Vol.2017-MPS-116 No.12
2017/12/11

辺重み付きグラフからのクリーク群抽出による構造分析

石原 諒大[1,a)]　山口 一章[1,b)]　増田 澄男[1,c)]

概要：近年，様々な情報がグラフとして表現され，分析が行われている．本稿では，項目間の関係の強さが与えられたときに繋がりの強いグループを探し出すような状況に対し，辺に重みのあるグラフ上の最適化問題としての定式化を提案する．また，その最適化問題の解を実用的な時間で得る発見的手法を提案する．ツイッターから作成したグラフを用いた実験により，定式化の妥当性および解法の有効性を検証する．

1. まえがき

物事の関係を表す方法としてグラフは様々な場面で用いられている．その量や種類は年々増加しており，その構造の解析について様々な研究が行われている．グラフの頂点は分析の対象となる項目に対応し，辺は項目間の関係の有無を表す．例えば，SNS におけるユーザ間の関係を分析する際は，各ユーザーを頂点とし，定期的にメッセージのやり取りを行っているなどの結びつきの強いユーザ間に辺を持つようなソーシャルグラフが用いられる．

グラフにおける密な部分グラフはクラスタと呼ばれる．特に，任意の 2 頂点間に辺が存在するような部分はクリークと呼ばれる．クリークのうち，他のクリークに包含されないようなものは極大クリークと呼ばれる．クラスタやクリークは元のデータにおける，関係性の深い集団（コミュニティ）に対応する．グラフのクラスタリングに基づくコミュニティ抽出に関しては数多くの研究が行われている．また，極大クリークの抽出に基づく分析法についても研究が行われている．グラフ中には指数関数的な数の極大クリークを含み得るので，全てを列挙するのではなく，極大クリークのうちある特徴を持つもののみ抽出する方法や，グラフ中の極大クリークの数が抑えるようなグラフの変形処理が提案されている．

本稿ではいくつかのコミュニティを内包するグラフが以下のような構造を持つものと仮定する．

- グラフの辺は向きを持たず，実数の重みを持つものとする．
- コミュニティはクリークを構成する．コミュニティ内の項目間の繋がりの強さはほぼ同じであるとする．
- グラフの辺の重みはその辺の両端点の頂点が属するコミュニティの繋がりの強さとする．
- 二つの頂点が複数のコミュニティに含まれる場合，その辺の重みはともに属するコミュニティの繋がりの強さの和となる．

以降，クリークはコミュニティと同義であるとし，コミュニティ内の繋がりの強さをクリークの重みと呼ぶことにする．

例として図 1 のグラフ G_1 が上記の構造を持つと仮定し，どのようなコミュニティによって構成されているかを考える．もし G_1 が 3 つのコミュニティからなるならば，重みが 30 のクリーク C_1，重みが 5 のクリーク C_2，重みが 10 のクリーク C_3 から構成されていると考えるのが自然である．辺 (v_1, v_3) の重みは 6 でクリークの重みに合致しないが，分析対象のグラフには誤差が生じ得るものと考える．本稿では，辺に重みのあるグラフが与えられたときに誤差の合計を最小にするようなクリークと各クリークの重みを求めるような問題の定式化を提案する．また，この問題に対する発見的手法を提案する．

本稿の構成は以下の通りである．2. で上記の最適化問題の数学的定式化とその特徴を示す．3. でその最適化問題に対する局所探索法の概要を示す．4. で計算機実験とその結果を示す．5. でまとめと今後の課題について述べる．

2. 定式化

1. に示した最適化問題の定式化を以下に示す．

[1] 神戸大学
Kobe Uniersity
[a)] 164t202t@stu.kobe-u.ac.jp
[b)] ky@kobe-u.ac.jp
[c)] masuda@kobe-u.ac.jp

