多次元 0-1 ナップサック問題に対する発見的解法 — 一部分列挙的連続緩和解を用いた順列空間の多点探索*

沼田 一道[†]・秀方友一郎[‡]

A Meta-heuristic Algorithm for the Multidimensional 0-1 Knapsack Problem — A Multi-point Local Search in Permutation Space Using Partially Enumerative Continuous Relaxations*

Kazumiti NUMATA[†] and Yuichiro HIDEKATA[‡]

In this paper, we propose a new meta-heuristic algorithm for the multidimensional 0-1 knapsack problem (MKP). The proposed method is based on the following four steps. First, a local search is performed in permutation space, where each permutation gives an order of project selection. Second, an initial point (permutation) is determined for the local search by using the information obtained from the solution of a continuously relaxed MKP. Third, a reduction by selection bias (RSB) meta-strategy is applied. Fourth, multi-start points (permutations) are generated by solving the continuously relaxed MKPs with the enumeratively assumed values of some of the variables.

To evaluate the proposed method, we used it to solved 60 benchmark problems in the OR-Library and 8 problems in the HCES (Hearin Center for Enterprise Science) resources, and compared the objective values and execution times to those obtained by using existing methods. The results show that the proposed algorithm competes well with the existing ones that are known to be effective.

1. はじめに

多次元 0-1 ナップサック問題 (0-1 Multidimensional Knapsack Problem; 以下 MKP とよぶ) は, たとえば, n 個のプロジェクト案 $\{P_j\}_{j=1,2,\dots,n}$ が与えられたとき, この中から m 種類の資源制約をすべて満たし総期待便益を最大にするようなプロジェクトの部分集合を選び出す問題である. プロジェクト P_j のもたらす期待便益を c_j , それが必要とする第 i 資源の量を a_{ij} , 第 i 資源の利用限度を b_i , プロジェクト P_j を選択するか否かを表す決定変数を x_j とすると, つぎのように定式化される.

$$\begin{cases} \text{maximize} & \sum_{j=1}^n c_j x_j \\ \text{sub. to} & \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, 2, \dots, m \\ & x_j \in \{0, 1\}, \quad j = 1, 2, \dots, n \end{cases}$$

MKP は, 係数を非負に限定した線形 0-1 計画問題であり, NP 困難な問題のクラスに属する.

MKP の解法については古くから多くの研究がなされてきた. 厳密解法としては, 分枝限定法を基本とし上界値の見積もり方法を様々に工夫したものが参考文献 [5, 17] などで提案されている. しかし MKP の場合「問題固有の構造」を利用できる余地が少ないので, 専用の分枝限定法が汎用ソルバーの性能を格段に上回るのは難しいように思われる. たとえば, 100 変数 30 制約程度の「難しい」問題例を lp_solve[3] で解くと, 3GHz の PC で数時間から十数時間を要する. 厳密に最適性を保証するという点では, CPLEX[8] 等の商用ソルバーでも大きな差はない. Osorio ら [13] は CPLEX に独自の制約生成追加機構を付加した CPLEX 強化版を提案しているが, 大きな改善は見られない. CPLEX や Osorio らによる CPLEX 強化版は, n または m が大きくなると時間制限や記憶領域不足で停止している. CPLEX 自身も制約生

* 原稿受付 2006 年 1 月 19 日

† 東京理科大学 工学部 経営工学科 Department of Management Science, Faculty of Engineering, Tokyo University of Science; 1-3 Kagurazaka, Shinjuku ward, Tokyo 162-8601, JAPAN

‡ (株)オーガス総研 Osaka Gas Information System Research Institute Co., Ltd.; 4-13-23 Shibaura, Minato-ku, Tokyo 108-0023, JAPAN

Key Words: multidimensional 0-1 knapsack problem, heuristic algorithm, continuous relaxation, permutation space.

成機能を備えており、両方法では、子問題ごとに生成される制約の保持が厳しいのではないかと想像される。ただし、途中で停止したときの暫定値の精度は高い。この程度の規模の問題をより速く解く、あるいは、より大規模の問題を扱うためには近似解法（発見的解法）が必要となる。

MKP に対する初期の発見的解法としては、Senju ら [16] の有効勾配法が有名である。これは全プロジェクトを選択した実行不能解から出発し、実行可能領域への“接近能率”を指標としてその値の大きいプロジェクトから逐次除外していくものである。Balas ら [1] は連続緩和問題の（緩和最適解に近い）基底解を 0-1 解へ丸めて良い解を探索する発見的解法を提案している。Pirkul [14] は m 個の制約式を 1 個の代理制約式に変換し、いわゆる効率値（代理資源 1 単位当りの利益）を指標とする発見的解法を提案している。このほかにも複数の指標を組み合わせて良い解を構成しようとする様々な構築法が提案されているが、これらは高速である反面、“難しい”問題に対しては精度の点で限界があるように思われる [4]。

精度の点では、メタ戦略を援用した局所探索法（逐次改善法）が優勢である ([6,4,15] など)。これらの解法は解空間 $\{0,1\}^n$ の探索を基本とするが、0-1 パターンを実行可能解に対応させるため何らかの手立てを講じている。Hanafi ら [6] はタブー探索の枠組みで実行可能解から実行不能解へ向かう探索と逆方向の探索を交互に行う方法を提案している。Chu ら [4] は効率性に基づく固定的な採択優先順序によって、また、Sakawa ら [15] は染色体としてコード化された可変の採択優先順序によって致死遺伝子の修復を行う遺伝的アルゴリズムを提案している。

一方、 $\{0,1\}^n$ 空間を探索する代わりに、採択優先順序（順列空間）を探索する方法も存在する。沼田ら [11] は、多点探索と変数の部分固定により問題縮小を繰り返す RSB (Reduction by Selection Bias) 戦略 [9,10] の枠組みで、順列空間を局所探索する方法を提案している。また Numata ら [12] は、順列探索の初期点として、連続緩和問題の最適解および被約費用から導いた採択優先順序が効果的であることを報告している。連続緩和問題の利用と変数の部分固定¹という考え方は、他の文献でも見られる。Vasquez ら [18] は一部の変数の値を部分列挙的に仮定して多数の連続緩和問題を作り、各最適解の値が 0 または 1 に一致した変数をその値に固定して問題サイズを縮小し、縮小された問題を [6] と同様のタブー探索で解いている。

メタ戦略を用いた解法の中では、多点探索を行う遺伝的アルゴリズム [4]（小規模の問題に対しては RSB 戦

略 [11] も）が良好な性能を示している。文献 [18] のタブー探索は精度の点でやや勝るが長大な実行時間を要する²。また、途中終了する CPLEX [8] あるいはその MKP 向け強化版 [13] は近似解法とみなせるが、その精度は [4] より高い。

