Partitionのある風景

飯田浩志 *

概要

本稿では, Partitionからのreductionによって近似率を導出した事例を三つ, 紹介するとともに, それら事例をもとに, 若干の考察を加える.

キーワード: 組合せ最適化; 分割問題; 近似アルゴリズム; 近似率

組合せ最適化には, Korte and Vygen [7] 等, それを書名に冠する成書を見れば分かるように, 多種多様な問題がある. そんな中, 2001年, 電子商取引の一部, 組合せオークションにおける落札者決定問題で, 各品物が一つきりではなく複数ある場合(Multi-Unit Winner Determination Problem)を整数計画問題として定式化したものが, 実は, 多制約ナップサック問題(Multidimensional Knapsack Problem)と同一であることが分かった. このように, 一見して何のつながりも無いような事柄がきっかけで, 事態が進展することもある. 事の詳細については, Pfeiffer [9] を参照されたい. また, 組合せオークションの概略は, Kellerer et al [6, 第15.7節] にも記述がある.

さて, 以下に掲げる三つの問題をご存知だろうか. 実は, まったく関連が無いように見えるこれら三つの問題それぞれにPartitionが関係している. Partition(分割問題)とは, n 個の自然数 c_j が与えられて, 二分割したそれぞれの総和が等しくなるようにできるか否かを, 言い換えれば, $\sum_{j\in S} c_j = \sum_{j\notin S} c_j$ とできる部分集合 $S \subseteq \{1, 2, \ldots, n\}$ の有無を問う判定問題であり, $\mathcal{NP}$ 完全である (Karp [4]). 以下では, Partitionがこれら三つの問題に如何に関係しているかを記す.

Bin-Packing ([7, 第18章])

入力: n 個の数 $a_1, a_2, \ldots, a_n$ $(0 < a_j \leq 1)$.

出力: $\sum_{j:\, f(j)=i} a_j \leq 1$ $(1 \leq i \leq k)$ を満たす函数 $f : \{1, 2, \ldots, n\} \to \{1, 2, \ldots, k\}$ を実現可能な最小の k, i.e., a_j 全部を詰め込むのに必要な容量1の瓶の最小個数.

Dedicated (or Identical) Parallel Machine Scheduling with Resource Dependent Processing Times (Grigoriev et al [2])

入力: n 個の仕事を m 台の機械で処理する. 資源の割り当て s で仕事 j を行うと, 機械にかかわらず p_{js} 時間かかる. ただし, 同一時の資源の使用個数は k を上限とし, 仕事をする間は資源の使用個数は変えられない.†

出力: 最終完了時刻 (makespan), 即ち, 最後の仕事の完了時刻を最小にする, すべての仕事の機械

†資源を, 機械に張り付く人と考える [2]. k 人しかいないので, 同時に機械に張り付けるのは k 人までである. Grigoriev et al [2] は, 人が多く張り付くほど仕事の処理時間は短縮される, i.e., $p_{j0} \geq p_{j1} \geq \cdots \geq p_{jk}$ を仮定している. 他方, Kellerer [5] では, この仮定は無い. 余談になるけれども, この問題への近似アルゴリズムに関しては今の所, Kellerer [5] が提案した $(3.5 + \epsilon)$-近似アルゴリズム (ϵ は任意の正の数) が最良であり, 近似率の限界として示されている1.5 (これについては後述) との間には, まだ大きなギャップがある. その一方で, Bin-Packingには, First-Fit Decreasing (FFD: n 個の数を非昇順に並べた後, この順番で各数を, 番号の若い瓶から見ていって最初に入れられる (First-Fit) 瓶に詰める) および Best-Fit Decreasing(BFD: 先のように番号の若い瓶から見ていくのではなくて, それが入れられる瓶の中で最も内容量の多い瓶 (該当する瓶が複数あれば, 番号が一番若い瓶) に詰める) なる1.5-近似アルゴリズムがある (Simchi-Levi [10]). 実際に, FFDやBFDが近似率1.5を与える例としては, $\{0.4, 0.4, 0.3, 0.3, 0.3, 0.3\}$ を考えてみるとよい [7, p.454]. 開いている瓶が一つのとき, その瓶に入るなら入れてしまうところに, ジレンマがある. Bin-Packingへの近似アルゴリズムについては, Coffman et al [1] に詳しいので, そちらも参照されたい.

*E-mail: Oggi@res.otaru-uc.ac.jp

への割り振りと各仕事の開始時刻, ならびにそれぞれの仕事の実行時の資源割り当て数.

K-th Heaviest Subset

入力: 整数 $c_1, c_2, \ldots, c_n, K, L$.

出力: $\{1, 2, \ldots, n\}$ の K 個の互いに排他的な部分集合族 $S_1, S_2, \ldots, S_K$ で, $\sum_{j \in S_i} c_j \geq L$ $(1 \leq i \leq K)$ を満たすものがあるか?

