修士研究論文

クラスター編集問題に対する効率的な解法の研究

20N8100005L 木村　陸

中央大学理工学部情報工学科　離散アルゴリズム研究室

2022 年 3 月

要約: クラスター編集問題とは、単純無向グラフである入力グラフを、最小回数の辺操作で非連結なクリークの集合に変換する問題である．この問題は NP 困難であることが知られているが、本研究ではデータの前処理、ヒューリスティック、分枝アルゴリズムなど、いくつかの手法を組み合わせることで計算の高速化を行う．

キーワード: カーネル化, 衝突, 分枝限定法

目次

1　初めに

$G_w = (V, E)$ を無向グラフとすると, 重み付きクラスター編集問題は以下のように定義される. 任意の頂点ペア $\{u, v\} \in \binom{V}{2} = \{\{u, v\} : u, v \in V, u \neq v\}$ に対して, $\{u, v\} =\in E$ の場合に G_w から $\{u, v\}$ を削除するコスト, $\{u, v\} \notin E$ の場合に G_w に $\{u, v\}$ を追加するコストがわかっているとする. このとき, 目的は最小合計コストによる辺操作で G_w をクラスターグラフ (非連結なクリークの集合から成るグラフ) に変化することである.

応用において, 上記のタスクはオブジェクトのクラスタリングに対応する. つまり, よく似たオブジェクトの集合を十分に分離された部分集合に分割する. データのクラスタリングは遺伝子発現データを使用した組織同定のためのクラス発見のような多くの生物学的及び医学的問題の重要な問題に用いられる.

先行研究において, 重みなしクラスター編集問題の NP 困難性が Křivánek と Moráve[] により示された.

また, 重み付きクラスター編集問題用に CLICK[], CAST[], HCS[] のようなヒューリスティックククラスタリングアルゴリズムを用いたソフトウェアも存在している. 重みなしクラスター編集問題は APX 困難であり

2.5 の定数近似が存在する [].

厳密解を見つけるために, Grötschel と若林 [] は整数計画問題としてクラスター編集問題を計算した.

重みなしクラスター編集問題において, 最小の辺操作の回数を k としたパラメータ化アルゴリズムは十分に研究されている. 修正コストがゼロの辺が含まれた問題は未だ未解決である [].

クラスター編集問題は応用面のおいて非常に重要で価値のある問題であるが, サイズが大きいインスタンスでは厳密アルゴリズムで現実的な時間で解くことが困難である. したがって, 本研究ではデータの前処理, ヒューリスティック, 分枝アルゴリズムなどのいくつかの手法を組み合わせて厳密アルゴリズムの計算の

高速化を狙う.

2 準備

グラフ内の互いに素な成分が全てクリークで構成するグラフを **クラスターグラフ** と呼ぶ. クラスター編集問題とは与えられたグラフに対して, 辺の挿入や削除などを行い, クラスターグラフに変換する問題である.

uv を順序づけられていないペア $u, v \in V$ の省略形とする. 単純無向グラフであるようなクラスター編集問題のインスタンスに対してコスト関数 $s\colon \binom{V}{2} \to \mathbb{R}$ を導入する. $s(uv) > 0$ の場合, ペア uv はグラフの辺であり, 削除コストが $s(uv)$ だけかかる. $s(uv) < 0$ の場合, ペア uv はグラフの辺ではなく, 挿入コストが $-s(uv)$ だけかかる. $s(uv) = 0$ の場合, uv をゼロ辺と呼ぶ. ゼロ辺が存在する場合クラスター編集問題を解くことが難しくなるので, ゼロ辺ができないように気をつけなければならない. 重みなしのグラフインスタンスに対しては, コスト関数 $s(uv) \in +1, -1$ を導入することにより, 重みありのときと同様に解くことができる.

以上より, クラスター編集問題の問題設定は以下のように書ける.

