A-016

選挙区割問題に対する ZDD を用いた近似的列挙手法の提案と評価

山崎 宏紀[1)]　川原 純[1)]　湊 真一[1)]

1　はじめに

グラフの分割問題は VLSI 回路設計、マルチプロセッサのタスクスケジューリング、避難計画割り当て等様々な応用先が存在する [1]。

選挙区割問題も重要な応用例である。各選挙区によって一人の代表者が当選するために必要な得票数は異なる場合が多い。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を基準とし最も多い選挙区がその何倍であるか、を表すために格差という言葉が用いられる。

格差が最小となるような区割を整数計画法により求める手法が知られている [2]。一方で、選挙区割問題においては格差を最小化するだけでなく、地理的、法的制約を満たすことも求められる。このような制約は整数計画法などの制約式として表すことが困難である。これに対する一つの解決法として、格差が比較的小さな区割を事前に列挙し、制約を満たすような区割を人間が選択する方法が考えられる。既存手法として、ゼロサプレス型二分決定グラフ (Zero-suppressed Binary Decision Diagram、ZDD) [3] 及びフロンティア法 [4] を用いて格差が指定した値以下になるような区割を全て列挙する手法が提案されている。しかし、既存手法では一部のインスタンスでメモリ不足により解が得られていない。本研究では組合せ数が小さくなることが期待できるように入力に近似を行い、ZDD のノード数削減を図る手法を提案する。

本研究ではまず ZDD 及び既存手法であるフロンティア法を用いた選挙区割列挙アルゴリズムについて述べる。その後近似手法を提案し、都道府県データを用いて評価を行う。

2　Zero-Suppressed Binary Decision Diagram

Zero-Suppressed Binary Decision Diagram (ZDD、ゼロサプレス型二分決定グラフ) [3] は集合族を有向非巡回グラフにより表現するデータ構造である。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。これまでに ZDD を用いて最長パス問題 [5] や、信頼性評価問題 [6]、様々なペンシルパズル [7] 等が解けることが知られている。ZDD は根ノード、2 種類の終端ノード ($\top, \bot$)、中間ノードからなる。終端ノードを除くノードは 0-枝、 1-枝 と呼ばれる出次枝を持ち、各枝は集合の要素を表すラベルを持っている。1-枝は集合にその要素が含まれることを、0-枝はその要素が含まれないことを表している。例えば、図 1 に示す ZDD は集合族 $\{\{a,b\},\{a,c\},\{c\}\}$ を表す。根から順にノードを辿り、$\bot$ に到達することは当該集合が含まれないことを表し、$\top$ への到達は集合族に含まれることを表す。ラベルとしてグラフ $G=(V,E)$ の辺 e_i, $1 \le i \le |V|$ を保持することで、G の部分グラフの集合を表現することができる。

3　フロンティア法

フロンティア法は ZDD をトップダウンに構築する手法であり、フロンティア法により全域木や森の列挙、

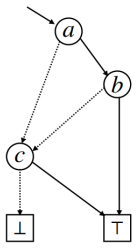

図 1　ZDD

マッチングの列挙など、様々な部分グラフ列挙問題を解けることが知られている [4]。フロンティア法では ZDD を根ノードを始点として幅優先順でノードを構築する。ラベル e_i がついた枝を処理した状態における頂点集合 $F_i = (\bigcup_{j=1,\dots,i} e_j) \cap (\bigcup_{j=i+1,\dots,|E|} e_j)$ を指してフロンティアと呼ぶ。構築の過程で各ノードに現時点の部分グラフのフロンティア上の頂点の情報を記憶し、同一の情報を持つノードが既に存在する場合はノードが共有される。

4　選挙区割問題

選挙区割問題における入力と出力を定義する。まず入力は日本の都道府県単位の地理情報を基に作成したグラフ $G=(V,E,h)$、区割個数を表す正整数 k、許容する格差を表す実数 $R \ge 1.0$ である。ここで V は頂点集合、 E は辺集合、 $h: V \to \mathbb{N}$ は頂点重み (=人口) を表している。出力は $|S|=k$ のグラフの分割 S で、$W_i = \sum_{v \in S_i} h(v)$, $W = \{W_1, W_2, \dots, W_k\}$ としたとき、$\frac{\max W}{\min W} \le R$ であるものの集合である。グラフの分割を $S = \{S_1, S_2, \dots, S_k\}$ $(S_i \subseteq V)$ で表すものとする。 ここで $\bigcup S_j = V, S_j \cap S_{j'} = \emptyset$ $(j \ne j')$ である。本研究では部分グラフの集合を ZDD として表現する。

