1999年度日本オペレーションズ・リサーチ学会
春季研究発表会

順序制約付きナップサック問題の近似解法と厳密解法*

02991720　防衛大学校　ナタウット　サムパイブーン†　SAMPHAIBOON Natthawut
01700900　防衛大学校　山田武夫　YAMADA Takeo

1　はじめに

ある商品を採択するには, それに先行するすべての商品が採択されていなければならないという先行順序制約を持つn個の商品 $1, 2, \cdots, n$ があり, それらの重量と利得をそれぞれ w_i, $p_i (i = 1, 2, \cdots, n)$ とする. これらを容量Bのナップサックに詰め込み, 順序制約を満たしながら総利得を最大とする問題を, **順序制約付きナップサック問題** (PCKP) と呼ぶ. これについては**動的計画法** (DP) による解法を提案した[1]が, DP解法では大きなサイズの問題を解くことは困難であった. これに対して, 本稿では比較的精度の良い近似解法を高速に求めることのできる**グリーディ解法**と, この結果を利用して問題のサイズを大幅に削減する**問題縮小法**を検討する.

2　問題の定式化

PCKP は有向グラフを用いて, 次のように表現できる. まず, 商品に対応して節点集合 $V = \{1, 2, \cdots, n\}$ をとる. 次に, 商品iが商品jに先行するとき, 節点iから節点jへ有向枝(i, j)を引き, このような枝全体の集合を$E \subseteq V \times V$ と記す. 順序関係の定義から, 有向グラフ $G = (V, E)$ にはサイクルは含まれないものとする. すなわち, G は**無閉路有向グラフ** (DAG) であり, その節点はトポロジカルに番号付けられていると仮定してよい.

図 1 はこのようなDAGの一例である. Gにおいて, 節点iから節点jへの有向道があるとき, iはjの**先行節点**, jはiの**続行節点**であるといい, $i \to j$と記す.

ここで,

$$x_i = \begin{cases} 1, & \text{節点} i \text{を採択するとき,} \\ 0, & \text{そうでないとき.} \end{cases}$$

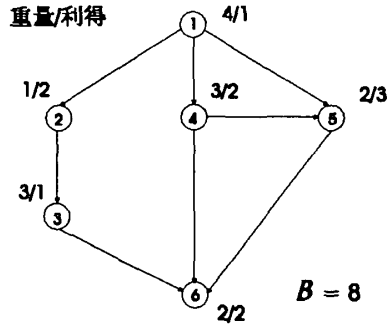

図 1: 無閉路有向グラフ上のPCKP

とすると, この問題は整数計画問題として以下のように定式化される.

PCKP(G):

$$\text{Max.} \quad \sum_{i \in V} p_i x_i \tag{1}$$

$$\text{s. t.} \quad \sum_{i \in V} w_i x_i \le B \tag{2}$$

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

$$x_i \in \{0, 1\}, \quad \forall i \tag{4}$$

3　グリーディ解法

一般性を失うことなく, 以下では節点1(n)がGの唯一の極大(小)節点であると仮定し,

$$w_1 \le B, \textstyle\sum_{i=1}^{n} w_i > B$$

とする. また, 節点iの先(続)行節点の集合を$\partial_i^+ (\partial_i^-)$と記す. すなわち,

$$\partial_i^+ := \{j \in V \mid (j, i) \in E\},$$
$$\partial_i^- := \{j \in V \mid (i, j) \in E\}.$$

以下のアルゴリズムによって得られる解ベクトルを**グリーディ解**と呼んでx_Gと記し, その解における利得をz_Gとする.

*大阪国際大学 (1999.3.23-24)
†E-mail: nata@cs.nda.ac.jp

アルゴリズム *Greedy*

Step 1 : $\boldsymbol{x} = (x_i \mid i = 1,2,\cdots,n)$ を $x_i := 0; \forall i$ とする. z_G:=0, $w_G := 0$ とおく.

