5A-04

ZDDを用いた0-1多目的ナップサック問題のパレート解列挙

鈴木　浩史†　　湊　真一†

†北海道大学　大学院情報科学研究科

1　はじめに

多目的組合せ問題では，複数の目的関数を同時に良くするような解を要求される．そのため，パレート解と呼ばれるある種の最適性を持つ解を探索することになる．一般にパレート解は複数存在し，それらを列挙することは応用上有効である．

多目的組合せ問題の一つとして0-1多目的ナップサック問題が知られている．n 個のアイテムと価値の種類数 m および容量 C が与えられ，i 番目のアイテムは非負整数の重さ w_i と非負整数の価値ベクトル $\boldsymbol{v}_i = (v_i^1, \ldots, v_i^m)$ を持つ．解はアイテムの部分集合 $x \subseteq \{1, \ldots, n\}$ で表され，$\sum_{i \in x} w_i \leq C$ を満たすならば実行可能解である．このとき，実行可能解 x に対するベクトル関数

$$\boldsymbol{v}(x) = \sum_{i \in x} \boldsymbol{v}_i = \left(\sum_{i \in x} v_i^1, \ldots, \sum_{i \in x} v_i^m \right)$$

を最適化する．具体的には，m 種類の各価値総和をそれぞれ大きくしたい．実行可能解集合 X 上の関係

$$\Delta = \{(x, x') \mid \boldsymbol{v}(x) \leq \boldsymbol{v}(x') \; and \; \boldsymbol{v}(x) \neq \boldsymbol{v}(x')\} \subseteq X \times X$$

は支配関係と呼ばれる．特に，$x \Delta x'$ であるとき，x は x' に支配される (x' は x を支配する) と言う．実行可能解 p が $p \Delta x$ であるような $x \in X$ を持たないとき，p はパレート解と呼ばれる．0-1多目的ナップサック問題に対して，動的計画法によりパレート解 (厳密にはその値) を列挙する手法を Bazgan らが提案している [1].

一方で，組合せ問題へのアプローチに有効なデータ構造として Zero-suppressed Binary Decision Diagram (ZDD) [2] が知られている．ZDD を用いた組合せ問題の実行可能解の列挙索引化手法は多岐にわたる [3]．本稿では，実行可能解を保持する ZDD を補助データ構造として用いることで，Bazgan らの手法を高速化し，かつ列挙されたパレート解を索引化する手法を提案する．

2　従来手法

Bazgan らの手法は，動的計画法に基づきアイテムを $1, \ldots, n$ の順に取捨選択する探索と，支配関係を用いた枝刈りにより構成される．アイテム i まで取捨選択を終えた実行可能解の集合を X_i，解 x の重み総和を $w(x) = \sum_{j \in x} w_j$ とする．このとき，二つの解 $x, x' \in X_i$ について，$\boldsymbol{v}(x) = \boldsymbol{v}(x')$ かつ $w(x) = w(x')$ であるとき，x と x' は等価であると見做し，X_i からいずれか一方を削除するというルールを設ける．

はじめに $X_0 = \{\phi\}$ を生成する．アイテム i の取捨選択における動作は以下の通りである．まず，アイテム $i-1$ までの取捨選択を終えたすべての解 $x \in X_{i-1}$ に対して，アイテム i を取る場合の解 $x_1 \leftarrow x \cup \{i\}$ と，アイテム i を捨てる場合の解 $x_0 \leftarrow x$ を生成する．その後，$w(x_k) \leq C$ ならば $X_i \leftarrow X_i \cup \{x_k\}$ とする $(k \in \{0, 1\})$.