本論文では MKP に対して、(1) 採択優先順序に対応する順列空間の局所探索 [11]、(2) 連続緩和問題の解情報を利用した初期順列の生成 [12]、(3) 多点出発局所探索の結果得られる多数の局所最適解中の相対的に良いものの一致部分をその値に固定して問題の縮小を繰り返す RSB 戦略 [9,10]、(4) 一部変数の値を列挙的に仮定した連続緩和問題を利用する複数初期順列の生成、を統合した新しい発見的解法を提案する。(1)、(2)、(3) は別々に報告され、その効果が部分的に、あるいは異なる問題で確認されているが、本研究は、新たに提案する (4) を加え、これらを総合してさらなる性能の向上を目指すものである。

提案法は解の探索を順列空間で行うが、解の部分固定は 0-1 ベクトル空間で行う。また、(4) の着想は参考文献 [18] から得ているが、連続緩和問題から得る情報、その利用法は全く異なる。

以下第 2 節では、MKP に対する順列空間の局所探索を導入する。近傍系を定め、探索の手間の削減について述べる。また、局所探索の初期点として、良い解に達する可能性の高い“良い順列”を MKP の連続緩和の情報から生成する規則を与える。第 3 節では、RSB 戦略を導入し、(逐次縮小される) 問題の一部変数の値を列挙的に仮定した連続緩和問題の解情報を用いて、探索の初期点（初期順列）を生成する方法を提案する。第 4 節で、以上を統合した提案法の詳細を述べる。第 5 節では、提案法を用いて、OR-Library [2] 中の 100/250 変数 30 制約の問題 30 問ずつ、および、HCES (Hearin Center for Enterprise Science) のウェブサイト [7] で公開されているベンチマーク問題から 8 問の計 68 問を解いた結果を報告する。提案法は、前者の 60 問で、Chu ら [4] の遺伝的アルゴリズムをやや上回り、“暫定値（解）を出力とする”近似解法としての CPLEX [8] あるいは Osirio ら [13] の CPLEX 強化版に十分匹敵する性能を示している。また、後者では、同所で公開されている最良値³を 8 問中 5 問で更新し、残り 3 問で同じ値を求めている。最後に、第 6 節で提案法とその意義をまとめ、今後の課題を示す。

¹いわゆる「釘付けテスト」は、“難しい”多次元ナップサック問題に対しては効果がない [18]。我々も 100 変数 30 制約の問題例において、「釘付け条件」を満たす状況を見いだせなかった。

²500 変数 30 制約の問題例ではあるが 30~50 時間という所要時間は近似解法としては破格である。参考文献 [18] は 500 変数問題の結果しか報告していない。

³HCES は参考文献 [13] の著者らが拠るサイトなので、CPLEX あるいは CPLEX 強化版で求めたものと思われる。

2. 順列空間の探索

MKP の解探索を考えると、その定式化からは、0-1 ベクトルの集合 $\{0,1\}^n$ を探索空間とするのが自然である。実際、多くの探索的解法は基本解空間 $\{0,1\}^n$ を探索している。しかし、基本解空間で局所探索を行う場合、近傍内の要素が必ずしも実行可能解ではないので、近傍を介した「解同士のつながり」が損なわれる恐れがある。任意の 0-1 ベクトルを実行可能ベクトル（解）に修復することは可能であるが、修復を行うと「解の近さ」は損なわれる。

一方、解を直接作り出す構築法や遺伝的アルゴリズムの修復オペレータでは、「プロジェクトの採択優先順序」が陰的あるいは補助的に利用されている [16,4,15]。沼田ら [11] はこの採択順序を順列空間内で直接的に近傍探索を試み、その効果を報告している。採択優先順序（順列）は、制約条件を明示的に考慮することなく、MKP の実行可能解に自然に対応するので、近傍を介した探索において「解（順列）同士のつながり」を保つことが期待される。

順列空間の要素数 $n!$ は基本解空間の要素数 2^n と比べると大きいですが、これは多数の順列が同一の実行可能解（0-1 ベクトル）に対応することを意味する。実際、最適解ベクトルにおける“1”の個数を k とするとき、少なくとも $k!(n-k)!$ 個の順列が最適解に対応している。探索に用いる近傍の大きさは、一般的な“交換近傍”の場合、 $n(n-1)/2$ である。これは基本解空間で“2 近傍”¹を採用した場合と同じであり、近傍内を探索する手間が問題になることはない。

本論文で提案する解法は順列空間の局所探索を基本とする。

2.1 順列と実行可能解

まず、順列から実行可能解への対応を規定する。MKP の変数の添字集合を $N = \{1, 2, \dots, n\}$ 、 N の要素から成る順列全体の集合を S_N と書く。順列は、採択優先順序の高い方から順に並べたプロジェクト番号の列を表わす。順列 $\sigma = (\sigma(1), \sigma(2), \dots, \sigma(n))$ に対して MKP の実行可能解 $\mathbf{x}^\sigma = (x_1^\sigma, x_2^\sigma, \dots, x_n^\sigma)$ をつぎのように対応させる²。

$$x_{\sigma(i)}^\sigma = \begin{cases} 1, & \mathbf{a}_{\sigma(i)} \leq \mathbf{b} - \sum_{k=1}^{i-1} \mathbf{a}_{\sigma(k)} x_{\sigma(k)}^\sigma \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

ただし、 $\mathbf{a}_j$ は MKP の係数行列の変数 x_j に対応する列ベクトル、 $\mathbf{b}$ は右辺ベクトルである。ベクトル間の不等号は要素ごとにその関係を満たすことを意味する。すべての順列が実行可能解に写像されるので、MKP の m 個の制約条件を明示的に考慮する必要はない。

¹0-1 ベクトルにおいて 2 要素までの反転を許す。

² $\mathbf{x}^\sigma = (x_{\sigma(1)}^\sigma, x_{\sigma(2)}^\sigma, \dots, x_{\sigma(n)}^\sigma)$ と値の決まる順に並べて書くこともあるが、解としては両者を同一視する。

2.2 近傍探索と探索の節約

提案法では、[11] と同様に、交換近傍系を採用する。順列 σ の交換近傍 $U(\sigma)$ は、 σ の任意の 2 要素を交換してできる順列全体の集合であり、以下のように書ける。

$$U(\sigma) = \{ \tau \in S_N \mid 1 \leq i < j \leq n : \tau(i) = \sigma(j), \tau(j) = \sigma(i); \forall k \neq i, j : \tau(k) = \sigma(k) \}$$

現在順列を σ とするとき、近傍 $U(\sigma)$ を探索し、その中の最良順列 τ^* を次の現在順列とする。最良順列 τ^* とは、近傍内で最大の目的関数値 $\mathbf{c}^\top \mathbf{x}^{\tau^*}$ を与える順列のことである。ただし $\mathbf{c}$ は MKP の目的関数の係数を並べたベクトルである。最良順列の与える目的関数値が現在順列の与える目的関数値を越えない場合は探索を終了する。初期順列 σ_0 から探索を開始し、近傍内を探索して最良解へ移行する（best move）ことを繰り返して、局所最適順列 σ^* に至る。