5　区割列挙アルゴリズム

フロンティア法を用いて条件を満たす区割を列挙する ZDD を構築するアルゴリズムが知られている [8]。この場合、ノードに保存するべき情報はフロンティア上のノードの接続関係、部分グラフ中の各連結成分の重み、確定した連結成分の個数、確定した連結成分の重みの最大値や最小値である。

各連結成分の重みの和を昇順に $a_1, a_2, \dots, a_k$ とする。このとき $S = \sum_{v \in V} h(v)$ として、$S \ge (k-1)a_1 + a_k \ge \frac{(k-1)a_k}{r} + a_k$ より $U := \frac{rS}{r+(k-1)} \ge a_k$、 また $S \le a_1 + (k-1)a_k \le a_1 + r(k-1)a_1$ より $L := \frac{S}{r(k-1)+1} \le a_1$ を満たす必要がある。

上記の情報をノードに保持しつつフロンティア法により ZDD を構築し、連結成分個数が k を超えた場合、L を下回る連結成分や U を上回る連結成分が確定した場合など、適宜探索の枝刈りを行うことで所望の ZDD を得ることができる。

6　提案する近似手法

既存手法では多くのインスタンスにおいて解を得られるが、一部のインスタンスでは ZDD ノード数が多く、メモリ不足により解が得られていない。既存手法にお

1)　京都大学大学院情報学研究科

いて ZDD のノード数が膨大になる原因として、フロンティア上の値が全て一致しないと ZDD ノードの共有が行えないが、その状態数が非常に大きくなることが挙げられる。提案手法により各頂点の重みに近似を行い、有効数字の幅を減少させることで組合せの数が小さくなり、ZDD ノードが共有されやすくなることが期待される。このとき、近似方法に制限を設けることで出力に対し近似保証を与えることができる。

グラフ $G = (V, E, h)$ に対し、$h(v), v \in V$ を昇順に並べ替えたものを w_{sorted} とする。許容誤差 ϵ に対し、$\sum_{i=1}^{n'} w_{sorted} \geq \frac{rS}{r+k-1}$ となるような最小の整数 n' を用いて、$x = \frac{\epsilon S}{(r(k-1)+1)n'}$ を取り、$h(v) \leq h(v)' \leq h(v) + x, \forall v \in V$ を満たすグラフ $G' = (V, E, h')$ を考える。このとき、あるグラフの分割 S に対し G が格差 R を持つならば G' における格差 R' は $(1-\epsilon)R \leq R' \leq (1+\frac{\epsilon}{R})R$ を満たすことが示せる (詳細は省略する)。

このような h' の内、ZDD ノード数を削減することが期待されるものを考える。例えば、w' を全て 100 の倍数とすれば下 2 桁を切り上げることと同義であり、状態数の削減に繋がると考えられる。

7 実験結果

日本の都道府県データを使用して提案する近似手法による計算機実験を行った。入力の詳細は 表 1 に示す。実験では h' として制約を満たすものの内で最大公約数が最大となるようなものを取り、そのときの値を表 1 に併せて掲載している。ZDD のノード数は変数順序に大きく依存するが、今回はヒューリスティックにより変数順序を定めた。計算環境は CPU が Intel Xeon CPU E5-2637 v3 (3.50GHz)、メモリが 1.5TB である。TdZdd ライブラリ [9] を利用し、C++ により実装した。近似手法を用いない場合を比較対象とし、実行時間、ZDD ノード数、得られる解の個数の比較を行った。また、いずれの実験でも $R = 1.40$ とした。

まず従来手法による実行結果を表 2 に示す。Osaka ではメモリ不足により解が得られていない。

表 1　入力データ

	頂点数 $\|V\|$	辺数 $\|E\|$	区割個数 k	gcd ($\epsilon = 0.01$)	gcd ($\epsilon = 0.05$)
Gifu	45	105	5	101	505
Nara	39	94	3	108	588
Tokyo	62	145	20	167	871
Osaka	72	166	19	193	940