X_i の生成を終えたら，X_i からパレート解に到達し得ない解を以下のルールで枝刈りする．

1. $\forall x \in X_{i-1}$ について，$w(x) + \sum_{j=i}^{n} w_j \leq C$ ならば，$X_i \leftarrow X_i \setminus \{x_0\}$ とする．
2. 二つの解 $x, x' \in X_i$ について，$x \Delta x'$ かつ $w(x) \geq w(x')$ ならば，$X_i \leftarrow X_i \setminus \{x\}$ とする．
3. 解 $x \in X_i$ から $i+1$ 番め以降のアイテムを取捨選択して得られる実行可能解集合を $X[x]$ とする．二つの解 $x, x' \in X_i$ について，$\forall a \in X[x], \boldsymbol{v}(a) \leq \boldsymbol{u}(x)$ なる m 次元整数ベクトル $\boldsymbol{u}(x)$ と，$gx \in X[x']$ をそれぞれ一つずつ定める．このとき，$\boldsymbol{u}(x) \leq \boldsymbol{v}(gx)$ ならば，$X_i \leftarrow X_i \setminus \{x\}$ とする．

枝刈り 3 において，$\boldsymbol{u}(x)$ は上界であり，Bazgan らは各価値 $1, \ldots, m$ ごとに独立して緩和問題を解くことで $\boldsymbol{u}(x)$ を求めている．また，gx として x' から得られる貪欲解を用いており，これは残りのアイテムを順に容量の許す限り取ることで得られる．これらの枝刈りによって，各ステップでパレート解に到達し得る解を刈ることはなく，X_n がパレート解の値のみを網羅する解集合になる (証明は文献 [1] を参照されたい).

枝刈りの操作は計算時間の支配的な部分であり，Bazgan らは枝刈りを高速に適用するアルゴリズムも提案している．また，アイテムの順序付けヒューリスティクスも与えており，アルゴリズム全体の高速化や良質な貪欲解の生成ができている．

3　提案手法

提案手法では，ZDD を用いて Bazgan らの動的計画法を補助する．ZDD は組合せ集合を表現する場合分け二分木を圧縮した構造を持つ非巡回有向グラフである．根と呼ばれる入次数 0 の節点が 1 つと，終端と呼ばれる出次数 0 の節点が 2 つ $(\top, \bot)$ あり，内部節点はすべて出次数が 2 である．内部節点にはアイテムのラベルが張られており，2 つの出枝はそれぞれアイテムを取る/捨てるという選択を表している．根から終端 $\top$ にた

Enumerating Pareto-optimal Solutions of the 0-1 Multi-objective Knapsack Problem using ZDD

†Hirofumi Suzui　†Shin-ichi Minato

†Graduate School of Information Science and Technology, Hokkaido University

どり着く経路は，ZDD が保持する組合せの 1 つを表す．(図 1) 0-1 多目的ナップサック問題に対して，実行可能解集合を保持する ZDD は容易に生成できる [3].

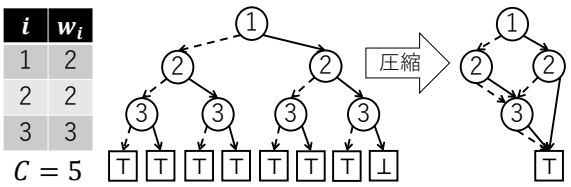

図 1: 実行可能解を保持する ZDD の例 (実線は取る選択，破線は捨てる選択に対応する．)

今，与えられたインスタンスの実行可能解集合を保持する ZDD を Z とする．Bazgan らの手法において，初期解 ϕ に Z の根を対応付け，アイテムを取捨選択する際に対応する出枝をたどり，Z 上の節点を遷移することを考える．ここで ZDD の形から，同じ節点に対応付く解すべては，残りのアイテムから取捨選択する手順がまったく同じになる．すると，以下の枝刈りを考えることができる．

- 2 つの解 $x, x' \in X_i$ について，$x \Delta x'$ かつ対応付く節点が同じならば，$X_i \leftarrow X_i \setminus \{x\}$ とする．

これは，ある手順で x から得られる解が，同様の手順で x' から得た解に必ず支配されるため可能である．

また，ある 1 つの価値に着目した場合，Z 上の各節点からその先で得られる最大の価値総和は，Z の節点数に比例する時間で求まる．具体的には，Z 上の各枝について，出本の節点に対応するアイテムの価値または 0 を重み付け，$\top$ への最長経路問題を解けば良い．これには，ZDD が非巡回有向グラフであることから，動的計画法を用いた解法が存在する．よって，予め各節点について，各価値のその先での最大総和を計算しておくことで，Bazgan らの手法における枝刈り 3 で，$\boldsymbol{u}(x)$ として厳密な上界を高速に求めることができる．

