数理解析研究所講究録 1409 巻 2005 年 101-114

付加制約のあるナップサック問題への釘付けアプローチ

A pegging approach to the knapsack problem with side constraints

防衛大学校情報工学科
柳 秉俊, セニスカ アミント, 山田 武夫

YOU Byungjun, SENISUKA Aminto, YAMADA Takeo
{g42044, g42043, yamada}@nda.ac.jp
Department of Computer Science, The National Defense Academy
Yokosuka, Kanagawa 239-8686, Japan

1 はじめに

ナップサック問題 (knapsack problem: KP) は OR や情報科学分野の基本問題で, 多数の研究がなされている [6, 4] が, 本稿ではこれにいくつかの制約式が付加されている場合を考える. このような問題の例には, **多次元ナップサック問題** (multi-dimensional knapsack problem: MKP) や, **順序制約付きナップサック問題** (precedence-constrained knapsack problem: PCKP), **排他制約付きナップサック問題** (disjunctively constrained knapsack problem: DCKP) などがある. 通常のナップサック問題がすでに $\mathcal{NP}$-**困難** なので, その拡張であるこれらの問題もまた $\mathcal{NP}$-困難である.

PCKP[7] は **0-1 整数計画問題** [5] として以下のように定式化される.

PCKP:

$$\text{maxmize} \quad \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leq C \tag{2}$$

$$x_i \geq x_j, \quad \forall (i,j) \in E \tag{3}$$

$$x_j \in \{0,1\}, \quad \forall j \in V \tag{4}$$

ここで, $V = \{1, 2, \ldots, n\}$ は商品の集合で, $E \subseteq V \times V$ は商品間の**順序関係**を表している. **順序制約**が意味を持つために, グラフ $G = (V, E)$ は有向グラフで, サイクルを含まないものとする. 通常のナップサック問題の場合と同様に, p_j, w_j は商品の**利得**と**重量**で, C はナップサックの**容量**である.

また, DCKP[9] は次のように定式化される.

DCKP:

$$\text{maximize} \quad \sum_{j=1}^{n} p_j x_j \tag{5}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \leq C \tag{6}$$

$$x_i + x_j \leq 1, \quad \forall (i,j) \in E \tag{7}$$

$$x_j \in \{0,1\}, \quad j = 1, \ldots, n. \tag{8}$$

この場合, E は排他的な商品対の集合を表しており, $G=(V,E)$ は無向グラフである.

本稿ではこれらの問題について, ラグランジュ緩和と釘付けテストにより問題サイズを縮小し, 厳密解を得る方法を示すが, ほとんどの議論は PCKP, DCKP のいずれにも同様に展開できるので, 数値実験の項以外は PCKP について記述する.

2 上下界値

本節では最初にラグランジュ緩和 [5, 6] により PCKP の上界値を求め, 次に局所探索法をベースとした近似解法により下界値を導く.

2.1 ラグランジュ緩和

PCKP の (連続型) ラグランジュ緩和問題は次のように定式化される.

LPCKP($\boldsymbol{\lambda}$):

$$\text{maximize} \quad L:=\sum_{j=1}^{n} p_j x_j + \sum_{(i,j)\in E} \lambda_{ij}(x_i - x_j) \tag{9}$$

$$\text{subject to} \quad \sum_{j=1}^{n} w_j x_j \le C \tag{10}$$

$$0 \le x_j \le 1, \quad j=1,...,n. \tag{11}$$

ここで, $\lambda_{ij} \ge 0$ は制約式 (3) に対応するラグランジュ乗数で, L は PCKP のラグランジアンである. ここで $S_j^+(S_j^-)$ を節点 v_j を始点 (終点) とする枝の終点 (始点) の集合とすると, 目的関数 L を次のように書き直すことが出来る.

$$L=\sum_{j=1}^{n}(p_j + \sum_{k\in S_j^+} \lambda_{jk} - \sum_{i\in S_j^-} \lambda_{ij})x_j \tag{12}$$

$\boldsymbol{\lambda}=(\lambda_{ij})$ を固定すると, LPCKP($\boldsymbol{\lambda}$) は連続型 0-1 ナップサック問題で, その解は容易に求められる. 最適解を $\bar{\boldsymbol{x}}(\boldsymbol{\lambda})=(\bar{x}_j)$, 対応する最適目的関数値を $\bar{z}(\boldsymbol{\lambda})$ とし, PCKP の最適目的関数値を z^* とすると, これらの間には下の関係が成立する.

$$z^* \le \bar{z}(\boldsymbol{\lambda}) \tag{13}$$

すなわち, $\bar{z}(\boldsymbol{\lambda})$ は PCKP の一つの上界値を与える.

$\bar{z}(\boldsymbol{\lambda})$ を $\boldsymbol{\lambda}$ の関数と考えたとき, これをラグランジュ関数という. これについて, 以下に基本的事項をまとめておく [6].