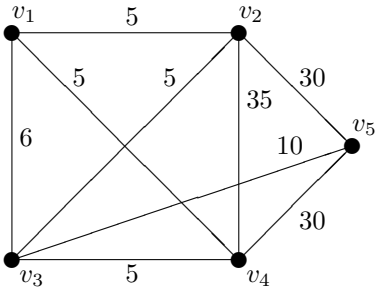

図 1 グラフ G_1

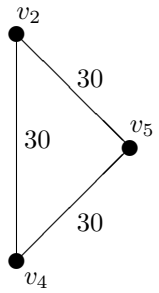

図 2 クリーク C_1

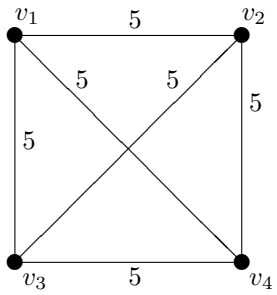

図 3 クリーク C_2

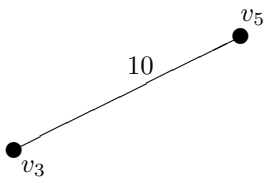

図 4 クリーク C_3

重み付きクリーク群抽出問題

入力として無向グラフ $G=(V,E)$ と各辺 (u,v) の重み $w_{u,v}$，自然数 k が与えられたとき，下記 $\delta(G,\mathcal{C},w)$ の値を最小にするような k 個のクリーク $\mathcal{C}=\{C_1,\ldots,C_k\}$ と各クリークの重み $w(C_1),w(C_2),\ldots,w(C_k)$ を求めよ．

$$\delta(G,\mathcal{C},w)=\sum_{\{u,v\}\subseteq E}|w_{u,v}-\sum_{C_i\in\mathcal{C}}\sum_{C_i\supseteq\{u,v\}}w(C_i)|$$

以降，重み付きクリーク群抽出問題（Weighted Cliques Extraction Problem）を WCEP と略記する．1. で示した通り，グラフ G_1 と $k=3$ を入力としたときの WCEP の最適解は $\mathcal{C}=\{C_1,C_2,C_3\}$，$w(C_1)=30$，$w(C_2)=5$，$w(C_3)=10$ であり，目的関数の値（辺重みの誤差の和）は $\delta(G,\mathcal{C},w)=1$ となる．$k=2$ ならば最適解は $\mathcal{C}=\{C_1,C_2\}$，$w(C_1)=30$，$w(C_2)=5$ で $\delta(G,\mathcal{C},w)=11$ となる．

WCEP を考える意義はいくつかある．辺に重みの付いたグラフを扱うので，その辺の重要度を考慮できるということがまず挙げられる．例えば，グラフ G_1，$k=1$ の最適解は要素数最大の C_2 でも図 5 のクリーク C' でもなく，極大でないクリーク C_1 である．これは，v_2,v_4,v_5 の間の辺重みに比べ v_3 との間の辺は重みが小さいため，異質のものとして排除され，その上で辺の重みが大きい（重要な）クリークが抽出されたと考えられる．

別の例を示す．以降，頂点集合 S による G の頂点誘導部分グラフを $G(S)$ と記す．グラフ G_2 には多数のクリークが存在するが，$k=3$ のときの最適解のクリークは $G_2(\{v_1,v_2,v_3\})$，$G_2(\{v_3,v_5,v_6\})$，$G_2(\{v_2,v_4,v_6\})$ で，

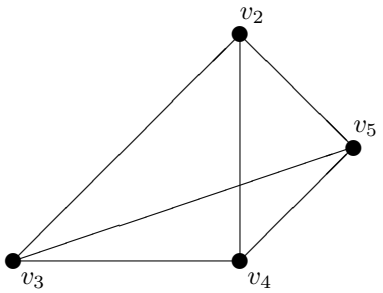

図 5 クリーク C'