表 2　従来手法

	実行時間 (秒)	ノード数	解の個数
Gifu	2674.2	8,405,276,159	160,906,411
Nara	135.7	450,040,682	29,329,054
Tokyo	353.0	399,991,715	163,285,368
Osaka	N/A	N/A	N/A

提案手法の実験結果を表 3、4 に示す。許容誤差 $\epsilon = 0.01, 0.05$ として実験を行った。実験結果から、近似手法を用いた場合に得られる解の個数は数パーセントの増減である一方、実行時間は最大 88% 減少しており、ZDD が効率的に得られることが分かる。また、Osaka についても近似手法を用いることで保証付きの解を求めることができた。

表 3　近似手法 ($\epsilon = 0.01$)

	実行時間 (増減率)	ノード数	解の個数 (増減率)
Gifu	1837.8(-31.3%)	4,935,738,286	161,759,025(+0.53%)
Nara	47.3(-65.2%)	109,908,784	29,437,376(+0.37%)
Tokyo	263.7(-25.3%)	291,650,993	161,657,978(-1.00%)
Osaka	N/A	N/A	N/A

表 4　近似手法 ($\epsilon = 0.05$)

	実行時間 (増減率)	ノード数	解の個数 (増減率)
Gifu	847.6(-68.3%)	2,289,675,876	164,299,163(+2.11%)
Nara	15.6(-88.5%)	31,976,069	29,976,810(+2.21%)
Tokyo	146.5(-58.5%)	143,314,044	160,252,144(-1.86%)
Osaka	15627.1(N/A)	16,327,610,062	2,905,040,996(N/A)

8 結論

本研究では選挙区割問題に対する近似手法を提案した。都道府県データを用いた評価により、提案手法を用いることで多くのインスタンスにおいて得られる解の個数を大きく変えることなく、より効率的に ZDD を構築できることを確認した。また、従来手法では解の個数を見積もることも困難であったインスタンスに対しても近似率を保証した上で解を列挙する ZDD を構築することができることを確認した。

謝辞

実験データを提供していただいた堀田敬介先生（文教大学）に感謝します。本研究の一部は、JSPS 科研費 JP18H04091、JP20H00605 の助成を受けたものです。

参考文献

[1] Andreev, K. and Racke, H.: Balanced graph partitioning, Theory of Computing Systems, Vol. 39, No. 6, pp. 929–939 (2006)

[2] 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, Vol. 20, pp. 136–147 (2005)

[3] Minato, S.: Zero-suppressed BDDs for set manipulation in combinatorial problems, Proceedings of the 30th international Design Automation Conference, pp. 272–277 (1993)

[4] Kawahara, J., Inoue, T., Iwashita, H. and Minato, S.: Frontier-based search for enumerating all constrained subgraphs with compressed representation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. 100, No. 9, pp. 1773–1784 (2017)

[5] Kawahara, J., Saitoh, T., Suzuki, H. and Yoshinaka, R.: Solving the longest oneway-ticket problem and enumerating letter graphs by augmenting the two representative approaches with ZDDs, International Conference on Computational Intelligence in Information System, Vol. 532, Springer, pp.294–305 (2016)

[6] Hardy, G., Lucet, C. and Limnios, N.: K-terminal network reliability measures with binary decision diagrams, IEEE Transactions on Reliability, Vol. 56, No. 3, pp. 506–515 (2007)

[7] Yoshinaka, R., Saitoh, T., Kawahara, J., Tsuruma, K., Iwashita, H. and Minato, S.: Finding all solutions and instances of numberlink and slitherlink by ZDDs, Algorithms, Vol. 5, No. 2, pp. 176–213 (2012)

[8] Kawahara, J., Horiyama, T., Hotta, K. and Minato, S.: Generating all patterns of graph partitions within a disparity bound, InternationalWorkshop on Algorithms and Computation, Vol. 10167, Springer, pp. 119–131 (2017)

[9] Iwashita, H. and Minato, S.: Efficient top-down ZDD construction techniques using recursive specifications, TCS Technical Report TCS-TR-A13-69, Hokkaido University, Division of Computer Science (2013)