命題 2.1 $\bar{z}(\boldsymbol{\lambda})$ は, $\boldsymbol{\lambda} \ge 0$ において区分的に線形な凸関数である.

命題 2.2 $\boldsymbol{\lambda}$ において $\bar{z}(\boldsymbol{\lambda})$ が微分可能なら

$$\frac{\partial \bar{z}(\boldsymbol{\lambda})}{\partial \lambda_{ij}} = \bar{x}_i - \bar{x}_j. \tag{14}$$

命題 2.3 任意の $\boldsymbol{\lambda} \ge \mathbf{0}$ において, $\bar{\boldsymbol{x}}(\boldsymbol{\lambda})$ が実行可能で (すなわち, 式 (2) ~ (4) を満たし),

$$\lambda_{ij}(\bar{x}_i - \bar{x}_j) = 0 \quad \forall (i,j) \in E \tag{15}$$

ならば, $\bar{x}(\lambda)$ は PCKP の最適値である.

2.2 劣勾配法

任意の $\lambda \geq 0$ に対し, 前に述べたように $\bar{z}(\lambda)$ は PCKP の上界値を与えるが, この値は出来るだけ小さい方が望ましい. このために, **劣勾配法** (subgradient method)[5] を導入する. 以下で, (14) を成分とするベクトル $\partial\bar{z}(\lambda)/\partial\lambda$ を**劣勾配**という.

劣勾配法

Step 1. $\lambda = 0$とする.
Step 2. LPCKP(λ) を解く.
Step 3. 劣勾配を計算し, 探索方向を $d := -\partial\bar{z}(\lambda)/\partial\lambda$ により定める.
Step 4. (1 次元探索)
$\bar{z}(\lambda + \alpha d)$ が最小となる$\alpha \geq 0$ を求める.
Step 5. (15) が成立するか, $\alpha \cong 0$ の場合, 終了.
Step 6. $\lambda \longleftarrow \lambda + \alpha d$ として **Step 2** へ戻る.

上のアルゴリズムが終了した時点での α を $\alpha^\dagger$ とし, 対応する λ を $\lambda^\dagger$ とする. そのときの $\bar{z}(\lambda^\dagger)$ を以下では $\bar{z}$ と記し, PCKP の**上界値**とする.

2.3 ラグランジュ近似解

前節で, ラグランジュ関数 $\bar{z}(\lambda)$ を最小にする λ の値 $\lambda^\dagger$ を求めた. そこでの LPCKP($\lambda^\dagger$) の**最適解** $\bar{x}(\lambda^\dagger)$ は明らかに制約条件 (2) を満たしている. 以下では$\bar{x}(\lambda^\dagger)$=$(\bar{x}_j^\dagger)$ とも記す. これが制約条件 (3), (4) も満たしているならば, これは PCKP の一つの実行可能解なので, その時の目的関数値は PCKP の下界値となる. しかし, 一般には$\bar{x}(\lambda^\dagger)$ は (3), (4) をすべて満たすとは限らないので, $x_j^\dagger$ が 0-1 条件を満たさない場合や, $\bar{x}_i^\dagger < \bar{x}_j^\dagger$ となっている順序制約 $(i,j) \in E$ が見つかった時は, $\bar{x}_j^\dagger \leftarrow 0$ として解を修正する. これによって PCKP の実行可能解が必ず得られるので, これを**ラグランジュ近似解**と呼び, 対応する下界値を $\underline{z}_L$ と記す.

2.4 2-opt 解

次に, ラグランジュ近似解をさらに改良するため, **2-opt** 解法を考える. PCKP の任意の実行可能解 $x = (x_j)$ について,

(i) ナップサックに入っていない商品の一つをナップサックに入れる,

(ii) ナップサックに入っている商品とそうでない商品の一組を入れ替える

のいずれかの操作によって得られる解が実行可能であるとき, その解は x に**隣接する**といい, そのような解全体の集合を $N(x)$ と記して x の **2-opt 近傍**と呼ぶ.

2-opt 解法はラグランジュ近似解を最初の解 x とし, $N(x)$ 内に x より良い解 y が見つかれば, これを新しい x とするという操作を反復するもので, 具体的には下のように書かれる.

アルゴリズム 2-opt

Step 1. $x := \underline{x}_L$ とおく.
Step 2. もし $\exists y \in N(x)$ で $z(y) > z(x)$ であれば **Step3** へ進む, そうでなければ $(x, z(x))$ を出力して終了.
Step 3. $x := y$ として, **Step2** へ戻る.

上のアルゴリズムの出力を $\underline{x}$ と記し, **2-opt 解**と呼ぶ. また, 対応する目的関数値 $\underline{z} = z(\underline{x})$ を**下界値**とする.