クラスター編集問題の問題設定

入力:

単純無向グラフ $G = (V, E)$

コスト関数 $s : \binom{V}{2} \to \mathbb{R}$

出力:

G を辺操作を行なった結果生まれるクラスターグラフ

辺操作を行なった辺集合

評価:

辺操作を行なった際に生じたコストを最小化する

クラスター編集問題を解くために, 初めに入力グラフの全ての連結成分を識別し, 各連結成分に対して最適解を個別に計算する. なぜなら最適解は接続されていない連結成分を接続しないためである. さらに, アルゴリズムの過程でグラフが分解された場合, 連結成分を個別に再帰的に処理できる.

3 頂点の組 uvw について, uv と vw が辺であるが, uw が辺でない場合, uvw を **衝突** と呼ぶ. 非連結なクリークの集合から成るグラフを **推移的** と呼ぶが, これはグラフ内に衝突が存在しない場合のみ当てはまる. したがって, クラスター編集問題を解くために入力グラフから衝突を全て削除することを目的とする.

問題を解く上で, ペア uv を **「禁止」**,**「永続」** に設定する場合がある. 「禁止」とは 2 つの頂点が同じクリーク/クラスターには必ずならないことを表す. 逆に, 「永続」とは 2 つの頂点が同じクリーク/クラスターに必ずなることを表す. この操作は前処理のステップあるいは後述する本研究のアルゴリズムで用いる探索木内で発生する可能性がある. 重み付きクラスター編集問題の利点は, 任意の永続ペア uv をすぐに縮約して, 頂

点

u と v を単一の頂点 u' に置き換え, 2 つのペア uw, vw を単一のペア $u'w$ のように全ての頂点 $w \in V \backslash \{u, v\}$

についてペアを置き換えることができることである. ペアの一方が辺で, もう一方が辺でない場合, これはすぐ

に「コストを生成」し, パラメータ化アルゴリズムの場合, コストパラメータを減らす.

$s(uv) = -\infty$ と設定することにより, 禁止されたペア uv を表す. 定義上, $s(uv) < 0$ であるため, 全ての

禁止されたペアは辺ではない. そして, 禁止されたペア uw は衝突 uvw の一部である可能性がある. また, こ

の操作は禁止された辺に付随する頂点を縮約する際にも正しく機能する.

3 先行研究

3.1 計算複雑性

3.2 パラメータ化アルゴリズムとデータ削減

辺操作の数をパラメータ k として使用する, クラスター編集のパラメータ化複雑性は特によく研究されている. パラメータ化アルゴリズムでは, 事前にコスト制限 k を指定する必要がある. コスト $\geq k$ の解が存在する場合, アルゴリズムはこの解を見つけて「はい」と返す. それ以外は「解なし」と返される. 最適解を見つけたいが k がわからない場合は $k = 0, 1, 2...$ のアルゴリズムを繰り返し呼び出すことで求めることができる. このとき, 最後の反復の実行時間が以前の呼び出しの実行時間を支配するため, 最悪の場合の時間計算量は

以下で紹介するようなものとなる.

クラスター編集問題のパラメータ化複雑性に関する最初の結果は, Gramm ら [] によって与えられた. 実行時間 $O(3^k + n^3)$ の単純なアルゴリズムを提案し, 洗練された分枝戦略を用いて $O(2.27^k + n^3)$ へと改善した. 後に Gramm ら [] は自動化された広範なケース分析によってこれを $O(1.92^k + n^3)$ に改善した. しか

し, 結果として得られるアルゴリズムには, 100 を超える初期分枝ケースがあり, 現在まで知る限り実装されたことはない. 重み付けされていないクラスター編集問題を整数に重み付けされた問題に変換することにより, 実行時間は Böcker ら [] によって $O(1.82^k + n^3)$ に進められた. その後, Böcker と Damaschke は, 多くの衝突を含まないグラフと特性を使用して, 実行時間 $O(1.76^k + m + n)$ のアルゴリズムを導出した []. 現在最速のアルゴリズムは, $O(1.62^k + m + n)$ 時間でクラスター編集問題を解くことができる.