$\mathcal{P} \neq \mathcal{NP}$ を仮定する限り, 瓶詰め (Bin-Packing) 問題ならびに処理時間が割り当てられた資源の数に依存するとした複数同一マシンのスケジューリング (Dedicated Parallel Machine Scheduling with Resource Dependent Processing Times) 問題のいずれも近似率 1.5 未満を達成し得ないことが知られている.‡ここに, どちらの証明においても, Partition から当該問題への reduction が構成されている. 一つ, この二つの問題は, 両方とも最小化問題であることに注意されたい.

具体的には, $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$ となる部分集合 S がある場合を最適値 2, そうでない場合を最適値 3 以上に対応させるように, Partition からの reduction を構成している. これにより, 1.5-近似より良い — 最悪の場合でも最適値の 1.5 倍未満の値を保証する — アルゴリズムが, Partition を解く (求める部分集合 S の有無を判定する) 多項式時間アルゴリズムとなって矛盾する. 詳細は, それぞれの問題を定義する際に掲げた文献 [7, 定理 18.1., 系 18.2.], [2, 定理 3.] を参照されたい.

加えて, K-th Heaviest Subset も, Partition からの reduction を構成できる ([7, p.390, 問 18]). 何となれば, n および各 c_j はそのままに, $K := 2$, $L := \lceil \sum_j c_j/2 \rceil$ とすればよい. $c(S) := \sum_{j \in S} c_j$ として, $c(S_1) \geq L$, $c(S_2) \geq L$ ならば,

$$2L \geq \sum_{j=1}^{n} c_j \geq c(S_1) + c(S_2) \geq 2L$$

なので (最初の不等号は天井 (ceiling) 函数の性質から, 二番目の不等号は S_1 と S_2 の排他性から), $\sum_j c_j = c(S_1) + c(S_2)$ を得る. 結局, S_1(もしくは S_2) が, 求める部分集合 S となる.

前二つの問題と同様, 近似率について考察するために, 判定問題である K-th Heaviest Subset に関連する以下の最適化問題を考える:

$$\begin{array}{rl} \text{最大化} & K, \\ \text{制約条件} & \sum_{j \in S_i} c_j \geq L,\ 1 \leq i \leq K, \\ & S_i \cap S_j = \emptyset,\ 1 \leq i < j \leq K, \\ & S_i \subseteq \{1, 2, \ldots, n\},\ 1 \leq i \leq K. \end{array} \tag{1}$$

ここで, 前と同じく n および各 c_j はそのままに, $L := \lceil \sum_j c_j/2 \rceil$ として, Partition からの reduction を構成する. (1) の最適値が 2 の場合は Partition において求める部分集合 S が存在し, 1 の場合は存在しない. この時, (1) に 0.5-近似§より良い — 最悪の場合でも最適値の半分を超える値を保証する — アルゴリズムがあるとすれば, そのアルゴリズムは, 最適値が 2 の時は $2\ (> 2 \times 1/2)$ を, 最適値が 1 の時は 1 を返すことになる. つまり, Partition を解く多項式時間アルゴリズムとなって $\mathcal{P} \neq \mathcal{NP}$ に矛盾する. したがって, (1) の近似率は 0.5 より大きくなり得ない, と結論できよう.

ここまで三つの共通点をまとめると, Partition において求める部分集合 S がある場合を最適値 2 になるように, そうでない場合を 2 より悪い最適値を持つように, Partition からの reduction を構成することができれば, 最小化問題であれば近似率 1.5 が, 最大化問題であれば近似率 0.5 が, それぞれ限界であることを示せる, ということになる. 但し, 以上の議論は, Partition からの reduction によって得られる最

‡既に第四版を重ねる [7] では, 近似アルゴリズムの性能を示す performance ratio と, 問題自体が持つ近似の困難さ (いかなる近似アルゴリズムを持ってしても超えられない performance ratio の限界) を示す approximation ratio は区別されるが, 本稿では, どちらも近似率と呼んでいる. これも余談になるけれども, FFD と BFD は, 先述した近似率のみならず, 漸近的近似率も 11/9 で等しい (Johnson et al [3]). FFD と BFD の類似性については, Li and Chen [8] の得た結果も興味深い.

§最大化問題に対する近似率を定義するにあたっては, 逆数を取って 2-近似とする流儀もある (例えば [7, p.393]). 逆数を取らない方では, 最大化問題か最小化問題かが近似率からすぐに見て取れる.

適化問題の目的函数の値域が整数でなければ意味がない.

また, Karp [4] は, Partition から Max-Cut への reduction を構成している. Max-Cut は, 次のように定義される:

Max-Cut

入力: 無向グラフ $G = (V, E)$, 枝に正の整数の重みを付する函数 $w : E \to \mathbb{N}$, 正の整数 W.

出力: $\sum_{e \in \{(u,v) \in E \mid u \in S, v \notin S\}} w(e) \geq W$ を満足する, 頂点の部分集合 $S \subseteq V$ があるか?

残念ながら, W の最大化をその目的とする最適化版の Max-Cut には 0.878-近似アルゴリズムが既に提案されており (例えば [7, 第 16.2 節] を見よ), ここでの議論は当て嵌まらないことになる.