3 釘付けテスト

通常の 0-1 ナップサック問題については, **釘付けテスト** [2, 1, 3] が知られており, これによって変数の一部を 0 または 1 に固定して, 問題サイズを (大幅に) 減少することが出来る. 本節では, この方法が PCKP (や DCKP) にも適用出来ることを示す.

3.1 釘付けテスト

PCKP において, 最適ラグランジュ乗数 $\boldsymbol{\lambda}^\dagger$, 上界値 $\bar{z} = \bar{z}(\boldsymbol{\lambda}^\dagger)$ と下界値 $\underline{z}$ が得られているとする.

$$\bar{p}_j = p_j + \sum_{k \in S_j^+} \lambda_{jk} - \sum_{i \in S_j^-} \lambda_{ij} \tag{16}$$

と置くと, ラグランジュ緩和問題は下のように書ける.
LPCKP($\boldsymbol{\lambda}^\dagger$):

$$\text{maximize} \quad \sum_{j=1}^{n} \bar{p}_j x_j \tag{17}$$

$$\text{subject to} \quad \sum_{i=j}^{n} w_j x_j \leq C \tag{18}$$

$$0 \leq x_j \leq 1, \quad \forall j \tag{19}$$

上で, 変数 x_l を 0 または 1 に固定して得られる問題の最適目的関数値をそれぞれ $\bar{z}_l^0$, $\bar{z}_l^1$ と記す.

ここで, もし

$$\bar{z}_l^0 < \underline{z} \tag{20}$$

ならば, PCKP の最適解 $x^* = (x_j^*)$ において $x_l^* = 0$ となることはあり得ない. すなわち, x_l^* は $x_l^* = 1$ と固定される. 同様に,

$$\bar{z}_l^1 < \underline{z} \tag{21}$$

の場合は, $x_l^* = 0$ となる. 以上が釘付けテストの原理であるが, 以下では (20), (21) の簡便な判定法を考える.

まず, LPCKP($\boldsymbol{\lambda}^\dagger$) で一般性を失うことなく以下を仮定する.

1° $\bar{p}_j > 0$, $\forall j$
2° 商品は $r_j := \bar{p}_j / w_j$ の大きいものから順に番号付けられている.

商品 i までの累積重量と累積利得を

$$W_i := \sum_{j=1}^{i} w_j, \qquad P_i := \sum_{j=1}^{i} \bar{p}_j \tag{22}$$

として (W_i, P_i) をプロットすると, 仮定 1°, 2° より, 図 1 のような区分的線形で単調増加な凹関数が得られる.

この折れ線が, 縦軸に平行な直線 $W = C$ と交わる点が上界値 $\bar{z}$ を与えるが, この時の商品 (商品 s とする) を**臨界商品**と呼ぶ. ここで, $l < s$ の商品 l を 0 に固定したときの $\bar{z}_l^0$ を考えると, 図 1 のように商品 l の部分が抜けて, それより右の折れ線が商品 l の分だけ左下に平行移動する.

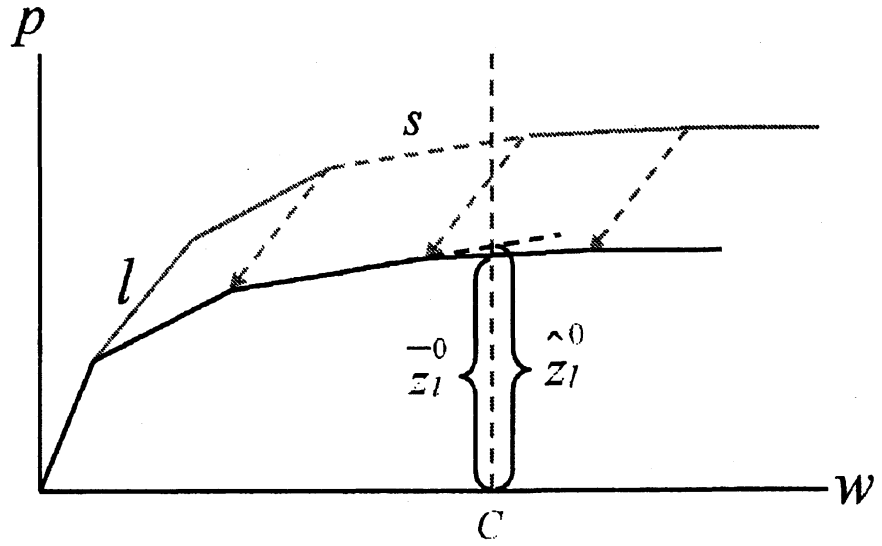

図 1: $x_l = 0$ と固定した場合

この平行移動した折れ線と $W = C$ の交点が $\bar{z}_l^0$ を与えるが, ここでは計算を簡単にするため, 臨界商品 s の直線が平行移動後に $W = C$ と交わる点より, $\bar{z}_l^0$ の上限を

$$\bar{z}_l^0 = \bar{z} - (\bar{p}_l - r_s w_l) \tag{23}$$

と評価する. ここで

$$\theta_l := \bar{p}_l - r_s w_l \tag{24}$$

とおくと, (20) より,

$$\bar{z} - \underline{z} < \theta_l \tag{25}$$

の時, $x_l^* = 1$ と固定されることがわかる. 以下では θ_l を**しきい値**と呼ぶ.