近傍 $U(\sigma)$ の大きさは $\mathcal{O}(n^2)$ であるが、近傍探索の際、その全要素について目的関数値 $(\mathbf{c}^\top \mathbf{x}^\tau)$ を計算する必要はない。上述したように多数の順列が一つの実行可能解に対応しているので、現在解 $(\mathbf{x}^\sigma)$ 自身に対応する近傍内順列の評価を省略することができる。省略可能な順列（交換）は $\mathbf{x}^\sigma$ の部分列のパターン（0/1 の並び方）によって決まる。このような性質は [11] でも一部利用されているが、ここでは、提案法で検知・利用している部分列のパターンを明示しておく。以下では、順列 σ の i 番目の要素 $\sigma(i)$ と j 番目の要素 $\sigma(j)$ を交換して得られる順列を $(i, j) \cdot \sigma$ で表わす。

【命題 1】 $\mathbf{x}^\sigma = (x_{\sigma(1)}^\sigma, x_{\sigma(2)}^\sigma, \dots, x_{\sigma(n)}^\sigma)$ において、

$$x_{\sigma(k)}^\sigma = x_{\sigma(k+1)}^\sigma = \dots = x_{\sigma(\ell)}^\sigma = 0 \text{ または } 1 \\ \Rightarrow k \leq \forall i < \forall j \leq \ell \quad \mathbf{x}^{(i,j) \cdot \sigma} = \mathbf{x}^\sigma.$$

この命題が成立することは σ から $\mathbf{x}^\sigma$ への写像を定義した (1) 式から明らかである。これにより、 $\mathbf{x}^\sigma$ の要素を σ の順（値の決まる順）に見たとき 0 または 1 が並んでいる部分があれば、 σ の対応部分の 2 要素を交換してできる順列は近傍探索から除いてもよいということになる。

【命題 2】 $\mathbf{x}^\sigma = (x_{\sigma(1)}^\sigma, \dots, x_{\sigma(p)}^\sigma, \dots, x_{\sigma(q)}^\sigma, \dots, x_{\sigma(n)}^\sigma)$ (ただし $p < q$) において、

$x_{\sigma(p)}^\sigma = 0$ かつ $x_{\sigma(q)}^\sigma = 1 \Rightarrow \mathbf{x}^{(p,q) \cdot \sigma} = \mathbf{x}^\sigma$.
 $\mathbf{x}^\sigma$ の要素を σ の順（値の決まる順）に見たとき、0 が $(p$ 番目に) 現れてから 1 が $(q$ 番目に) 現れるならば、 $\sigma(p)$ と $\sigma(q)$ を交換してできる順列は近傍探索から除いてもよいということである。交換によって $x_{\sigma(p)}^\sigma, x_{\sigma(p+1)}^\sigma, \dots, x_{\sigma(q)}^\sigma$ が不変（したがって $\mathbf{x}^{(p,q) \cdot \sigma} = \mathbf{x}^\sigma$ ）であることは明らかである。

【命題 3】 $\sigma = (\sigma(1), \dots, \sigma(p), \dots, \sigma(q), \dots, \sigma(n))$ と $(p, q) \cdot \sigma = \tau = (\sigma(1), \dots, \sigma(q), \dots, \sigma(p), \dots, \sigma(n))$ (ただし $p < q$) について、

$$x_{\sigma(p)}^\sigma = 0 \text{ かつ } x_{\tau(p)}^\tau = 0 \\ \Rightarrow \mathbf{x}^\tau = \mathbf{x}^{(p,q) \cdot \sigma} = \mathbf{x}^\sigma.$$

プロジェクト $\sigma(p)$ の優先順位は σ では p 番目, τ では q 番目, $\sigma(q)$ の優先順位は σ では q 番目, τ では p 番目であり, どちらも優先順位の高い p 番目で不採択なので, 低い q でも不採択のはずである. $\sigma(p)$ と $\sigma(q)$ 以外のプロジェクトは優先順位と採否判断時の詰め込み状況が同一なので, $\mathbf{x}^\tau = \mathbf{x}^{(p,q) \cdot \sigma} = \mathbf{x}^\sigma$ となる. 要するに, σ に互換 (p,q) を施した順列 $\tau = (p,q) \cdot \sigma \in U(\sigma)$ を検査するとき, $x_{\sigma(p)}^\sigma$ が 0 であり, かつ $\mathbf{x}^\tau$ を p 番目まで評価して $x_{\tau(p)}^\tau = 0$ と判明すれば, 以後の処理は不必要ということである.

交換近傍の探索に際して以上の性質を利用すると, 探索時間を 2~3 割減らすことができる.

2.3 順列探索の初期点

局所探索を行うには初期点 (順列) が必要である. 参考文献 [11] はランダムに生成した順列を初期点としているが, 参考文献 [12] は連続緩和問題の解情報を用いた初期順列の生成法を提案し, それを出発点とする探索が良い解に達する可能性の高いことを報告している. 提案法は参考文献 [12] の方法をさらに組織的な枠組みで用いるものであるが, ここでは, 基礎となる [12] の順列生成法を示しておく.

まず, MKP において制約条件 $\{x_j \in \{0, 1\}\}$ を $\{0 \leq x_j \leq 1\}$ に緩和した連続緩和問題 $\overline{\text{MKP}}$ を考える. $\overline{\text{MKP}}$ の最適基底解を $\mathbf{x}^* = (x_1^*, x_2^*, \dots, x_n^*)$, 最適基底における被約費用を $\mathbf{c}^* = (c_1^*, c_2^*, \dots, c_n^*)$ とし, プロジェクト j の採択優先度 γ_j を

$$\gamma_j = x_j^* + c_j^* \quad (2)$$

で定義する. 最適基底において各変数 (x_j) は, 0 と 1 の間の値をとる基底変数で被約費用が 0 (γ は 0 以上 1 以下) であるか, 値 1 をとる非基底変数で被約費用が正值 (γ は 1 以上) であるか, 値 0 をとる非基底変数で被約費用が負値 (γ は 0 以下) であるか, のいずれかである. x_j はプロジェクト j を採択するか否かを表す変数であるから, 採択の望ましい方から順に, 値 1 をとるもの, 0 と 1 の間の値をとるもの, 値 0 をとるもの, とするのが妥当と考えられる. γ の値は 3 グループ間の採択の望ましさを正しく反映している (Fig. 1).

各グループ内においても採択の望ましさを度合いは γ の値の大小関係と一致する. 基底変数の中では, $\gamma_j = x_j^*$ であり, γ_j の順序と x_j^* の順序は一致する. 値 1 をとる非基底変数の中では, 当該プロジェクトを採択 (1) から除外 (0) へ変更したときの目的関数の減少を表す $c_j^* = \gamma_j - 1$ の大きい (すなわち γ の大きい) ものを優先するのが有利であろう. 同様に値 0 をとる非基底変数の中では, 除外 (0) から採択 (1) へ変更したときの目的関数の減少を表す $-c_j^* = -\gamma_j$ の小さい (すなわち γ の大きい) プロジェクトを優先するのが自然であろう.

以上より, 先に定義した γ_j は, 緩和最適解に “近い”

Fig. 1 Index γ (priority measure of projects)

解をもたらす “良い” 採択優先度を反映するものと考えられる. 提案法では, 与えられた MKP (およびそれから派生する MKP) の連続緩和問題の解情報から γ 値を計算し, その降順にプロジェクト番号を並べた順列を各探索の初期点として用いる.