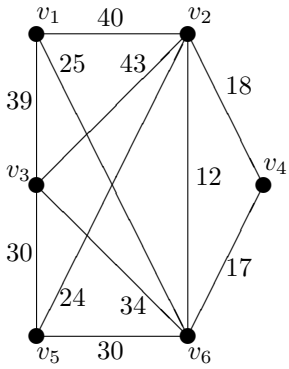

図 6 グラフ G_2

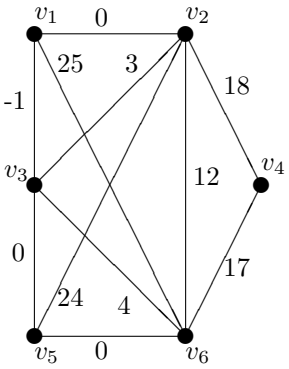

図 7 グラフ G_3

重みはそれぞれ 40, 30, 17 となる．G_2 中，辺 (v_2,v_6) を含む極大クリークは $G_2(\{v_1,v_2,v_6\})$, $G_2(\{v_2,v_3,v_6\})$, $G_2(\{v_2,v_4,v_6\})$, $G_2(\{v_2,v_5,v_6\})$ の 4 個であるが，最適解に含まれるクリーク $G_2(\{v_2,v_4,v_6\})$ は G_2 において辺の重みが最小である．なぜそれが選ばれたのかは，G_2 から重みの大きな二つのクリークの重みを引き去ったグラフ G_3 を見れば分かる．G_2 においては辺重みが最小の極大クリークであった $G_2(\{v_2,v_4,v_6\})$ は G_3 では重み最大となっている．

以上に示した通り，WCEP では，単に重みの大きい順にクリークが解として選ばれるのではなく，クリーク内から異質なものを排除したり，辺に重複のある類似したクリークを選ぶことを回避しようとする性質があると考えられる．このように，WCEP で得られる解は従来の方法では得られないものであり，グラフの構造分析において有用であると考えられる．

3. 局所探索法

WCEP はグラフの全ての辺の重みが 1 で $k=1$ の場合，最大クリーク問題となる．最大クリーク問題はＮＰ困難であるのでクリーク群抽出問題もＮＰ困難であり，最適解を得るのは困難である．そこで，本稿ではクリーク群抽出問題に対して短い時間である程度の良い解が得られる局所探索法を提案する．提案法は，最大重みクリークを求める局所探索法を繰り返し用いて k 個のクリークを作成してはその一部を入れ替えるという処理を繰り返すものである．

以下では提案法の中で用いられる記号の定義を以下に

Algorithm 1 クリーク群探索

INPUT: $G=(V,E)$, k, α, L, $timeLimit$
OUTPUT: $\mathcal{C}_{best}$,w_{best}
1: $\mathcal{C} \leftarrow \emptyset$
2: $\mathcal{C}_{best} \leftarrow \emptyset$
3: **while** $time < timeLimit$ **do**
4: **while** $|\mathcal{C}| < k$ **do**
5: $C, w \leftarrow CliqueSearch(G, \mathcal{C}, w, L)$
6: $\mathcal{C} \leftarrow \mathcal{C} \cup C$
7: **end while**
8: **if** $\delta(G,\mathcal{C},w) < \delta(G,\mathcal{C}_{best},w_{best})$ **then**
9: $\mathcal{C}_{best} \leftarrow \mathcal{C}$
10: $w_{best} \leftarrow w$
11: **end if**
12: $\mathcal{C}$ から $\lfloor \alpha \cdot k \rfloor$ 個のクリークをランダムに除く．
13: w を更新する．
14: **end while**

示す．頂点 v に隣接する頂点の集合を $N(v)$ と表す．頂点集合 $S \subseteq V$ に対し，$N(S) = \cap_{v \in S} N(v)$ と定める．局所探索における探索中の暫定的なクリークを C とする．$C_0 = N(C)$ と定める．C_1 を以下のように定める．

$$C_1 = \{v \mid \exists u,\ v \in N(C \setminus \{u\})\} \setminus (C \cup C_0)$$

すなわち，C_1 は，C からある一つの頂点を取り除けば C_0 に含まれる可能性のある頂点のうち，現時点では C にも C_0 にも含まれないものの集合である．

クリーク探索では以下の操作を用いる．

- Add : C に頂点 $v \in C_0$ を加える．
- Drop : C から頂点 $v \in C$ を除く．
- Swap : C に頂点 $v \in C_1$ を加え，C から $u \notin N(v)$ なる頂点 u を 取り除く．