$l > s$ の商品 l を $x_l = 1$ と固定した場合も同様の議論より,

$$\bar{z} - \underline{z} < -\theta_l \tag{26}$$

の時, $x_l^* = 0$ と固定されることが分かる. 以上をまとめると, 次の定理を得る.

定理 3.1 PCKP の最適解 $x^\star = (x_j^\star)$ において,

(i) $\bar{z} - \underline{z} < \theta_j$ ならば, $x_j^\star = 1$

(ii) $\bar{z} - \underline{z} < -\theta_j$ ならば, $x_j^\star = 0$

3.2 ブロック釘付けテスト

PCKP の順序関係を表す有効グラフ $G = (V, E)$ において, 節点 v_i から節点 v_j への有向道があるとき, v_i は v_j の**先行節点**, v_j は v_i の**続行節点**であるといい, 節点 v_i の先行節点の集合及び続行節点の集合をそれぞれ v_i の**上流**, **下流**と呼び, A_i, D_i と記す.

3.1節と同じように, ラグランジュ緩和問題 LPCKP($\lambda^\dagger$) を考え, 商品番号を相対利得 $\bar{p}_j/w_j$ の順につけかえておく. 臨界商品を s とし, 商品 $l < s$ に対してその下流の商品で, 番号が s より小さい商品の集合を D_l' とする. ここで, $x_l = 0$ と固定すると, すべての $j \in D_l'$ が $x_j = 0$ となるので, 図 1 の折れ線が

$$(\sum_{j \in D_l'} w_j, \sum_{j \in D_l'} \bar{p}_j) \tag{27}$$

だけ左下にシフトする. そこで

$$w_l' := \sum_{j \in D_l'} w_j, \quad \bar{p}_l' := \sum_{j \in D_l'} \bar{p}_j \tag{28}$$

として,

$$\Theta_l := \bar{p}_l' - r_s w_l' \tag{29}$$

とおくと,

$$\bar{z} - \underline{z} < \Theta_l \tag{30}$$

の場合, $x_l^\star = 1$ と固定されることがわかる. 以下では Θ_l を**ブロックしきい値**と呼ぶ.

$l > s$ の商品を, $x_l = 1$ と固定した場合も同様で, 以上をまとめて, 次を得る.

定理 3.2 PCKP の最適解 $x^\star = (x_j^\star)$ において,

(i) $\bar{z} - \underline{z} < \Theta_j$ ならば, $x_j^\star = 1$

(ii) $\bar{z} - \underline{z} < -\Theta_j$ ならば, $x_j^\star = 0$

4 仮想釘付けテスト

釘付けテストの効果は上下界値間のギャップに依存する. このギャップが十分小さくない場合, 問題はあまり縮小されないので, 前節の方法の有効性は限られたものとなる. 本節では, これに対処するため**仮想釘付けテスト**を導入する.

4.1 仮想釘付けテストの原理

釘付けテストでは, 上下界値 $\bar{z}$, $\underline{z}$ を入力として, 問題を固定部分とそれ以外の縮小問題に分割した. 上下界値は, 当然

$$\underline{z} \leq z^* \leq \bar{z} \tag{31}$$

の関係を満たしているが, 以下では $[\underline{z}, \bar{z}]$ 間の任意の l を下界値と見たてて釘付けテストを行うことを考える. この l を, **試行値**と呼ぶ.

式 (2)〜(4) を満たす実行可能解 $x = (x_j)$ 全体の集合を $\mathbf{X}$ とする. $\bar{z}$ と l を用いて釘付けテストを実行すると, 一部の x_i が 1 または 0 に固定されるが, l が下界値とは限らないので, この釘付けは正しいとは保証されない. しかし, このように (仮に) 釘付けされた変数の添字集合をそれぞれ $F_1(l), F_0(l)$ として, 次を考える.

縮小問題 $R(l)$:

$$\text{maxmize} \quad \sum_{j=1}^{n} p_j x_j \tag{32}$$

$$\text{subject to} \quad x \in \mathbf{X} \tag{33}$$

$$x_j = 1, \quad j \in F_1(l) \tag{34}$$

$$x_j = 0, \quad j \in F_0(l) \tag{35}$$

ただし, $R(l)$ が実行不可能なら, $z_l^* = -\infty$ とする. このとき, 以下が成立する.