3. RSB 戦略の適用

RSB (Reduction by Selection Bias) 戦略は, 組合せ最適化問題 P に対する発見的解法 X の解探索能力 (精度) を, 計算時間の増加と引換えに向上させる一般的な枠組みであり, 次のような観察 (原理) に基づいている.

- 解法 X は問題 P の規模 (サイズ) が小さいほど良い解 (最適解) を探しやすい.
 - 問題 P の多数の近似解の中で相対的に良い解が一致する部分は P の最適解と一致している可能性が高い.
- ただし, X は想定する規模の問題 P に対して一定程度の精度を有することを前提とする.

参考文献 [9] は, 巡回セールスマン問題を P , 2-opt 法を X として, RSB 戦略を最初に提案し, その効果を検証している. 参考文献 [10] は, 同じく巡回セールスマン問題に対し, X を変形 Lin-Kernighan 法に変更して, RSB 戦略に基づく高精度の解法を提案している. また, 先行研究 [11] は, 多次元ナップサック問題に対して, 順列空間における交換近傍の局所探索を X とした RSB 戦略の適用を提案している.

3.1 RSB 戦略の概要

RSB 戦略の基本的枠組みは以下の通りである.

- (1) (逐次縮小される) 問題 P に対して, 方法 X により, 多数 (L 個) の局所最適解集団を生成する.
- (2) L 個の局所最適解において, 目的関数値が良いほうから順に k 番目までの解すべてで値が一致した変数を, その一致した値に固定して問題を縮小する.
- (3) 問題のサイズ (固定されていない変数の個数) が十分小さくなるまで (1), (2) を繰り返し, それまでに得られた最良の解を出力する.

多次元ナップサック問題を対象問題として、実際に RSB 戦略を適用するには、

- 解法 X
- 多数の局所最適解を生成する方法
- 変数固定/問題縮小の仕方、世代交代方式、 k の設定、誤固定対策

を決める必要がある。

提案法は、参考文献 [11] と同様、「順列空間における交換近傍 $U(\sigma)$ の局所探索」を基本解法 X とする。すなわち、順列空間で探索を行い、基本解空間 $\{0, 1\}^n$ で変数固定/問題縮小を行う。局所解集団の生成法については、参考文献 [12] の規準に参考文献 [18] の考え方を応用したものを 3.2 節で提案する。提案法における変数固定/問題縮小の仕方等については、3.3 節で述べる。これらを記述する際の便宜を考え、以下で逐次縮小される MKP 問題の形式的な表現を与えておく。

まず、 $t-1$ 回縮小された t 回目に対象とする MKP 問題を $P^{(t)}$ で表す。問題 $P^{(t)}$ の目的関数の固定 (定数) 部分は、 $P^{(t)}$ において 0 (1) に固定されている変数の添字集合を $F_0^{(t)}$ ($F_1^{(t)}$) とすると、 $z_{base}^{(t)} = \sum_{j \in F_1^{(t)}} c_j$ で与えられる。また、制約式右辺 (残存資源量) は $b_i^{(t)} = b_i - \sum_{j \in F_1^{(t)}} a_{ij}$ で与えられる ($i=1, 2, \dots, m$)。さらに、固定されていない変数の添え字を元々の番号の若い順に $x_1^{(t)}, x_2^{(t)}, \dots, x_{n^{(t)}}^{(t)}$ と再番号付けする ($n^{(t)} = n - |F_0^{(t)}| - |F_1^{(t)}|$)。これらに対する係数 a_{ij} ($i=1, 2, \dots, m$)、 c_j も同様に番号を付け直し、 $a_{ij}^{(t)}$ ($i=1, 2, \dots, m$)、 $c_j^{(t)}$ ($j=1, 2, \dots, n^{(t)}$) とする。問題 $P^{(t)}$ はつぎのように書ける。

$$\begin{cases} \text{maximize} & \sum_{j=1}^{n^{(t)}} c_j^{(t)} x_j^{(t)} + z_{base}^{(t)} \\ \text{sub. to} & \sum_{j=1}^{n^{(t)}} a_{ij}^{(t)} x_j^{(t)} \leq b_i^{(t)}, \quad i=1, 2, \dots, m \\ & x_j^{(t)} \in \{0, 1\}, \quad j=1, 2, \dots, n^{(t)} \end{cases}$$

3.2 解集団の生成

提案法は、(2) 式で与えられる指標 γ の降順にプロジェクト番号を並べた順列を初期点として、問題 $P^{(t)}$ の局所最適解を生成する。多様性を確保しつつ連続緩和解の情報を利用するために、 $P^{(t)}$ の一部変数の値を列挙的に仮定した緩和問題群を考え、各緩和問題の解情報 γ から L 個の初期順列を得る¹。一部変数の値を部分列挙的に仮定した緩和問題群を利用することは、Vasquez らが [18] で提案している。彼らは、緩和問題群の全緩和解において値が 0 または 1 で一致した変数をその値に固定し、縮小後の問題に対して $\{0, 1\}^n$ 空間のタブー探索を行っている。提案法は「列挙的緩和問題群」というアイデアを [18] に負っているが、利用する解情報、利用の仕方、問題縮小回数等については [18] と明確に異なる。

提案法における緩和問題群の生成は [18] にならって 2 段階で行う。まず $P^{(t)}$ に対して、値を仮定する変数を決める。これは緩和問題 $\overline{P^{(t)}}$ の解を見て 0 か 1 か最も推定しにくい変数、すなわち、「値」が $1/2$ に近いもの²から順に $(|\gamma_j - 0.5|)$ の小さい順に ℓ_1 個選ぶ。このように選んだ変数の順序集合を $H(\ell_1, \overline{P^{(t)}})$ と表記する。ここで、 $[0, 2^{\ell_1} - 1]$ の範囲の整数 u を長さ ℓ_1 の 2 進数で表したときのビットパターンを $u_{<2, \ell_1>}$ で表す。そして、 $P^{(t)}$ について $H(\ell_1, \overline{P^{(t)}})$ 内の変数に、 $u_{<2, \ell_1>}$ の各ビットを順に割り当てた (仮定した) 問題を

$$Q^{(t)}(u) \equiv P^{(t)} \left[H(\ell_1, \overline{P^{(t)}}) \leftarrow u_{<2, \ell_1>} \right]$$

とする ($u=0, 1, \dots, 2^{\ell_1} - 1$)。さらに、 2^{ℓ_1} 個の問題 $Q^{(t)}(u)$ のそれぞれに対して、 $P^{(t)}$ の場合と同様に、値を仮定する変数を決める。すなわち、緩和問題 $\overline{Q^{(t)}(u)}$ の解を基に $(|\gamma_j - 0.5|)$ の小さい順に ℓ_2 個の変数を選ぶ。このように選んだ変数の順序集合は $H(\ell_2, \overline{Q^{(t)}(u)})$ であり、 u に依存している。最後に、 $Q^{(t)}(u)$ について $H(\ell_2, \overline{Q^{(t)}(u)})$ 内の変数に、 $v_{<2, \ell_2>}$ の各ビットを順に割り当てた (仮定した) 問題を

$$R^{(t)}(u, v) \equiv Q^{(t)}(u) \left[H(\ell_2, \overline{Q^{(t)}(u)}) \leftarrow v_{<2, \ell_2>} \right]$$

