5K-04

ZDD の反復的トップダウン構築による 選挙区割問題に対する厳密解列挙高速化

山崎宏紀 † 川原純 ‡ 湊真一 §
京都大学 † 京都大学 ‡ 京都大学 §

1 導入

グラフ分割問題はグラフ理論における基本的な問題であり、選挙区割問題もグラフ分割問題の重要な応用例である。選挙区内に所属する人口によって、一人の代表者が当選するために必要な得票数が異なる場合がある。この時、一人当たりの当選に必要な得票数について最も少ない選挙区と最も多い選挙区との比を指して格差と呼ぶ。選挙区割における格差は各区割内の人口の下限や上限に対する制約としても表現することができる [1]。選挙区割問題では対象地域の情報を表すグラフと格差の上限を表す値が与えられ、格差が指定された値以下となる区割を見つける、もしくは列挙することが目的となる。

ゼロサプレス型二分決定グラフ (ZDD) [2] を用いて格差が指定した値以下になる全ての分割を圧縮して表す既存手法 [3] が存在するが、構築過程で ZDD ノード数が指数的に増加するため、一部インスタンスでメモリ不足により解が得られないという問題がある。本稿では、サブセッティング法 (SS 法) [4] を反復的に適用し、選挙区割問題に対する厳密解を包含する近似解集合を表す ZDD を経由して厳密解集合を表す ZDD を構築する手法を提案する。計算機実験によって厳密解手法の性能について評価を行い近似解集合を経由しない場合 [3] と比較して実行時間、消費メモリ量共に大幅に改善されることを確認する。

2 準備

選挙区割問題を重み制約付きグラフ分割列挙問題として表現する。入力はグラフ $G = (V, E, w)$、分割制約 k、そして重み制約 $\mathcal{C}$ である。出力は G の分割 $\mathcal{S}$ であって、連結成分個数がちょうど k 個であるものの内、重み制約 $\mathcal{C}$ を満たすもの (以下、k-グラフ分割 と呼ぶ) の集合である。重み制約について、本稿では特に重み下限制約のみを扱う。重み下限制約は正整数パラメータ L で与えられる。「連結成分 $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ に対し、$\min_{C \in \mathcal{S}} \sum_{v \in C} w(v) \geq L$ を満たすグラフの集合」という制約を「重み下限制約 L」あるいは単に 「下限制約」 と呼ぶ。

本稿ではグラフの集合をゼロサプレス型二分決定グラフ (Zero-suppressed Binary Decision Diagram, ZDD) [2] を用いて表現する。ZDD とは、集合族を表現するために用いられる有向非巡回グラフを利用したデータ構造である。ZDD を用いることにより多くの対象に対し、ときには指数的な数の集合族を効率的に保持・表現することができる。

複数の (独立な) 制約を満たす集合族を表す ZDD を構築する場合、各制約を満たす集合族を表す ZDD を構築した後、順に、ZDD の再帰的構造を利用した集合演算を適用することで全ての制約を満たす ZDD を得ることができる。この方法を用いる場合、最終的に得られる ZDD が小さくなる場合でも、過程の各制約を満たす ZDD は大きくなり得る。この場合、各 ZDD を構築する過程で時間がかかったり、メモリ不足により解が得られない可能性がある。

再帰的構造による集合演算に対し、ZDD を根ノードに近いノードから順に構築する方法をトップダウン構築と呼ぶ。トップダウン構築方法の一つにサブセッティング法 (SS 法) [4] が知られている。SS 法では集合族 F を表す構築済みの ZDD Z_F と、 F と同一の台集合を持つ集合族 S を入力とし、F と S の共通集合を表す ZDD $Z_{F \cap S}$ を構築する。区割列挙問題において、F は k-グラフ分割の集合、S は下限制約を満たすグラフ分割の集合に対応する。SS 法では S を表す ZDD のトップダウン構築処理を行いつつ、 Z_F を利用して Z_F に含まれない集合族の枝刈りを行うことで $Z_{F \cap S}$ を構築する。SS 法を用いることで、S を表す ZDD Z_S を陽に構築することなく $Z_{F \cap S}$ を構築することができる。

3 提案手法

本稿では、SS 法を反復的に適用し、下限制約付きグラフ分割問題に対する厳密解集合を高速・省メモリに構築する手法を提案する。

k-グラフ分割の集合を表す ZDD を Z_k、下限制約を満たす k-グラフ分割の集合を表す ZDD を $Z_{k,L}$ と表記する。従来手法では Z_k から $G = (V, E, w)$ を入力として一回のサブセッティングにより $Z_{k,L}$ を構築する。提案手法では G を基に重み関数のみが異なる div 個のグラフ $\mathcal{G} = \{G'_1 = (V, E, w'_1), G'_2 = (V, E, w'_2), \ldots, G'_{div} = G\}$ を構築し、$\mathcal{G}$ の要素を順に入力として SS 法を適用する。

ここで最終的に厳密解を得るための必要条件は $\forall v \in V, w(v) \leq w'_i(v)$ を満たすことである。w' を適切に構成することで実質的な頂点重みの有効桁数が削減され、w をそのまま入力とする場合に比べて SS 法による ZDD 構築過程における状態数が減少する。

有効桁数の小さな重み制約下で近似解集合を表すコン

Accelerating Enumeration of Exact Solutions for Electoral Districting Problems by Iterative Top-down ZDD Construction
† Yamazaki Hiroki, Kyoto University
‡ Kawahara Jun, Kyoto University
§ Minato Shin-ichi, Kyoto University