提案手法のベースとなるアルゴリズムを Algorithm 1 に示す．入力として，WCEP の入力である辺重み付き無向グラフ $G=(V,E)$ と自然数 k 以外に，1 未満の正実数パラメータ α，局所探索における繰り返し回数を定める自然数 L，計算時間の制限 $timeLimit$ が与えられるとする．$\mathcal{C}$ が求めるべきクリークの集合である．

初めに $\mathcal{C}$ を空にする．局所探索 $CliqueSearch$ により $\delta(G,\mathcal{C} \cup C, w)$ が小さくなるようなクリーク C を探索し，その結果得られた解の評価値 w' を計算する．この処理を繰り返し行い，$\mathcal{C}$ のサイズが k になるまで C を $\mathcal{C}$ に加えていく．次に $\mathcal{C}$ の再初期化として $\mathcal{C}$ から $\lfloor \alpha \cdot k \rfloor$ 個のクリークをランダムで取り除く．以上の操作を制限時間内繰り返し，最も $\delta(G,\mathcal{C},w)$ が小さいクリーク群 $\mathcal{C}_{best}$ と対応する各クリークの重み w_{best} を解として出力する．

クリーク探索 (line 5 :$CliqueSearch$) については 3 通り試した．以下でそれらについて説明する．

3.1 Multi Neighborhood Tabu Search

Multi Neighborhood Tabu Search[1]（以降，MN/TS と略す）はタブーサーチに基づく解法であり，3 つの近傍解集合を用いる点に特徴がある．

Algorithm 2 MN/TS

1: **function** $CliqueSearch(G,\mathcal{C},w,L)$
2: C と tabuList を初期化
3: $NI \leftarrow 0$
4: $C_{best} \leftarrow C$
5: $w_{best} \leftarrow w$
6: **while** $NI < L$ **do**
7: C と tabuList から N_1, N_2, N_3 を求める
8: $C \leftarrow$ 最良の近傍解 $C_N \ \in N_1 \cup N_2 \cup N_3$
9: w と tabuList を更新
10: $NI \leftarrow NI + 1$
11: **if** $\delta(G,\mathcal{C} \cup C, w) < \delta(G,\mathcal{C} \cup C_{best}, w_{best})$ **then**
12: $NI \leftarrow 0$
13: $C_{best} \leftarrow C$
14: $w_{best} \leftarrow w$
15: **end if**
16: **end while**
17: **return** C_{best}, w_{best}
18: end function

MN/TS のアルゴリズムを Algorithm 2 に示す．探索の際，タブーリストを考慮しながら C から以下に示す三つの近傍 $N_1(C), N_2(C), N_3(C)$ を作る．ただし $N_1(C)$ は候補集合 $C_0 \setminus tabuList$ から頂点を選び C に Add 操作を行うことで得られるクリークの集合，$N_2(C)$ は候補集合 $C_1 \setminus tabuList$ から頂点を選び C に Swap 操作を行うことで得られるクリークの集合，$N_3(C)$ は C に Drop 操作を行うことで得られるクリークの集合である．これら $N_1(C), N_2(C), N_3(C)$ から $\delta(G,(\mathcal{C} \cup C), w)$ が最も小さくなる C 選択する．その後，$w(C)$ が C によって得られる部分グラフに含まれる辺の重みの中央値を示すように w を更新する．以上の操作を繰り返す．1 度取り除かれた頂点は T 回ループの間，$tabuList$ に加えられる．$tabuList$ に含まれる頂点を加えることによって得られる近傍解は選択できない．L 回ループしても解が更新されなくなったら，それまでに見つかった解の中で最良のクリークとそのクリークの重みを出力する．

3.2 リストヒューリスティック

リストヒューリスティック [2] は頂点系列を任意の場所から順番に選択して解の改良する手法である．1 回のループよって得られる解の質はよくないが，短い時間でたくさんの解を求めることができる．

リストヒューリスティックのアルゴリズムを Algorithm 3 に示す．はじめに何らかの基準に従って頂点系列 Π を作成する．アルゴリズム中の $\Pi(i)$ は系列 Π 中の i 番目の要素をあらわす．その頂点系列を先頭から順番に選択していく．ただし，暫定解 C に Add できない場合や解が改良されない場合，その頂点は解に加えずにスキップする．リストを 1 周したら探索を止める．続いて，C を初期化した後に，はじめに選ぶ頂点の位置を 1 つ分だけリストの末尾方