とする ($v=0, 1, \dots, 2^{\ell_2} - 1$)。以上の操作により、部分列挙的に変数値を仮定した問題を $1 + 2^{\ell_1} + 2^{\ell_1 + \ell_2}$ 個生成する (Fig. 2)。

Fig. 2 Partial enumeration tree of $P^{(t)}$

生成した問題群の各緩和解情報から計算される指標 γ を用いて、探索の初期点となる順列を $1 + 2^{\ell_1} + 2^{\ell_1 + \ell_2}$ 個作成する。 $P^{(t)}$ については、 $N^{(t)} = \{1, 2, \dots, n^{(t)}\}$ の要素を $\overline{P^{(t)}}$ における γ の降順に並べたもの ($\sigma^{\gamma(\overline{P^{(t)}})}$) を初期順列とする (Fig. 3(a))。 $Q^{(t)}(u)$ については、 $N^{(t)} - H(\ell_1, u)$ の要素を $\overline{Q^{(t)}(u)}$ における γ の降順に並

¹実際には得られた個数を L とする。

²文献 [18] では「値」として x_j^* を用いているが、我々は γ_j を用いる。基底変数であれば両者は一致する。

べ $(\sigma^{\gamma(\overline{Q(u)})})$, その前後に $H(\ell_1, \overline{P^{(t)}})$ 内の変数番号を付加したものを初期順列とする. ただし, 値1を仮定したものを前に (優先順位の高いほうに), 0と仮定したものを後ろに補う (Fig. 3(b)). $R^{(t)}(u, v)$ については, $N^{(t)} - H(\ell_1, u) - H(\ell_2, v)$ の要素を $R^{(t)}(u, v)$ における γ の降順に並べ $(\sigma^{\gamma(\overline{R(u, v)})})$, その前後に $H(\ell_2, \overline{Q^{(t)}(u)})$ 内の変数番号を付加し, さらに, その前後に $H(\ell_1, \overline{P^{(t)}})$ 内の変数番号を付加したものを初期順列とする. ただし, 値1を仮定したものを前に, 0と仮定したものを後ろに補う (Fig. 3(c)).

Fig. 3 Initial permutations (start points)

以上を初期点とする局所探索を $N^{(t)} = \{1, 2, \dots, n^{(t)}\}$ 上の順列空間で行い, $L = 1 + 2^{\ell_1} + 2^{\ell_1 + \ell_2}$ 個の局所最適解を得る¹.

3.3 変数の固定/問題の縮小

$P^{(t)}$ に対し, (重複を含めて) L 個の 0-1 局所最適解が求まると, これらをもとに値 (0-1) が一致した変数をその値に固定し, つぎの縮小された問題 $P^{(t+1)}$ を作る. このとき, 以下を検討しておく必要がある.

- 世代交代方式
- k (上位解個数) の設定
- 誤固定対策

(1) 世代交代方式 RSB 戦略は逐次縮小される問題に解法 X を繰り返し適用するので, 原則的には, 各世代の解集団は世代間で独立に生成される. 親世代の良い解をランダムに小変更して子世代の探索の初期点とし, 2 世代 $2L$ 個の解から上位 L 個を選んで子世代とする方式 [10] も提案されているが, 保持する親世代の解の表現形式と子世代の解の探索形式が違う場合, 「ランダムな小

¹ $n^{(t)}$ あるいは資源上限 $b_i^{(t)}$ が小さく, $\ell_1 + \ell_2$ が大きい場合, u, v の値によっては, $Q^{(t)}(u)$ または $R^{(t)}(u, v)$ が実行不能となる可能性もある. 実行不能となった場合は単にその問題を無視するので, 初期点 (局所最適解) の個数は減ることになる. $n^{(t)}$ を小さくし過ぎなければ, この現象は起きない.

変更」の適用は難しい. また, 探索初期点を確定的規則で組織的に生成する場合, 2 世代から淘汰すると同傾向の解に急速に収束する危険もある. 提案法は問題を陽に縮小し 3.2 の方法で各世代独立に探索初期点を生成する方式を採用する. ただし, 親世代の最良解を次世代に残すため, $L + 1$ 個の解集団から上位 L 個を選ぶエリート戦略を併用する.

(2) k の設定 目的関数値の良いほうから順に (重複を含めず) k 個の解の一致部分をその一致した値に固定する. 予備実験では, k/L が小さいと変数固定が急速に進み過ぎて誤固定が起こりやすかった. そこで, $k = K_{\max}$ から出発して ($K_{\max} = L$ であれば, 最初は全解で一致した変数を固定する), 以後, 1 回あたりの固定が 5 変数以上で最少となるように k を動的に制御することにした. こうすると変数固定はゆっくり進行し, 誤固定は最終段階でまれに起こる程度である. 原則として $K_{\max} = L$ としたが, L が 1024 を超える場合には 1024 で打ち切った. また, n が大きい場合には (実行時間の関係で) 100 で打ち切っている.

(3) 誤固定対策 RSB 戦略は, 誤固定から逃れるために固定状態の変数を一定割合でランダムに未固定状態に戻すことも試みている [10, 11]. 平均精度の向上に有効な手段ではあるが, 解法にランダムな要素が加わる. 探索の初期点をランダムに与えるような場合には自然な対策と思われるが, 提案法のように確定的な初期点を用いる場合には, できるだけ確率的要素の混入を避けたい. 提案法では, 「ランダムな固定解除」を採用せず, (2) で述べたように「1 回あたりの固定変数を抑える」ことで誤固定に対処した.

提案法は上記のような方針で $n^{(t)}$ が m (制約数) 程度になるまで問題の縮小を繰り返し, それまでに現れた最良の解を近似解として出力する. 暫定値が 20 回連続して改善されない場合にも同様に出力を行い終了する.

以下では, 提案法を RSB-LP-perm 法とよぶことにする. 参考文献 [11] の解法は RSB-perm 法とよぶ.

4. RSB-LP-perm 法

RSB-LP-perm 法の全体は以下のように書ける.