定理 4.1

(i) $l \leq z^* \Rightarrow z_l^* = z^*$

(ii) $l > z^* \Rightarrow z_l^* \leq z^*$

(iii) $l \leq l' \Rightarrow z_l^* \geq z_{l'}^*$. 特に, $R(l)$ が実行不可能 $\Rightarrow$ $R(l')$ も実行不可能.

(iv) $l \leq z_l^* \Rightarrow z_l^* = z^*$

4.2 仮想釘付けアルゴリズム

上の定理より, $[\underline{z}, \bar{z}]$ 間の適当な l で釘付けテストを行って $R(l)$ を解いたとき, (iv) が成立していれば問題は解けたことになる. このとき, gap $:= \bar{z} - l$ が小さいと, $R(l)$ は元問題よりはるかに小さい問題となっていることが多いので, この解法が有利と期待される. (iv) が成立しない場合の処理を含めて, 以下の解法を提案する.

仮想釘付けテスト

Step 1. $l \leftarrow \max\{\bar{z} - \alpha, \underline{z}\}$
Step 2. 試行値 l で釘付けテストを行い, その結果生じた $R(l)$ を解いて z_l^* を求める.
Step 3. $l \leq z_l^*$ なら **Step 5** へ, そうでなければ **Step 4** へ進む.
Step 4. $l \leftarrow \max\{l - \alpha, z_l^*\}$ として **Step 2** へ戻る.
Step 5. $z_l^* = z^*$ で, 最適解が得られた.

ここで, α は適当な整数で, 試行値を最初 $l = \bar{z} - \alpha$ (と $\underline{z}$ の大きい方) とし, ここで最適解が得られなければ, 試行値をさらに α だけ下げて, $l = l - \alpha$ などとすることを意味している.

5 数値実験：PCKP

本節では，PCKP にラグランジュ緩和と釘付けテストを適用した場合の計算機実験を行い，その性能を評価する．アルゴリズムは ANSI C 言語により実装し，IBM RS/6000 SP44 Model 270(CPU : POWER3-II SMP 2way, 375MHz) 上で実験を行った．

5.1 例題の生成

商品数 n を 1000〜32000 とし，w_j と p_j は以下の相関タイプ [6] によりランダムに発生させたが，以下ではスペースの関係で無相関の場合のみを報告する．

- 無相関 (UNCOR)

  p_j : [1,1000] 間の一様乱数

  w_j : [1,1000] 間の一様乱数，p_j と独立

- 弱相関 (WEAK)

  w_j : [1,1000] 間の一様乱数

  p_j : $w_j + \theta_j$

  但し，θ_j : [0,200] 間の一様乱数，w_j と独立

ナップサック容量は $C = 250n$ に固定したが，これは商品の平均重量が 500.5 なので，全商品の約半分を収容できることを意味する．また，順序制約は ${}_nC_2 = n(n-1)/2$ 個の可能な対から，確率 $d/(n-1)$ でランダムに抽出した．これより，生成される順序制約式の数は，平均 $m := |E| \cong nd/2$ となる．d は，付加制約の生起率を制御するパラメータで，以下では $d = 0.2, 0.4, 0.8$ のケースを主として実験した．

5.2 NUOPT による厳密解

表 1 は $n = 1000 \sim 8000$ の範囲で PCKP をランダムに生成し，商用ソフト NUOPT[8] で解いた結果で，最適値 z^* と計算時間を示している．ここで，各行は（後出のすべての表においても同様であるが）すべて 10 回の独立な試行についての平均値であり，'備考' は 10 回の試行中で最適解が得られた回数を示している．解けなかったケースは，メモリ不足または CPU 時間が 9999 秒をオーバした場合である．

これより，NUOPT では n が 4000 以上になると，解けない問題が増えることが分かる．

5.3 釘付けテスト

表 2 は n を 1000〜32000 として通常の釘付けテスト (PLAIN) とブロック釘付けテスト (BLOCK) を適用したときの問題縮小率と計算時間を計測したものである．ここで，それぞれの釘付けテストによって得られた縮小問題のサイズを n'(商品数), m'(順序制約数) とすると，縮小率は

$$\text{縮小率} = 100 \cdot \sqrt{n'm'/nm} \quad (\%) \tag{36}$$

により計算している．

表 1: NUOPT による計算 (PCKP)

d	n	m	z^*	CPU(秒)	備考
0.2	1000	107.4	399135.8	8.25	10
	2000	200.8	801011.2	18.33	10
	4000	395.8	1605890.3	41.70	6
	8000	800.2	3211493.6	6328.08	8
0.4	1000	206.2	395072.7	11.19	10
	2000	402.9	792372.0	28.84	10
	4000	807.1	1584061.0	86.79	6
	8000	1593.0	3181313.7	422.99	6
0.8	1000	406.0	386432.9	15.99	10
	2000	802.3	776369.8	88.92	10
	4000	1606.3	1556960.0	141.74	7
	8000	3196.7	3092626.7	188.00	3