カーネル化は, パラメータ k を持つ特定のインスタンス I をパラメータ $k' \leq k$ を持つ新しいインスタンス I' に変換する多項式時間アルゴリズムであり, 以下の条件を満たす. (i) インスタンス (I, k) はインスタンス (I', k') が解をもつ場合に限り解を持つ. (ii) インスタンス I' のサイズが計算可能な関数 f に対して最大で $f(k)$ である. カーネル化は多くの場合, 処理しやすいインスタンスの部分を切り取る一連の削減ルールを適

用することによって実現される. カーネルサイズはカーネルの有効性の尺度を表す.

Gramm ら [] は $O(n^3)$ 時間で $O(k^2)$ 頂点を使用したクラスター編集問題のカーネルを計算する方法を示した. また, より高速なカーネル化は, $O(n + m)$ 時間で $2k^2 + k$ 頂点を持つカーネルを計算した Prottiet ら [] によって与えられた. IWPEC 2006 では, Fellows[] は, 問題の線形計画法の定式化に基づいて, $24k$ の頂点を持つ多項式時間カーネルを提案した. 2007 年には, Fellow ら [] は, クラウンタイプの削減に基づいて $6k$ の頂点を持つ組み合わせカーネルを提案した. Guo[] はクリティカルクリークの概念に基づいて, $O(n^3)$ 時間で計算できる $4k$ の頂点を持つカーネルを提示した. これは後に Chen と Cao[] によって $2k$ 頂点に改善された. 同様に, $O(k^2)$ 頂点 [] を持つ整数重みの問題のカーネルは $2k$ 頂点 [] に改善された.

カーネル法は, データ削減と見なすことできる. パラメータ化されたデータ削減ルールは, [] で説明されているようにパラメータに依存しないようにすることができる. これにより, (多項式時間の) データ削減を事前に適用し, 厳密アルゴリズムあるいはヒューリスティックアルゴリズムを用いて残りのインスタンスを解く

ことができる.

4 アルゴリズム

提案手法の大まかな流れは以下の通りである.

提案手法の流れ

Step1:

グラフインスタンスに対してカーネル化アルゴリズムを用いてデータ削減を行う.

Step2:

Step1 を適用したインスタンスに対して, パラメータ化アルゴリズムを用いて厳密解を求める.

カーネル化アルゴリズムは Cao と Chen[] が提案した辺カットに基づいたアルゴリズムを用いる. パラメータ化アルゴリズムについては最悪計算量が最良ではないが非常に単純な分枝アルゴリズムを用いている. なら, どんなグラフインスタンスにも適用することができ, 分枝戦略が非常に単純で実装が容易であるからである.

4.1 カーネル化アルゴリズム

4.2 分枝アルゴリズム

今回用いる分枝アルゴリズムは衝突 uvw 内の辺 uv に対して以下の 2 つの分枝ケースをグラフ内に衝突がなくなるまで再帰的に行う.

(a) uv を禁止にする

(b) uv をマージする

この再起的手順がいつか最適解を生成することを示していく. 以下では探索木のサイズを分析する. 辺 uv

を削除するときは, パラメータを $s(uv)$ だけ減らす. 頂点 u と v を縮約するとき, 各頂点 w について, ペア uv と vw が辺である場合はパラメータは変化せず重み $s(uv) + s(vw)$ として新たなペアを作る. ペア uv が辺であるが, vw が辺でない場合について, (i) $s(uw) \neq -s(vw)$ の場合, パラメータ k は $min\{s(uw), -s(vw)\}$ だけ下げることができる. (ii) $s(uw) = -s(vw)$ の場合,

4.3 分枝アルゴリズムの高速化

前述した分枝アルゴリズムを高速化するためにいくつかの工夫を取り入れた. 具体的には以下の工夫である.