RSB-LP-perm 法 (n, m, A, b, c は与えられている)

step0 $L = 1 + 2^{\ell_1} + 2^{\ell_1 + \ell_2}$; (ℓ_1, ℓ_2 : パラメータ)
 $K_{\max} := \min(L, 1024 \text{ または } 100)$; (上位数限界)
 $t := 0$; $n^{(0)} := n$; $N^{(0)} := \{1, 2, \dots, n\}$;
 $F_0^{(0)} := F_1^{(0)} := \emptyset$; $z_{\text{base}}^{(0)} := 0$;
 $\hat{z} := -\infty$; (暫定値) ($\hat{x}$ (暫定解): 不定)

step1 L 個の緩和問題
 $P^{(t)}, Q^{(t)}(u) (u = 0, 1, \dots, 2^{\ell_1} - 1),$
 $R^{(t)}(u, v) (u = 0, 1, \dots, 2^{\ell_1} - 1; v = 0, 1, \dots, 2^{\ell_2} - 1)$
を解く.
指標 γ に基づく順列 (Fig. 3) を出発点として順列

探索を行い、 L 個の局所最適解 (0-1 解) を生成する。 $(z_{base}^{(t)})$ および $F_0^{(t)}$ と $F_1^{(t)}$ を補い、 $P^{(0)} \equiv MKP$ に対する解とする)

$\hat{x}(\hat{z})$ を加えた $L+1$ 個の解のうち、上位 $K_{\max}$ 個を (目的関数値の) 降順に格納する。

暫定値 $\hat{z}$, 暫定解 $\hat{x}$ を更新する。

step2 if $n^{(t)} < m + \ell_1 + \ell_2$ then goto step5;

if 連続非改善反復回数 > 20 then goto step5;

step3 ($t=0$ の場合: $k = K_{\max}$)

$K_{\max}$ 個の解から $F_0^{(1)}$, $F_1^{(1)}$ を抽出する。

($t \geq 1$ の場合: k は動的に調整)

$K_{\max}$ 個中の上位 k 個の解の一致部分の変数添え字集合を $F_0(k)$, $F_1(k)$ とする。

$$k^* := \max_{\substack{1 \leq k \leq K_{\max} \\ d(k) \geq 5}} \{k\}$$

$$d(k) \equiv \left(|F_0(k)| + |F_1(k)| \right) - \left(|F_0^{(t)}| + |F_1^{(t)}| \right)$$

$$F_0^{(t+1)} := F_0(k^*); \quad F_1^{(t+1)} := F_1(k^*);$$

$$n^{(t+1)} := n^{(0)} - |F_0^{(t+1)}| - |F_1^{(t+1)}|;$$

$$N^{(t+1)} := N^{(0)} - F_0^{(t+1)} - F_1^{(t+1)};$$

$$P^{(t+1)}, z_{base}^{(t+1)} \text{ の生成。}$$

step4 $t := t+1$; goto step1;

step5 $\hat{z}$, $\hat{x}$ を近似値 (解) として出力する。

この解法はランダムな要素を含まない確定的なものである。また、解法に含まれるパラメータは、 ℓ_1 , ℓ_2 と $K_{\max}$ だけである。ただし、固定変数の増分下限 5 と連続非改善反復回数 20 は変更可能である。step1 の緩和問題は、初期基底解を有効勾配法 [16] で構成し、0 と 1 を限界値 (非基底変数の値) とする標準的単体法で解いている。

5. 計算機実験

提案した RSB-LP-perm 法の性能を評価するため、OR-Library [2] 中の 60 問と HCES [7] で公開されているベンチマークから 8 問の計 68 問を解いて、参考文献 [4, 11, 13] の結果と比較した。これ以外に、OR-Library の比較的容易な問題 55 問を解いて提案法の基本性能を確認している。実験は PC-AT 互換機 (Pentium4 3GHz, メモリ 512MB, Windows 2000, Delphi 5) の上で行った。対象とした問題はつぎの 4 グループである。

(1) mknapi, mknapi2 (確認用)

(6~105 変数, 2~30 制約; 55 問)

(2) mknapi7 (100 変数, 30 制約; 30 問)

(3) mknapi8 (250 変数, 30 制約; 30 問)

(4) Mk_gk01~Mk_gk08

(100 変数 15 制約~500 変数 50 制約; 各 1 問)

(1) mknapi, mknapi2 (確認用ベンチマーク問題)

これらは諸文献でテスト用に提供された問題例を集めたもので、最適値 (解) はすべて知られている。

パラメータを一律に $\ell_1 = \ell_2 = 3$ ($L = K_{\max} = 1+8+$

$64=73$) と設定した提案法でこれらを解いたところ、どの問題例についても、計算時間 1 秒未満 (10 ミリ秒未満から 730 ミリ秒程度) で最適解を出力した。問題縮小の回数は 1~4 であり、すべての問題例に対して、1 回目 ($t=0$) の探索で最適解を発見している。部分列挙的緩和 (指標 γ) に基づく初期順列と順列探索の効果が大きく、RSB 戦略の働く余地はなかったと考えられる。

遺伝的アルゴリズム [4] や RSB-perm 法 [11] も、これら 55 問題例についてはすべて最適解を求めており、精度の面での優劣はつけにくい。ただし、参考文献 [4, 11] の解法は確率的要素を含むので評価には複数回の試行が必要となるが、提案法は確定的であるため、1 回の試行で十分である。

(2) mknapi7 (最適解既知の難しい問題)

OR-Library が提供する 100 変数 30 制約の問題群である。目的関数の係数 c_j と制約式の係数の和 $\sum_{i=1}^m a_{ij}$ ($j=1, 2, \dots, n$) を相関させ、制約式右辺を $b_i = \alpha \sum_{j=1}^n a_{ij}$ ($i=1, 2, \dots, m$; $\alpha=0.25, 0.5, 0.75$) と設定して作成したものが 30 問用意されている。問題番号 0~9 が $\alpha=0.25$, 10~19 が $\alpha=0.5$, 20~29 が $\alpha=0.75$ に対応する。

提案法のパラメータを一律に $\ell_1 = \ell_2 = 4$ ($L = K_{\max} = 1+16+256=273$) 固定して、これらの 30 問を解いた結果を Table 1 に示す。比較のため、汎用ソルバー lp_solve [3] で厳密に解いた結果も示した。“- : x” の列は提案法で求めた値が最適値に等しい (-) か否 (x) かを表す。“e.time” の列は同一マシン上での lp_solve と提案法の計算時間 (単位は秒) である。提案法の計算時間は、“ $n^{(t)} < m + \ell_1 + \ell_2$ ” または “連続非改善反復回数 > 20 ” が成立して終了するまでの時間である。“ t_f ” の列には最終解を発見した反復番号を、“ $\#n/\#r$ ” の列には終了時点での非固定変数の数 (n) と問題縮小回数 (r) を示す。どの問題も “ $n^{(t)} < m + \ell_1 + \ell_2$ ” の条件で終了している。

提案法は、30 問中 25 問で 1 回目 ($t=0$) の探索時に最終解 (22 問で最適解) を発見している。部分列挙的緩和 (指標 γ) に基づく初期順列と順列探索の効果はこの問題群でも大きい。しかし、5 問では 3~9 回の問題縮小後に最終解 (4 問で最適解) を発見しており、RSB 戦略の効果も確認できる。

提案法のパラメータ ℓ_1 , ℓ_2 については、予備実験の結果から均等に配分した。たとえば 8 個の変数を一度に仮定するより、4 個と 4 個に分けて緩和問題を作成したほうが若干ではあるが良い結果を与える。もちろん、 $\ell_1 + \ell_2$ を大きくすれば精度は上がるが所要時間も増大する。

$\ell_1 = \ell_2 = 4$ で最適解に至らなかった 4 問について、その値を 1 ずつ増やしながら実行したところ、それぞれ Table 2 に示す値で最適解を出力した。表中の “ t_f ”, “ $\#n/\#r$ ”, “e.time” の意味は Table 1 と同様である。