表 2: 縮小率と計算時間 (PCKP)

d	n	PLAIN		BLOCK	
		縮小率 (%)	CPU(秒)	縮小率 (%)	CPU(秒)
0.2	1000	9.47	0.35	8.08	0.35
	2000	4.64	1.10	3.92	1.03
	4000	6.22	3.63	5.29	3.57
	8000	3.53	13.74	3.14	13.86
	16000	3.21	46.13	2.74	46.28
	32000	3.61	272.24	3.05	269.93
0.4	1000	12.57	0.63	10.68	0.62
	2000	19.31	2.17	16.58	2.23
	4000	15.53	9.74	13.33	9.73
	8000	17.51	29.09	15.98	29.04
	16000	11.45	93.54	9.41	94.49
	32000	23.82	411.05	19.79	409.69
0.8	1000	28.79	1.50	23.85	1.48
	2000	28.32	7.23	24.54	7.25
	4000	31.47	32.21	26.20	32.18
	8000	21.69	90.84	17.82	85.96
	16000	17.20	412.67	14.08	453.77
	32000	46.90	1431.22	39.41	1640.37

表 2 から以下の所見が得られる.

- 問題が大きくなればなるほど縮小に時間がかかるが, 計算時間の大半は, 上下界値の算出に費やされている.
- 同一商品数の場合, d の値が大きい程縮小効果は小さくなる.
- 通常の釘付けテストとブロック釘付けテストは計算時間はほぼ同等であるが, 縮小効果は後者の方がやや大きい.

上の実験より, 以後釘付けは専らブロック釘付けテストによることとする.

5.4 ブロック釘付けによる厳密解法

PCKP にブロック釘付けテストを適用し, 縮小された問題を NUOPT で解くことによって, 元の問題の厳密解が得られる. 表 3 は, この方法の実験結果である. 表には問題のラグランジュ緩和による上界値 $\bar{z}$, グリーディ解法 + 2-opt による下界値 $\underline{z}$, 計算時間 (上下界値の算出+ブロック釘付けテスト+NUOPT) が示されている. '備考' は 10 回の試行で最適解を得た回数であり, 最適解が得られない場合の理由は 5.2 節と同様である.

表 1 と, 表 3 の比較から以下の所見が得られる.

- NUOPT による直接解法に比べ, より大型の問題が解け, 計算時間も短縮された.
- n が数千個の問題まで解けるようになった.

表 3: ブロック釘付けによる厳密解法 (PCKP)

d	n	m	$\bar{z}$	$\underline{z}$	CPU(秒)	備考
0.2	1000	107.4	399146.1	399086.6	0.85	10
	2000	201.2	801017.4	800980.4	1.95	10
	4000	395.8	1607429.7	1607384.1	18.31	10
	8000	800.2	3209157.4	3209133.7	280.67	10
	16000	1609.1	6415325.9	6415303.3	196.74	8
	32000	3223.3	12844179.4	12844154.3	728.30	5
0.4	1000	206.2	395083.8	394998.5	1.48	10
	2000	402.9	792403.4	792273.7	11.16	10
	4000	807.1	1589992.6	1589881.3	11.55	9
	8000	1593.0	3177971.2	3177809.3	65.88	8
	16000	3210.3	6350769.5	6350695.1	4718.12	8
	32000	6422.2	12716665.8	12716508.8	637.42	4
0.8	1000	406.0	386472.3	386280.4	4.77	10
	2000	802.3	776405.5	776176.8	21.78	10
	4000	1606.3	1555481.6	1555254.0	60.79	10
	8000	3196.7	3115075.0	3114930.9	99.07	7
	16000	6402.6	6222329.4	6222216.7	611.28	9
	32000	12822.3	12458771.2	12458408.9	1382.28	1

5.5 仮想釘付けテストによる厳密解法

第4節で述べた仮想釘付けテストを用いて, 前節で解けなかった問題を解くことを試みる. 表4は $\alpha = 10$ として仮想釘付けテストを適用し, 縮小問題をNUOPTで厳密に解いた結果である. 各表にはラグランジュ緩和による上界値 $\bar{z}$, 最適値 z^*, 仮想釘付けアルゴリズムにおける (Step 2 ~ Step 4 の) 反復回数, 計算時間 (仮想釘付けテスト+NUOPT) が示されている. ここで, '備考' は10回の計算で最適値が得られた回数である.

表5から, ブロック釘付け (表3) に比べ, 解ける問題がさらに増加しており, 解けた場合は計算時間も一般に短くなっていることが分かる.

