の一様乱数なので, $\alpha = 500$ はほとんど全ての商品が収容可能であることを示す. α が 200 ~ 300 付近にピークがある関数となっている.

5. 数値実験: 利得と重量の間に相関のある場合

商品の利得と重量の間の相関の影響を調べるために, 相関タイプとして前節の UNCOR 以外に, 次の 2 つのケースを考える¹³⁾.

- 弱相関 (WEAK)

w_j : $[1, 1000]$ の一様乱数

p_j : $[w_j, w_j + 200]$ の一様乱数

- 強相関 (STRONG)

w_j : $[1, 1000]$ の一様乱数

p_j : $p_j = w_j + 100$

5.1 弱相関の場合

表 5-1 (a) ~ (c) は, 弱相関 (WEAK) の場合の結果を示す. 表の見方は表 4.2 と同様で, 各行は $C = 200n$ のときのランダムな 10 回の試行の平均値である.

表 4.2 と表 5.1 を比較すると, p_j と w_j の間の相関の有無により, 以下の差異が生じることが認められる.

1. n, λ, s が同じ場合, † 不連続, ‡ KP は全般に WEAK の場合の方が UNCOR に比べかなり小さい.
2. その結果, 計算時間も WEAK の場合の方が全般にかなり少なくて済んでいる.
3. 最適目的関数値は, 同じ n, λ で比較すると, $s = 2$ では UNCOR の場合より WEAK の場合の方がやや小さく, $s = 8$ ではその逆となる.

このうち, 1. と 2. は以下のように説明される. すなわち, UNCOR の場合, 関数 $\bar{z}(c)$ は図 2.1 に示したように, 緩やかに上に湾曲した凹関数となり, 区間 $[\underline{c}_L, \underline{c}_U]$ の幅が大となる. これに対し, WEAK の場合は図 5.1 に示したように $\bar{z}_{KP}(c)$, $\bar{z}_{KSP}(c)$ が直線に近くなり, その結果 $\bar{z}(c)$ が鋭角的なピークを持って, $[\underline{c}_L, \underline{c}_U]$ の幅が極めて小さくなる. これより, この区間に含まれる不連続点数も, KP を解く回数も小さくなり, そのために計算時間も少なくてすむと考えられる.

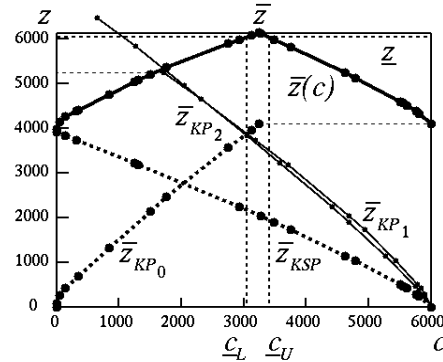


図 5.1 関数 $\bar{z}(c)$ (弱相関の場合)

Fig. 5.1 Function $\bar{z}(c)$ for the case of WEAK.

5.2 強相関の場合

この場合の結果を表 5.2 に示す. $n \geq 2048$ ではほとんど歯がたたず, $n \leq 1024$ でも $\lambda = 1/2$ の場合は CPU 時間 50000 秒以内に解けたケースは皆無であった.

この原因は, アルゴリズム中で KP を何回か解くわけであるが, 強相関の場合 KP を解くこと自体が極めて難しい¹³⁾ ということに尽きる. 従って, この種類の問題については, KP を反復して解くという本稿のアプローチでは, これ以上のサイズの問題を厳密に解くことは望み薄と言わざるを得ない.

ただし, 実際に解けた問題については何れも最初の近似解 (すなわち, 2.3 節の \underline{z} を与える実行可能解) がそのまま最適解となっており, 計算時間の大半は最適性の確認に費やされている. 従って, このような場合でも本稿の方法は短時間で精度の高い解を見出す近似解法として有望と思われる.

6. まとめ

前に報告したナップサック共有問題を拡張して一般化ナップサック共有問題を定式化し, それを厳密に解くアルゴリズムを構築した. その結果, 商品の利得と重量間に相関がない場合, もしくは相関が弱い場合については数万個までの商品を含む問題を解くことができた.

本稿の方法は, KP を反復して解くことにより GKSP を解こうとするものなので, 相関の強い場合には別のアプローチなど, 何らかの工夫が必要となる. これについては今後の課題としたい.