Osorio ら [13] は、GA [4], CPLEX [8] および CPLEX

Table 1 Results for mknpcb7

P No.	lp_solve		RSB-LP-perm(4-4)			
	exact v.	e.time	-:x	t_f	#n/#r	e.time
00	21946	5997	-	1	33/8	21
01	21716	6987	-	1	35/9	25
02	20754	9603	-	1	33/9	25
03	21464	9633	-	7	36/9	27
04	21844	34395	-	1	33/11	30
05	22176	25840	-	1	36/8	24
06	21799	41704	-	1	37/7	23
07	21397	32341	-	1	35/9	27
08	22525	76764	x	1	37/7	21
09	20983	4915	-	1	36/10	31
10	40767	48231	-	3	35/10	38
11	41308	38690	x	1	34/10	36
12	41630	33229	-	1	36/8	32
13	41041	7963	-	1	35/10	38
14	40889	51197	x	1	33/11	46
15	41058	15939	-	7	35/10	38
16	41062	32958	-	1	37/10	40
17	42719	8916	-	3	34/9	32
18	42230	11802	-	1	32/9	33
19	41700	9575	-	1	34/9	33
20	57494	1885	-	1	37/7	26
21	60027	5398	x	5	33/7	23
22	58052	10311	-	1	34/9	33
23	60776	1229	-	1	35/7	23
24	58884	1006	-	1	34/7	23
25	60011	1948	-	1	36/7	24
26	58132	2327	-	1	35/6	25
27	59064	1436	-	1	36/7	25
28	58975	11472	-	1	33/9	32
29	60603	5485	-	1	36/8	30

Table 2 Tuning parameters ℓ_1, ℓ_2 for 4 instances

No. ($\ell_1 = \ell_2 = 4$)	$\ell_1 - \ell_2$	t_f	#n/#r	e.time
08 (x: 22482)	6-5	7	37/9	214
11 (x: 41304)	6-5	10	34/11	341
14 (x: 40872)	5-5	1	33/13	249
21 (x: 60026)	5-5	1	33/9	121

に変更を加えた彼らの提案法を用いて mknpcb7 を解いた結果を、10 問ごとの目的関数値の平均で比較している。これに提案法の結果を加えたものを Table 3 に示す。一律に $\ell_1 = \ell_2 = 4$ とした提案法を RSB(4-4)、Osorio らの方法を Fix+Cuts と表記する。CPLEX と Fix+Cuts

は $\alpha = 0.25$ で 2 問、 $\alpha = 0.50$ で 1 問、 $\alpha = 0.75$ で 8 問を厳密に解いているが、残りの 19 問はすべて 3 時間の時間制限で終了した場合の暫定値である。RSB-perm 法 [11] はパラメータおよび試行ごとの変動があるので、いろいろ試して最良のもので比較すれば遜色ないが、平均的にはこれらの方法に及ばないので割愛した。表中の*印は最良値を意味する。Fix+Cuts の数値は四捨五入したものが報告されている。

Table 3 Comparison of 4 methods for mknpcb7

α	GA	CPLEX	Fix+Cuts	RSB(4-4)
0.25	21654.2	21660.2*	21656	21656.1
0.50	41431.3	41440.4*	41437	41438.3
0.75	59199.1	59201.8*	59202	59201.7

CPLEX の精度が最も高く¹、以下、RSB(4-4)、Fix+Cuts、GA の順に続く。ただし、これらの間の精度の差は小さい。計算時間を比較するのは難しいが、GA については、Silicon Graphics Indigo workstation (R4000 100MHz, 48MB) で、約 600 秒 ($\alpha = 0.25$)、約 780 秒 ($\alpha = 0.5$)、約 900 秒 ($\alpha = 0.75$) と報告されている [4]。CPLEX と Fix+Cuts の実行時間は Windows NT マシン (Pentium III 450MHz, 64MB) で 2~3 時間程度 [13]、提案法の計算時間は Table 1 に示す通りである。

(3) mknpcb8 (最適解未知の難しい問題)

mknpcb8 は 250 変数 30 制約の 30 問で、mknpcb7 と同様、 $\alpha = 0.25, 0.5, 0.75$ の各 10 問からなる。

パラメータを一律に $\ell_1 = \ell_2 = 5$ ($L = 1 + 32 + 1024 = 1057$, $K_{\max} = 100$) とした提案法でこれらを解いた。探索過程を観察すると、100 変数の場合と同様に、初回に見いだした解が最終解となることが多かった。部分列挙的緩和策から導いた初期順列と多点順列探索の効果は 250 変数の問題に対しても大きいと推測される。30 問とも「 $n^{(t)} < m + \ell_1 + \ell_2$ 」の条件で終了し、#n/#r の平均は、(32.9, 11.3), (35.4, 10.7), (35.2, 9.1) であった。

参考文献 [13] は、mknpcb8 の場合にも 10 問ごとの目的関数値の平均で GA、CPLEX および Fix+Cuts を比較している。これに提案法 RSB(5-5) の結果を加えたものを Table 4 に示す。精度を比較すると、GA がやや低く、他の 3 方法はほぼ同等である。3 者は α の値ごと

Table 4 Comparison of 4 methods for mknpcb8

α	GA	CPLEX	Fix+Cuts	RSB(5-5)
0.25	56875.9	56888	56959*	56907.2
0.50	106673.9	106679	106686	106690.4*
0.75	150443.5	150479*	150467	150464.6

¹ $\alpha = 0.25$ の 1 または 2 問を除く全問で最適解を発見して (求めて) いる。

に首位の座（*印）を分け合っている。

計算時間については、GA の場合 Indigo workstation で 1500～2500 秒と報告されている [4]。また、CPLEX と Fix+Cuts の実行時間は 2～3 時間程度と推定される [13]。提案法の平均計算時間は、 $\alpha=0.25$ で約 400 秒、 $\alpha=0.5$ で約 600 秒、 $\alpha=0.75$ で約 500 秒である。計算時間については、GA、提案法、CPLEX と Fix+Cuts の順であるが、前 2 者と後 2 者の間には開きがある。ただし、その差は 100 変数 30 制約の場合より縮まっている。

$K_{\max}=100$ の RSB(5-5) は最初の変数固定で 250 変数から 100 変数程度に問題を縮小するが、この縮小された問題は mknpcb7 の 100 変数問題より “さらに難しい” 問題のように思われる¹。

(4) Mk_gk01～Mk_gk08

HCES (Hearin Center for Enterprise Science) のウェブサイト [7] で公開されている MKP ベンチマーク問題 11 問中の 8 問を、一律に $\ell_1=\ell_2=4$ ($L=K_{\max}=1+16+256=273$) と設定した提案法 RSB(4-4) で解いた。その結果を Table 5 に示す。表中の “ t_f ”, “ $\#n/\#r$ ”, “e.time” の意味は Tables 1, 2 と同様である。