表 4: 仮想釘付けによる厳密解法 (PCKP)

d	n	m	$\bar{z}$	z^*	反復回数	CPU(秒)	備考
0.2	1000	107.4	399146.1	399135.8	1.4	0.47	10
	2000	201.2	801017.4	801011.2	1.2	1.52	10
	4000	395.8	1607429.7	1607426.7	1.0	5.06	10
	8000	800.2	3209157.4	3209155.0	1.0	210.44	10
	16000	1609.1	6413226.9	6413225.5	1.0	130.23	8
	32000	3200.0	12826159.7	12826152.9	1.6	1994.74	7
0.4	1000	206.2	395083.8	395072.7	1.4	0.75	10
	2000	402.9	792403.4	792372.0	3.6	18.06	10
	4000	807.1	1589992.6	1589989.1	1.0	9.46	10
	8000	1593.0	3177971.2	3177939.9	3.8	293.63	10
	16000	3206.9	6335645.4	6335634.4	1.9	984.54	7
	32000	6394.0	12700397.8	12700395.5	1.0	529.06	4
0.8	1000	406.0	386472.3	386432.9	5.0	3.86	10
	2000	802.3	776405.5	776369.8	4.0	14.02	10
	4000	1606.3	1555481.6	1555448.2	4.0	68.57	10
	8000	3197.8	3117422.2	3117417.2	1.1	75.77	9
	16000	6402.6	6222329.4	6222318.2	1.9	768.23	10
	32000	12822.3	12386060.0	12386060.0	1.0	1328.15	1

6 数値実験 : DCKP

DCKP についても同様の実験を行った. 例題の生成法は前節と同様である.

6.1 NUOPTによる厳密解

DCKP を商用ソフトNUOPT で解いた時の結果を表5に示す. 各行はランダムに生成した10題の平均値で, 生成子問題数と CPU 時間が示されている. '備考' は10回の試行のうち最適解が得られた回数で, 平均値は計算が成功したもののみについての集計である. 計算が不成功に終る理由は, メモリー領域の不足である.

表 5: NUOPT による厳密解 (DCKP)

d	n	m	生成子問題数	CPU(秒)	備考
0.2	1000	99.0	2014.0	9.61	10
	2000	197.0	2228.1	24.95	10
	4000	394.3	3191.8	83.35	9
	8000	805.0	4717.1	223.88	7
	16000	1598.4	3595.4	600.55	7
0.4	1000	198.9	1362.1	9.11	10
	2000	396.5	2347.4	26.62	10
	4000	792.1	2161.2	88.14	9
	8000	1605.6	1946.0	179.24	9
	16000	3215.3	2304.6	427.22	3
0.8	1000	396.9	1246.8	14.11	10
	2000	798.6	3229.4	76.36	10
	4000	1600.0	1560.0	92.84	9
	8000	3221.8	1683.4	161.56	5
	16000	6411.0	-	-	0

表 5 より, 次の所見を得る.

- 商品数 $n = 4000$ あたりから解けない問題が出現し, $n = 16000$ では, ほとんどの問題が困難になる.
- n, d の増加とともに計算時間が増加し, 解ける問題数も減少して, 問題は困難になる.

6.2 釘付けテストによる厳密解

表 6 に釘付けテストの問題縮小効果を示す. ここで, $\bar{z}$, $\underline{z}$ はそれぞれラグランジュ緩和による上界値とグリーディ解法による下界値で, gap:=$\bar{z} - \underline{z}$ である. n' と m' は縮小問題のサイズで, それぞれ変数の数と制約式数を表す. 縮小率は $\sqrt{n'm'/nm}$ として計算している.

表 7 は, 計算時間を示している. ここで, CPU_1, CPU_2 はそれぞれ上下界値の計算と縮小問題の NUOPT による求解に要した問題で, CPU_T はそれらの和である.

これらの表より, d の増加とともに gap が大幅に増え, 釘付けテストによる問題縮小効果が激減することが分かる.

6.3 仮想釘付けテストによる厳密解

より大型の問題として, $n = 16000$〜64000 程度の問題を考える. この場合, 排他制約の数が多いとどの方法でも厳密解を得ることは難しいので, d の値を $d = 0.05$〜0.10 として実験した. 結果を表 8 に示す. ここで, 仮想釘付けの欄で '反復回数' とあるのは, 4.2 節のアルゴリズムでの Step 2 〜 Step 4 の反復回数であり, 'NUOPT' の欄は NUOPT による直接解法を示している. '回数' は, いずれも 10 例題中それぞれの方法で最適解が得られた回数を示す. いずれの場合も仮想釘付けテストの方が多くの問題が解け, 計算時間でも仮想釘付けが勝っている.