Step 2 : 節点 $i \in V$ で, $x_i = 0$, $\forall j \in \partial_i^+$ について $x_j = 1$, $w_G + w_i \leq B$ を満たすものを探す. そのような i が見つかればStep 3へ, そうでなければStep4 へ進む.

Step 3 : $x_i := 1$, $z_G := z_G + p_i$, $w_G := w_G + w_i$ として, Step 2へ戻る.

Step 4 : $(\boldsymbol{x}, z_G)$ を出力して終了.

例えば, 図1でB=8として上のアルゴリズムを適用すると, グリーディ解 $\boldsymbol{x}_G = (1,1,0,1,0,0)$ と $z_G = 5$ を得る.

4 問題の縮小化

G の節点 i, $j \in V$ について, i の**上(下)流部分** $V^i(V_i)$ を

$$V^i := \{j \in V \mid j \to i\},$$
$$V_i := \{j \in V \mid i \to j\}.$$

により定義する. $\bar{V}_i := V \backslash V_i$ は, V_i の補集合で,

$$W^i := \sum_{j \in V^i} w_j,$$
$$P_i := \sum_{j \in \bar{V}^i} p_j.$$

と置くと, 次が成立する.

定理1. PCKPの最適解を $\boldsymbol{x}^* = (x_i^*)$ とすると,

(i) $W^i > B$ であれば $x_i^* = 0$,

(ii) $P_i < z_G$ であれば $x_i^* = 1$.

証明 : 明らかである. □

図1の例題については下表の結果を得る.

i	V^i	V_i	W^i	P_i
1	1	1,2,3,4,5,6	4	$0^\dagger$
2	1,2	2,3,6	5	6
3	1,2,3	3,6	8	8
4	1,4	4,5,6	7	$4^\dagger$
5	1,4,5	5,6	9^*	6
6	1,2,3,4,5,6	6	15^*	9

表中の $*$, $\dagger$ は定理1の(i)(ii)により, それぞれ0, 1に固定される部分を示す. このようにして, 固定された部分を差し引くと, 図1は図2のように縮小される.

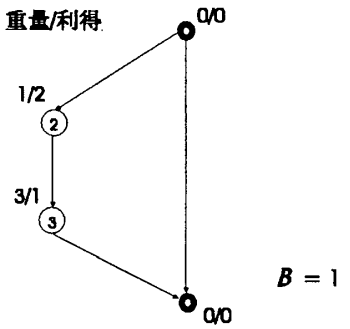

図 2: 縮小されたPCKP

5 数値例

より大型の例題として, n=200, B=1000, p=0.4 で, w_i, p_i を[1,100]間の一様乱数より発生させたものを用いたところ, z_G=894のグリーディ解を得, 200変数中14個が1に, 179個が0に固定され, 7変数のみが残った. これをDPで解いて, z^*=937を得たが, これに要した計算時間は全体で0.003秒であった. これに対して元の問題を直接DPで解くと1.58秒を要した.

6 まとめ

順序制約付きナップサック問題の近似解法と, 問題縮小法を検討し, かなり大型の問題を解けるようになった. 今後, 計算機実験をより本格的に行い, その結果と考察を発表当日に報告する予定である.

参考文献

[1] S. ナタウット, 山田武夫, "順序制約付きナップサック問題のDP解法", OR学会秋季研究発表会(日本大学会館, 1998.10.15-16).

[2] 今野浩, 鈴木久敏, 「整数計画法と組み合わせ最適化」, 日科技連(1982).

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

[4] D. S. Johnson and K. A. Niemi, On knapsacks, partitions, and a new dynamic programming technique for trees, *Mathematics of Operations Research 8(1983)*, 1-14.

[5] G. Cho and D. X. Shaw, A depth-first dynamic programming algorithm for the tree knapsack problem, *INFORMS Journal on Computing* 9(1997), 431-438.