しかしながら, ここまでの議論を振り返れば, Partition からの reduction によって得られる最適化問題で, Partition において求める部分集合 S がある場合を最適値 2 に限定する必要はないことが分かる. つまり, 最適値が $k\,(\geq 2)$ とすれば, 最小化問題であれば近似率 $(k+1)/k$ が, 最大化問題であれば近似率 $(k-1)/k$ が限界である事を導ける. ただ, Max-Cut については, Karp [4] の構成では, Partition における一つの数 c_j が Max-Cut の一頂点 j に対応し, 枝 (i, j) の重みが $c_i \cdot c_j$ で与えられることから, Partition において求める部分集合 S がある場合に対応する最適値をあらゆる具体例について, ある値に固定するには無理がある. 換言すれば, いくらでも大きくできる. 例えば, Partition の例 $\{M, M-1, 1\}$ から構成される最適化版の Max-Cut を考えると (図 1), 重み $M(M-1)$ と M の枝二本を切るのが最適 (もとの Partition においても $M = (M-1)+1$ となる) であり, このとき, 最適値は M^2 である.¶

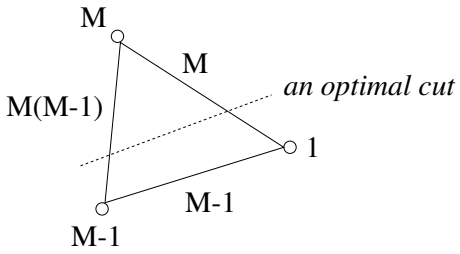

図 1: Partition $\{M, M-1, 1\}$ から構成された Max-Cut

以上, Partition を基軸とした話題を提供した. 序文で述べたように, 思いがけないところに問題解決の糸口が見つかることもある. 本稿が, 何かを生み出す一助となれば幸いである.

REFERENCES

[1] E. G. Coffman Jr., G. Galambos, S. Martello and D. Vigo, Bin packing approximation algorithms: Combinatorial analysis. In: D.-Z. Du and P. M. Pardalos (eds.) Handbook of Combinatorial Optimization — Supplement Volume A, pp. 151–207, Dordrecht: Kluwer 1999.

[2] A. Grigoriev, M. Sviridenko and M. Uetz, Unrelated parallel machine scheduling with resource dependent processing times. In: M. Jünger and V. Kaibel (eds.) Proceedings of the 11th International IPCO Conference held at Berlin in 8–10 June 2005, LNCS 3509, pp. 182–195, Berlin: Springer 2005.

[3] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM J Comput **3**(4) 299–325 (1974).

¶Karp [4] の構成では, $W := \lceil (\sum_j c_j)^2/4 \rceil$ である. もし $\exists S \subseteq \{1, 2, \ldots, n\}$ s.t. $\sum_{j \in S} c_j = \sum_{j \notin S} c_j$ ならば, $W = \lceil (2\sum_{j \in S} c_j)^2/4 \rceil = (\sum_{j \in S} c_j)^2$. S に含まれる頂点と $V \setminus S$ に含まれる頂点をつなぐ枝の重みの総和は $\sum_{j \in S}(c_j \sum_{j \notin S} c_j) = (\sum_{j \notin S} c_j)\sum_{j \in S} c_j$ であり, 今, これは W と一致する. これ以外の場合では, 必ず $(\sum_{j \in S} c_j)(\sum_{j \notin S} c_j) < W$ である. 実際, $\sum_j c_j = 2k+1$ の時, $W = \lceil (2k+1)^2/4 \rceil = k^2 + k + 1$. ここで, カットされる枝の重みの総和は, $\forall l \geq 0$, $(k-l)(k+l+1) = k^2 + k - l(l+1) \leq k^2 + k < W$. 加えて, $\sum_j c_j = 2k$ かつ二分割したそれぞれの総和が異なる時, $\forall l \geq 1$, $(k-l)(k+l) = k^2 - l^2 < k^2 = \lceil (2k)^2/4 \rceil = W$.

[4] R. M. Karp, Reducibility among combinatorial problems. In: R. E. Miller and J. W. Thatcher (eds.) Complexity of Computer Computations, pp. 85–103, New York: Plenum Press 1972.

[5] H. Kellerer, An approximation algorithm for identical parallel machine scheduling with resource dependent processing times. Oper Res Lett **36**(2) 157–159 (March 2008).

[6] H. Kellerer, U. Pferschy and D. Pisinger, Knapsack Problems, Berlin: Springer 2004.

[7] B. Korte and J. Vygen, Combinatorial Optimization — Theory and Algorithms (Algorithms and Combinatorics **21**), 4th edition, Berlin: Springer 2008.

[8] C.-L. Li and Z.-L. Chen, Bin-packing problem with concave costs of bin utilization. Naval Res Logist **53**(4) 298–308 (June 2006).

[9] J. Pfeiffer, Combinatorial Auctions and Knapsack Problems — An Analysis of Optimization Methods, Saarbrücken, Germany: VDM Verlag Dr. Müller 2007.

[10] D. Simchi-Levi, New worst-case results for the bin-packing problem. Naval Res Logist **41**(4) 579–585 (June 1994).