表 1　入力データ

Name	n	m	k	L
G_1 (hokkaido)	68	161	12	306,288
G_2 (fukushima)	55	130	5	272,187
G_3 (saitama)	72	185	15	325,515
G_4 (chiba)	60	134	12	323,471
G_5 (nagano)	77	185	5	296,023

パクトな ZDD を構築し、SS 法によるトップダウン構築の過程で大まかな解空間の枝刈りを行う。その ZDD を利用して SS 法によって徐々に、より厳密な重み制約を課すことで最終的に厳密解集合を表す ZDD を構築する。

4　計算機実験

本項では、近似手法と SS 法を用いて下限制約に対して厳密解を表す ZDD の構築を行なった結果を示す。

CPU が Intel Xeon Gold 6134 (3.20GHz)、メモリが 3.0 TB のマシンを利用して計算機実験を行った。C++17 によって実装し、gcc を用いて -O3, -march=native オプションを付加してコンパイルを行った。また、ライブラリとして SAPPOROBDD [*1]、TdZdd [*2] を利用した。

日本の都道府県ごとの地理的データを基とし、市区町村を頂点、隣接関係を辺、人口を頂点重みとした入力データを利用した。結果については、辺数の多いインスタンスを抜粋して掲載する。入力データについて、 グラフのパラメータを表 1 に示す。下限制約 L については k を分割数、定数 $r = 1.50$、重み和 $P = \sum_{v \in V(G)} w(v)$ に対し、$L = \frac{P}{r(k-1)+1}$ として定めた。

近似重み生成個数 div は生成する重み列の個数、或いは SS 法の適用回数を表す可変パラメータである。従来手法 [3] は生成個数 $div = 1$ とした場合と等価である。div を大きくするとより正確に枝刈りが行われやすくなるため、消費メモリ量は減少することが予想されるが、一方で SS 法の適用回数の増加に伴って実行時間の増加も懸念される。従って、div 変化を変化させて実行時間・消費メモリ量に対する影響を実験により確認した。

また、div を十分大きくした場合の挙動を確認するため、一例として、サブセッティング 1 回ごとに近似精度を 1bit ずつ上げるように $div = \lceil \log L \rceil$ として近似重み生成を行った場合の結果も併せて示す。実験では丸め単位を順に $\lfloor \frac{L}{2^1} \rfloor, \lfloor \frac{L}{2^2} \rfloor, \ldots, \lfloor \frac{L}{2^i} \rfloor, \ldots, 1$ として重み生成を行い、SS 法を適用した。

SS 法の各ステップにおける入力となる重み関数は元の頂点重みをソートし、事前に定めたパラメータ div 個ごとに取り出し、その値を丸め単位 $rounds$ として、各頂点重みを $rounds$ 毎に切り上げた値を近似重みとして用いる。例えば $div = 2$ の場合は w の中央値を、$div = 4$ の場合は四分位数を丸め単位として用いる。

実行時間についての結果を表 2 に示す。G_5 で生成個数

表 2　実行時間 (秒)

div	(従来手法)	2	4	$\lfloor \log L \rfloor$
G_1	9,447	1,073	**593**	2,648
G_2	22,180	2,683	**1,219**	1,380
G_3	221	95	**92**	252
G_4	149	81	**72**	303
G_5	718	47	**32**	82

表 3　消費メモリ量 (MB)

div	(従来手法)	2	4	$\lfloor \log L \rfloor$
G_1	1,736,730	219,258	100,779	**34,522**
G_2	1,032,000	366,785	159,905	**21,774**
G_3	21,936	6,513	3,529	**2,848**
G_4	34,618	15,346	8,519	**3,480**
G_5	171,250	6,330	3,711	**1,663**

$div = 4$ とした場合に 22.4 倍高速化されている。他のインスタンスについても従来手法より 2.1 倍以上高速化されている。実行時間の観点では全てのインスタンスにおいて $div = 4$ で最も効率的となる。

消費メモリについての結果を表 3 に示す。G_5 で 最大 103.0 倍省メモリとなっている。他のいずれのインスタンスについても従来手法より 7.7 倍以上改善されている。消費メモリ量の観点では全てのインスタンスで $div = \lfloor \log L \rfloor$ の場合で最小となる。

G_5 は重みが 10,000 以下の比較的小さな頂点が多数存在するため、丸めの結果、w の値の種類数が大きく減少する (順に、7 種類、13 種類、20 種類)。これによってノード同士の共有が行われやすくなり、消費メモリ量が大きく減少したと考えられる。

5　結論

本稿では下限制約を持つグラフ分割問題に対し、反復的にサブセッティング法を用いることで厳密解集合を表す ZDD を効率的に構築する手法を提案した。計算機実験を行い、従来手法に比べて最大 21.0 倍以上高速かつ 51.0 倍以上省メモリに厳密解集合を表す ZDD を構築できることを確認した。

謝辞

本研究の一部は, JSPS 科研費 JP19K12098, JP20H00605, JP20H05964 の助成を受けたものです。実験データを提供していただいた堀田敬介教授 (文京大) に感謝いたします。

参考文献

[1] 根本俊男, 堀田敬介: 衆議院小選挙区制における一票の重みの格差の限界とその考察, 選挙研究, No. 20, pp. 136–147,226 (2005).
[2] Minato, S.: Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems, DAC '93, pp. 272–277 (1993).
[3] Kawahara, J. et al.: Generating All Patterns of Graph Partitions Within a Disparity Bound, *WALCOM*, pp. 119–131 (2017).
[4] Iwashita, H. and Minato, S.: Efficient Top-Down ZDD Construction Techniques Using Recursive Specifications, Technical report (2013).

[*1] https://github.com/Shin-ichi-Minato/SAPPOROBDD
[*2] https://github.com/kunisura/TdZdd