表 6: 釘付けテストによる問題縮小 (DCKP)

d	n	m	$\bar{z}$	$\underline{z}$	gap	n'	m'	縮小率
0.2	1000	99.0	394132.6	394077.2	55.4	115.6	8.5	0.10
	2000	197.0	794909.3	794884.7	24.6	105.1	5.8	0.04
	4000	394.3	1584789.9	1584744.8	45.1	385.3	31.0	0.09
	8000	805.0	3162459.6	3162425.3	34.3	566.6	48.6	0.06
	16000	1598.4	6328402.4	6328376.5	25.9	885.6	66.8	0.05
0.4	1000	198.9	386437.4	386352.7	84.7	164.6	27.2	0.15
	2000	396.5	778653.3	778567.3	86.0	302.5	91.7	0.16
	4000	792.1	1553682.9	1553597.4	85.5	696.7	121.1	0.16
	8000	1605.6	3097011.0	3096725.8	285.2	3236.2	621.7	0.40
	16000	3215.3	6199034.2	6198603.8	430.4	8935.4	1728.7	0.55
0.8	1000	396.9	370545.3	370023.7	521.6	647.9	242.9	0.63
	2000	798.6	747467.6	746674.1	793.5	1596.9	625.8	0.79
	4000	1600.0	1492060.0	1490449.4	1610.6	3899.8	1558.8	0.97
	8000	3221.8	2975262.2	2972009.1	3253.1	8000.0	3221.8	1.00
	16000	6411.0	5947688.4	5941536.0	6152.4	16000.0	6411.8	1.00

表 7: 釘付けテストによる厳密解法 (DCKP)

d	n	m	CPU_1	CPU_2	CPU_T	備考
0.2	1000	99.0	0.18	0.49	0.67	10
	2000	197.0	0.51	0.56	1.07	10
	4000	394.3	1.53	4.11	5.64	10
	8000	805.0	4.12	15.58	19.69	10
	16000	1598.4	11.79	18.19	29.98	10
0.4	1000	198.7	0.32	1.57	1.89	10
	2000	396.5	0.85	5.48	6.34	10
	4000	792.1	2.22	18.39	20.61	10
	8000	1605.6	9.93	59.51	69.45	10
	16000	3215.3	31.24	111.42	141.68	6
0.8	1000	396.9	0.63	8.38	9.01	10
	2000	798.6	1.97	63.71	65.67	10
	4000	1600.0	5.72	108.97	114.68	9
	8000	3221.8	20.98	160.98	182.08	5
	16000	6411.0	54.05	-	-	0

表 8: 仮想釘付けによる厳密解法 (DCKP)

d	n	m	仮想釘付け			NUOPT	
			反復回数	CPU	回数	CPU	回数
0.05	16000	408.5	1.0	10.6	10	688.7	7
	32000	801.2	1.0	30.1	10	1122.3	4
	64000	1582.8	1.0	1676.6	10	1760.5	2
0.1	16000	802.6	1.0	12.6	10	588.6	7
	32000	1605.0	1.4	80.4	10	137.2	1
	64000	3182.3	1.3	1355.3	9	1760.9	2

7 まとめ

順序制約や排他制約など, 付加制約の付いたナップサック問題において, これらの制約式をラグランジュ緩和することにより釘付けテストを適用可能とし, さらに仮想釘付けテストを導入して, 商品数が数千から数万程度の問題を解くことに成功した. 今後, α の調整法を工夫するなどしてより大規模な問題を解くことを試みる予定である.

参考文献

[1] Dembo, R. S. and Hammer, P. L., "A reduction algorithm for knapsack problems", *Methods of Operations Research*, Vol. 36, pp. 49-60, 1980.

[2] Fayard, D. and Plateau, G., "Resolution of the 0-1 knapsack problem: comparison of methods", *Mathematical Programming*, Vol. 8, pp. 272-307, 1975.

[3] Ingargiola, G. P. and Korsh, J. F., "A reduction algorithm for zero-one single knapsack problems", *Management Science*, Vol. 20, pp. 460-463, 1973.

[4] Kellerer, H., Pferschy, U. and Pisinger, D., *Knapsack Problems*, Springer Verlag, 2004.

[5] 今野 浩, 鈴木久敏 (編), 整数計画法と組合せ最適化, 日科技連, 1982.

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

[7] Samphaiboon, N., Yamada, T., "Heuristic and exact algorithms for the precedence-constrained knapsack problem", *JOTA*, Vol. 105pp. 659-676, 2002.

[8] (株) 数理システム, NUOPT Manual, http://www.msi.co.jp/nuopt, 2002.

[9] Yamada, T., Kataoka, S. and Watanabe, K., "Heuristic and exact algorithms for disjunctively constrained knapsack problem", *IPS Journal*, Vol. 43, pp. 2864-2870, 2002.