表 5.1 (a) 数值実験結果 (弱相関, $s = 2$)Table 5.1 (a) The result of experiments (WEAK, $s = 2$).

n	λ	最適値	不連続	‡KSP	‡KP	CPU(sec)
512	1/2	127519.4	238.4	3.3	565.2	0.550
	1/4	105164.2	4.4	1.0	18.4	0.012
	1/8	87180.9	3.3	1.0	16.5	0.018
1024	1/2	254911.4	422.3	2.7	1032.9	2.771
	1/4	211701.8	4.1	1.0	16.9	0.032
	1/8	175337.4	3.4	1.0	15.0	0.030
2048	1/2	511051.1	601.5	3.4	1425.7	9.665
	1/4	424127.6	3.7	1.0	14.9	0.065
	1/8	352725.4	3.4	1.0	15.6	0.074
4096	1/2	1022660.1	769.2	2.8	1343.0	32.135
	1/4	846463.8	3.6	1.1	13.9	0.172
	1/8	704942.3	3.1	1.0	13.4	0.133
8192	1/2	2045190.8	1041.2	2.9	1502.0	171.262
	1/4	1690568.5	3.5	1.1	14.3	0.600
	1/8	1408351.9	3.4	1.3	15.1	0.465
16384	1/2	4090299.4	1364.2	3.6	1731.9	969.199
	1/4	3382300.0	3.8	1.3	13.0	3.006
	1/8	2813502.7	3.4	1.6	12.6	1.647
32768	1/2	8181635.5	1762.6	6.3	1957.0	4853.373
	1/4	6762091.4	4.3	1.6	12.9	15.181
	1/8	5629033.8	3.8	1.5	10.3	5.738

表 5.1 (b) 数值実験結果 (弱相関, $s = 4$)Table 5.1 (b) The result of experiments (WEAK, $s = 4$).

n	λ	最適値	不連続	‡KSP	‡KP	CPU(sec)
512	1/2	126892.1	101.6	3.4	484.6	0.225
	1/4	90496.8	3.9	1.0	32.0	0.010
	1/8	62224.7	3.3	1.0	29.3	0.012
1024	1/2	253609.3	199.3	3.5	950.0	1.272
	1/4	183184.0	3.7	1.0	29.2	0.022
	1/8	125590.6	3.5	1.0	29.6	0.030
2048	1/2	508314.7	306.2	3.2	1421.2	4.629
	1/4	366548.6	3.5	1.0	27.5	0.063
	1/8	253698.6	3.3	1.0	26.7	0.070
4096	1/2	1017781.5	406.2	4.0	1793.8	17.440
	1/4	730029.7	3.6	1.0	25.1	0.164
	1/8	506959.7	3.3	1.0	24.7	0.140
8192	1/2	2035157.5	567.9	4.2	2328.1	98.996
	1/4	1456232.2	3.5	1.0	23.1	0.464
	1/8	1010978.5	3.2	1.0	22.0	0.284
16384	1/2	4070231.1	722.2	3.8	2542.9	554.726
	1/4	2914961.1	3.3	1.1	20.3	1.833
	1/8	2017694.9	3.2	1.2	21.0	0.848
32768	1/2	8141805.1	914.3	3.2	2115.3	2706.472
	1/4	5826061.8	3.7	1.3	19.7	9.442
	1/8	4038326.1	3.3	1.1	16.8	3.324

表 5.1 (c) 数値実験結果 (弱相関, $s = 8$)

Table 5.1 (c) The result of experiments (WEAK, $s = 8$).

n	λ	最適値	‡ 不連続	‡KSP	‡KP	CPU(sec)
512	1/2	126847.0	8.4	1.0	9.6	0.016
	1/4	83150.0	4.6	1.0	69.1	0.011
	1/8	49745.1	3.6	1.0	59.0	0.012
1024	1/2	253418.2	68.3	1.9	452.3	0.410
	1/4	168917.5	4.2	1.0	57.4	0.024
	1/8	100704.8	3.6	1.0	56.8	0.028
2048	1/2	507845.8	210.8	4.1	1812.6	3.110
	1/4	337759.4	3.8	1.0	55.6	0.059
	1/8	204181.8	3.7	1.0	54.5	0.067
4096	1/2	1016903.4	287.0	4.3	2451.6	11.421
	1/4	671807.3	3.7	1.0	46.9	0.146
	1/8	407966.9	3.3	1.0	44.9	0.125
8192	1/2	2033541.4	380.1	4.6	3084.4	65.622
	1/4	1339059.3	3.7	1.0	44.7	0.458
	1/8	812289.1	3.4	1.0	42.4	0.271
16384	1/2	4067029.5	533.5	4.8	4191.4	397.001
	1/4	2681285.0	3.7	1.0	40.2	1.909
	1/8	1619794.0	3.2	1.0	38.5	0.688
32768	1/2	8135309.6	615.8	4.9	4130.4	1784.810
	1/4	5358043.8	3.3	1.0	34.3	6.643
	1/8	3242973.2	3.2	1.1	32.7	2.268

表 5.2 数値実験結果 (強相関)

Table 5.2 The result of experiments (STRONG).