Table 5 Results of proposed method for Mk_gk01...08

No.	size	best known	RSB (4-4)	t_f	$\#n/\#r$	e.time
01	100-15	3766	3766	1	19/13	18
02	100-25	3958	3957	1	30/13	43
03	150-25	5650	5651	1	31/21	180
04	150-50	5764	5764	9	57/16	380
05	200-25	7557	7558	1	33/22	289
06	200-50	7672	7674	20	54/25	1064
07	500-25	19215	19216	1	122/22	2226
08	500-50	18801	18802	29	62/49	10903

RSB(4-4) は 8 問中 5 問で既存の最良値を更新し、2 問で同一値を与えている。Mk_gk02 に対する値は最良値より 1 小さいが、RSB(5-4) に変更して解き直すと、同一の値 3958 を得る²。8 問中 3 問で問題縮小後に最終解を発見しており RSB 戦略もよく機能している。

HCES は参考文献 [13] の著者らが拠るサイトなので、既存の最良値は CPLEX またはその強化版 Fix+Cuts で求めたものと考えられる。この結果は、提案法がこの種の問題例については CPLEX/Fix+Cuts をわずかではあるがしのぐ可能性があることを示している。対象とした 8 問題は、求まった解から、 $\alpha=0.5$ 程度の制約の強さで生成されたと推定される。また、目的関数、制約式

の係数は OR-Library と比べて 1 桁以上小さい値である。さらに、制約数と変数の数の比 m/n は OR-Library より若干大きめに設定されている。これらの条件は、この問題群に比較的強い「組合せ的性質」を与えているのではないかと想像される。組合せ的性質が強くなると、制約式を陽に扱う汎用ソルバーより探索に重点を置く発見的解法が優位になると思われる。逆の条件下、たとえば m/n が小さい、 α が 0 または 1 に近い、制約式係数の有効桁数が大きく制約超平面を区別しやすい、などの場合には、汎用ソルバー的解法が優位に立つのではないかと思われる。

6. おわりに

本論文では多次元ナップサック問題に対して、

- (1) 順列空間の局所探索
- (2) 連続緩和問題の解情報を利用した初期順列の生成
- (3) 多点局所探索を基礎とする RSB 戦略の適用
- (4) 部分列挙的緩和情報解の多出發初期点への応用

からなる新たな発見的解法を提案した。提案法は確率的動作を含まず、少数のパラメータで探索過程を確定的に制御するので扱いやすい。

OR-Library 中の 100 変数 30 制約と 250 変数 30 制約の問題各 30 問、および HCES のウェブサイトで公開されている 8 問を解いたところ、提案法は既存の有力な解法に十分匹敵する良好な性能を示した。特に HCES の問題では、8 問中 5 問（500 変数の問題 2 問を含む）で既存の最良値を更新している。組合せ的性質の強い問題例に対して、提案法は特に有効であるように思われる。

実験の過程で、MKP の問題例の性質に関するいくつかの仮説を得た。

- 問題例の難しさ（規模）は、 n だけでなく m にも強く影響される。
- 変数を固定した後の縮小問題は、同程度の変数（同じ制約数）の “難しい” 問題例よりさらに難しい。
- 「 m/n , α , 係数値の有効桁数」等によって、問題例の「組合せ的性質」は変化する。
- 提案法は、組合せ的性質の強い問題例に対して、制約を陽に扱う汎用解法より優位である。

これらの性質の解明・検証は、問題例固有情報の更なる抽出・利用とともに今後の課題である。また、500 変数以上の問題での実験は 2 例に留まっているが、この程度の問題例に対する提案法の所要時間は発見的解法としてはやや大きい。データ構造や記述言語の見直しを含めて実装上の改善も今後の課題である。

謝 辞

有益なコメント、ご指摘を下された査読者の方々に深く感謝いたします。

¹問題縮小を含む解法を提案した Vasquez ら [18] も同様のことを指摘している。

²この場合の計算時間は 108 秒、1 回目の探索で見いだし、15 回縮小して、非固定変数の個数 24 で終了する。

参考文献

- [1] E. Balas and C. H. Martin: Pivot and complement—a heuristic for 0-1 programming; *Management Science*, Vol. 26, pp. 86–96 (1980)
- [2] J. E. Beasley: OR-Library, <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/mknapinfo.html>
- [3] M. Berkelaar: lp_solve 4.0.1.0, ftp://ftp.es.ele.tue.nl/pub/lp_solve/
- [4] P. C. Chu and J. E. Beasley: A genetic algorithm for the multidimensional knapsack problem; *Journal of Heuristics*, Vol. 4, pp. 63–86 (1998)
- [5] B. Gavish and H. Pirkul: Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality; *Mathematical Programming*, Vol. 31, pp. 78–105 (1985)
- [6] S. Hanafi and A. Freville: An efficient tabu search approach for the 0-1 multidimensional knapsack problem; *European Journal of Operational Research*, Vol. 106, pp. 659–675 (1998)
- [7] HCES: Benchmarks for “The Multiple Knapsack Problems”, <http://hces.bus.olemiss.edu/tools.html>
- [8] ILOG, Inc.: CPLEX, <http://www.ilog.com/products/cplex/>
- [9] 沼田, 児玉: 解の淘汰により変数の固定を行う探索法について; システム制御情報学会論文誌, Vol. 12, No. 6, pp. 375–377 (1999)
- [10] 沼田, 岩倉: 解の部分固定により探索空間を縮小するメタ戦略の検討—巡回セールスマン問題への適用; システム制御情報学会論文誌, Vol. 17, No. 3, pp. 103–112 (2004)
- [11] 沼田, 小沢: 解の順列表現と探索空間縮小戦略による多次元ナップサック問題の発見的解法; システム制御情報学会論文誌, Vol. 17, No. 5, pp. 32–34 (2004)
- [12] K. Numata and K. Ozawa: On packing-order local search for the multidimensional knapsack problem; *Proceedings of 2004 Japan-USA Symposium on Flexible Automation*, C-1-2 (2004)
- [13] M. A. Osorio, F. Glover and P. Hammer: Cutting and surrogate constraint analysis for improved multidimensional knapsack solutions, Technical Report HCES-08-00, Hearin Center for Enterprise Science, The University of Mississippi, August (2000)
- [14] H. Pirkul: A heuristic solution procedure for the multiconstraint zero-one knapsack problem; *Naval Research Logistics*, Vol. 34, pp. 161–172 (1987)
- [15] M. Sakawa and K. Kato: Genetic algorithms with double strings for 0-1 programming problems; *European Journal of Operational Research*, Vol. 144, pp. 581–597 (2003)
- [16] S. Senju and Y. Toyoda: An approach to linear programming with 0-1 variables; *Management Science*, Vol. 15, pp. 196–207 (1968)
- [17] W. Shih: A branch and bound method for the multiconstraint zero-one knapsack problem; *Journal of the Operations Research Society*, Vol. 30, pp. 369–378 (1979)
- [18] M. Vasquez and Y. Vimont: Improved results on the 0-1 multidimensional knapsack problem; *European Journal of Operational Research*, Vol. 165, pp. 70–81 (2005)
- [19] 柳浦, 茨木: 組合せ最適化—メタ戦略を中心として—, 朝倉書店 (2001)