(A) 事前にヒューリスティックアルゴリズムで解を求め, それをパラメータ k とする.

(B) 再帰 1 回ごとにしていたグラフの衝突の組の計算を減らす.

(C) 縮約もしくは禁止にする衝突内の辺の選び方の工夫.

(D) (C) の工夫において LP の解を用いる.

(A) については, $\cdots$

5 実験

今回はヒューリスティックアルゴリズム, 厳密アルゴリズム両方の実験を行う. ヒューリスティックでは 4 章で述べた, ランダムピボット, LP ピボット, LP を用いたヒューリスティックアルゴリズムを解と計算速度の観点から比較していく. 厳密アルゴリズムでは, 4.2 で述べた分枝アルゴリズムを基準に 4.3 の工夫を取り入れるか否かのパターンで実験を行う. つまり, $2^4 = 16$ 通りのパターンを計算速度の観点から比較する. データはクラスター編集問題を扱ったコンペティション, "PACE2021"で用いられた厳密アルゴリズム用のインスタンスを用いる.

6 結論

参考文献

[1] Sharan, R., Maron-Katz, A., Shamir, R.: CLICK and EXPANDER: a system for clustering and visualizing gene expression data. Bioinformatics 19(14), 1787 – 1799 (2003)

[2] Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J. Comput. Biol. 6(3-4), 281 – 297 (1999)

[3] Hartuv, E., Schmitt, A.O., Lange, J., Meier-Ewert, S., Lehrach, H., Shamir, R.: An algorithm for clustering cDNA fingerprints. Genomics 66(3), 249 – 256 (2000)

[4] Böcker, S.: A golden ratio parameterized algorithm for cluster editing. J. Discrete Algorithms 16, 79 – 89 (2012)

[5] Cao, Y., Chen, J.: Cluster editing: Kernelization based on edge cuts. Algorithmica 64(1), 152 – 169 (2012)

[6] Böcker, S., Briesemeister, S., Klau, G.W.: Exact algorithms for cluster editing: Evaluation and experiments. Algorithmica 60(2), 316 – 334 (2011)

[7] Gramm, J., Guo, J., H ¨ uffner, F., Niedermeier, R.: Graph-modeled data clustering: Fixed-parameter algorithms for clique generation. Theory Comput. Syst. 38(4), 373 – 392 (2005)

[8] Gramm, J., Guo, J., H ¨ uffner, F., Niedermeier, R.: Automated generation of search tree algorithms for hard graph modification problems. Algorithmica 39(4), 321 – 347 (2004)

[9] Böcker, S., Briesemeister, S., Bui, Q.B.A., Truss, A.: Going weighted: Parameterized algorithms for cluster editing. Theor. Comput. Sci. 410(52), 5467 – 5480 (2009)

[10] Böcker, S., Damaschke, P.: Even faster parameterized cluster deletion and cluster editing. Inform. Process. Lett. 111(14), 717 – 721 (2011)

[11] Protti, F., da Silva, M.D., Szwarcfiter, J.L.: Applying modular decomposition to parameterized cluster editing problems. Theory Comput. Syst. 44(1), 91 – 104 (2009)

[12] Fellows, M.R.: The lost continent of polynomial time: Preprocessing and kernelization. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 276 – 277. Springer, Heidelberg (2006)

[13] Fellows, M.R., Langston, M.A., Rosamond, F.A., Shaw, P.: Efficient parameterized preprocessing for cluster editing. In: Csuhaj-Varj ´ u, E., Esik, Z. (eds.) FCT 2007. ´ LNCS, vol. 4639, pp. 312 – 321. Springer, Heidelberg (2007)

[14] Guo, J.: A more effective linear kernelization for cluster editing. Theor. Comput. Sci. 410(8-10), 718 – 726 (2009)

[15] Chen, J., Meng, J.: A 2k kernel for the cluster editing problem. J. Comput. Syst. Sci. 78(1), 211 – 220 (2012)