n	s	λ	最適値	‡ 不連続	‡KSP	‡KP	CPU(sec)
256	2	1/4	52026.7	9.2	1.0	20.9	0.454
		1/8	42683.6	5.7	1.0	18.8	13.124
	4	1/4	44870.9	7.7	1.0	36.3	0.037
		1/8	30477.1	4.8	1.0	31.6	0.443
8	1/4	41298.8	6.8	1.0	61.0	0.012	
	1/8	24381.8	4.6	1.0	58.3	0.011	
512	2	1/4	104073.9	15.2	1.0	28.3	169.739
		1/8	85804.4	9.6	1.0	21.3	3905.249
	4	1/4	89928.3	10.1	1.0	33.8	0.777
		1/8	61772.7	6.4	1.0	30.2	51.819
	8	1/4	82856.6	8.0	1.0	59.1	0.104
		1/8	49753.6	5.9	1.0	58.0	0.393
1024	8	1/4	166363.1	13.5	1.0	64.7	3.275
	1/8	99783.3	8.3	1.0	55.6	42.066	

参考文献

- 1) Brown, J. R., "The knapsack sharing problem", *Operations Research*, **27**, 341-355 (1979).
- 2) Brown, J. R., "Bounded knapsack sharing", *Mathematical Programming*, **67**, 343-382 (1994).
- 3) Du, D. -Z. and Pardalos, P. M.(eds.), *Minimax and Applications*, Kluwer, 1995.
- 4) 二川真由美, ナップサック共有問題に関する研究, 修士論文, 防衛大学校, 1996.

- 5) Garey, M. R. and Johnson, D. S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman and Company, 1979.
- 6) 林芳男, 0-1 ナップザック問題の数理とアルゴリズム, 近畿大学商経学会, 2000.
- 7) Horowitz, E. and Sahni, S., "Computing partitions with applications to the knapsack problem", *Journal of ACM*, **21**, 277-292 (1974).
- 8) Jacobsen, S., "On marginal allocation in single constraint min-max problems", *Management Science*, 780-783 (1971).
- 9) Kaplan, S., "Application of programs with maximin objective functions to problems of optimal resource allocation", *Operations Research*, **22**, 802-807 (1974).
- 10) Kuno, T., Konno, H. and Zemel, E., "A linear-time algorithm for solving continuous maximin knapsack problems", *Operations Research Letters*, **10**, 23-26 (1991).
- 11) Luss, H., "A nonlinear minimax allocation problem with multiple knapsack constraints", *Operations Research Letters*, **10**, 183-187 (1991).
- 12) Luss, H., "Minimax resource allocation problems: optimization and parametric analysis", *European Journal of Operational Research*, **60**, 76-86 (1992).
- 13) Martello, S. and Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley & Sons, 1990.
- 14) Nemhauser, G. L. and Wolsey, L. A., *Integer and Combinatorial Optimization*, John Wiley & Sons, 1988.
- 15) Porteus, E. L. and Yormark, J. S., "More on min-max allocation", *Management Science*, **18**, 502-507 (1972).
- 16) (株) 数理システム, NUOPT マニュアル, 2002. URL: <http://www.msi.co.jp/nuopt>
- 17) Tang, C. S., "A max-min allocation problem: its solutions and applications", *Operations Research*, **36**, 359-367 (1988).
- 18) Yamada, T. and Futakawa, M., "Heuristic and reduction algorithms for the knapsack sharing problem", *Computers & Operations Research*, **24**, 961-967 (1996).
- 19) Yamada, T., Futakawa, M. and Kataoka, S., "Some exact algorithms for the knapsack sharing problem", *European Journal of Operational Research*, **106**, 177-183 (1998).
- 20) Yamada, T., Takahashi, H. and Kataoka, S., "A branch-and-bound algorithm for the mini-max spanning forest problem", *European Journal of Operational Research*, **101**, 93-103 (1997).
- 21) Zeitlin, Z., "Integer allocation problems of min-max type with quasiconvex separable functions", *Operations Research*, **29**, 207-211 (1981).

一般化ナップサック共有問題の厳密解法

An Exact Algorithm for the Generalized Knapsack Sharing Problem

(Dedicated to Prof. K. IIDA)

By Masako FUJIMOTO*, Takeo YAMADA**, Seiji KATAOKA*** and
Kohtaro WATANABE****

(Received March 14, 2003; accepted for publication April 23, 2003)

Abstract

We are concerned with a variation of the knapsack problem as well as of the knapsack sharing problem, where we are given a set of n items and a knapsack of a fixed capacity. As usual, each item is associated with its profit and weight, and the problem is to determine the subset of items to be packed into the knapsack. However, in the problem there are s players and the items are classified into $s + 1$ disjoint groups, N_i ($i = 0, 1, \dots, s$). The player k is concerned only with the items in $N_0 \cup N_k$, where N_0 is the set of *common* items. We formulate the generalized knapsack sharing problem as the one to maximize the minimum of the profits over all the players. An algorithm is developed to solve this problem to optimality, and through a series of computational experiments, we evaluate the performance of the developed algorithm.

Key words : knapsack problem, knapsack sharing, combinational optimization, exact algorithm.

* Graduate student, Department of Computer Science, The National Defense Academy

** Professor, Department of Computer Science, The National Defense Academy

*** Professor, Department of Computer Science, The National Defense Academy

**** Professor, Department of Computer Science, The National Defense Academy