以上から，提案手法によって枝刈りの質が高まり，アルゴリズムの高速化が見込まれる．さらに，探索の遷移を模倣した ZDD はパレート解集合を保持する ZDD となる．解集合の ZDD 構築については別途論ずることとし，本稿では高速化の効果について評価を行う．

4 計算機実験

$m = 3, 4$ の場合について，Bazgan らの手法と提案手法で実行時間を比較した ($m = 2$ の場合については，計算時間に大きな違いが見られなかった)．Bazgan らの手法は 0-1 多目的ナップサック問題のパレート解をすべて求めることについて，$m \geq 3$ の場合に対する初めてのアルゴリズムであると文献 [1] に述べられているが，$m = 3$ の場合までしか実験が行われていない．よって，本実験により $m = 4$ の場合に対する初めての結果が示される．

各インスタンスは，m 個の価値がランダムに決まる ($1 \leq v_i^j \leq 1000$) タイプ 1，または m 個の価値に相関がある ($900 \leq v_i^1 + \cdots + v_i^m \leq 1100$) タイプ 2 に属す．アイテムの重さは一様ランダム ($1 \leq w_i \leq 1000$) であり，容量は $C = \sum_{i=1}^{n} w_i/2$ とした．実験環境に 64-bit Ubuntu 16.04 LTS，Intel Core i7-3930K 3.2GHz CPU，64GB RAM を用い，各インスタンスタイプで n を変化させ，それぞれ 10 ケースの平均計算時間を測定した．n は小規模なものにとどめている．

結果を図 2 と図 3 に示す．いずれのインスタンスタイプでも，提案手法により 2 から 3 倍程度の高速化を達成した．また，$m = 3$ の場合よりも $m = 4$ の場合の方が，高速化にやや大きく貢献しており興味深い．

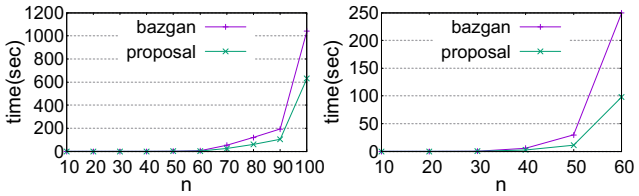

図 2: タイプ 1 の計算時間 (左：$m = 3$，右：$m = 4$)

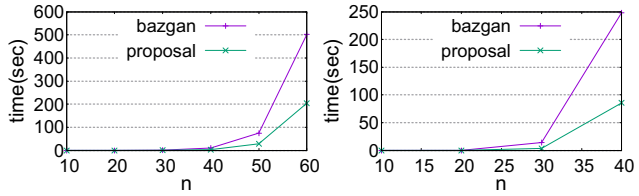

図 3: タイプ 2 の計算時間 (左：$m = 3$，右：$m = 4$)

5 まとめ

本稿では，0-1 多目的ナップサック問題に対して，ZDD を補助データ構造として用いることで既存の動的計画法を高速化する手法を提案した．小規模ながらも文献 [1] で扱われた問題よりも目的数の多い問題でも実験を行い，提案手法の有効性を確認した．

今後は，本結果を得た要因を詳細に調べ，より一層の高速化を目標とし，中規模の問題へもアプローチ可能なアイデアを得たい．また，他の問題への拡張性についても検討している．

謝辞

本研究の一部は科研費基盤 (S) 15H05711 の助成による．

参考文献

[1] Cristina Bazgan, Hadrien Hugot, and Daniel Vanderpooten. Solving efficiently the 0-1 multi-objective knapsack problem. *Computers & OR*, Vol. 36, No. 1, pp. 260–279, 2009.

[2] Shin-ichi Minato. Zero-suppressed BDDs for set manipulation in combinatorial problems. *Proc. of 30th ACM/IEEE Design Automation Conf. (DAC 1993)*, pp. 272–277, 1993.

[3] 川原純, 湊真一. 組合せ問題の解を列挙索引化する ZDD 構築アルゴリズムの汎用化. 電子情報通信学会技術研究報告. COMP, コンピュテーション, Vol. 112, No. 93, pp. 1–7, jun 2012.