Algorithm 3 リストヒューリスティック

```
1: function CliqueSearch(G, C, w, L)
2:     C_best ← ∅
3:     w_best ← w
4:     頂点系列 Π を作成
5:     i ← 1
6:     while i ≤ L do
7:         C ← {Π(i)}
8:         j ← 1
9:         while j ≤ |V| do
10:            if Π(j) ∈ C_0 then
11:                C' ← C ∪ {Π(j)}
12:                w' ← 更新された w
13:                if δ(G, (C ∪ C'), w') < δ(G, C ∪ C, w) then
14:                    C ← C'
15:                    w ← w'
16:                end if
17:            end if
18:            j ← j + 1
19:        end while
20:        if δ(G, C ∪ C, w) < δ(G, C ∪ C_best, w_best) then
21:            C_best ← C
22:            w_best ← w
23:        end if
24:        i ← i + 1
25:    end while
26:    return C_best, w_best
27: end function
```

Algorithm 4 Add/Swap 探索

```
1: function CliqueSearch(G, C, w, L)
2:     C ← ∅
3:     NI ← 0
4:     C_best ← C
5:     w_best ← w
6:     while NI < L do
7:         NI ← NI + 1
8:         while C_0 ≠ ∅ do
9:             C ← 最良の近傍解 C_N1 ∈ N_1
10:            w を更新
11:            タブーリストを更新
12:        end while
13:        if δ(G, C ∪ C, w) < δ(G, C ∪ C_best, w_best) then
14:            NI ← 0
15:            C_best ← C
16:            w_best ← w
17:        end if
18:        if C_1 ≠ ∅ then
19:            C ← 最良の近傍解 C_N3 ∈ N_3
20:            w を更新
21:        end if
22:        タブーリストを更新
23:    end while
24:    return C_best, w_best
25: end function
```

向に移動させてから先ほどと同様にクリークを生成する．この処理を L 回繰り返す．そして，生成された L 個のクリークのなかで最良のクリークとそのクリークの重みを出力する．

3.3 Add/Swap 探索

Add/Swap 探索のアルゴリズムを Algorithm 4 に示す．Add を実行できる間，N_1 の中で最良の解に遷移し続ける．ただし，解の値が改悪されてしまうような解には遷移しない．Add が実行できなくなったら，Swap を実行する．ただし，N_3 の中で最良の解に遷移する．Swap で取り除かれた頂点は Add もしくは Swap が T 回実行されるまでタブーリストに加えられる．以上の操作を L 回ループしても解が更新されなくなるまで繰り返す．最後に探索した中で最良のクリークとそのクリークの重みを出力する．

4. 計算機実験

提案する三つの探索法について比較を行った．また，WCEP と極大クリーク列挙について得られるクリークの違いを比較した．以下ではそれらの結果を順に示す．

4.1 計算機実験 1

提案手法を C++ で実装し比較を行った．実験に使用した計算機の CPU は Intel(R) Core(TM)2 Duo T7700 2.40GHz，メモリは 2GB，OS は CentOS 6.9 である．使用したコンパイラは g++ 4.9.2 で最適化オプション-O2 を利用した．

実験ではランダムグラフ，DIMACS [3]，BHOSLIB [4]，ニュース記事に含まれるセンテンスから作られたグラフ（RTN グラフ） [5] 及び Twitter のツイートに含まれる単語の共起関係を元にしたグラフ（tweet グラフ）を用いた．ランダムグラフ，DIMACS 及び BHOSLIB の各辺には最大で 10，最小で 1 の重みを与えた．

各インスタンスに対して k は 5,30,100 の三通り, 制限時間 60 秒で乱数の種を変えて 5 回実行した．各手法のパラメータは，予備実験の結果から $\alpha = 0.5$,$T = 7$,$L = 100$ とした．リストヒューリスティクの頂点系列の生成方法は予備実験の結果からランダムにした．

表 1 に実験結果の比較を示す．表の値は $\delta(G, \mathcal{C}, w)$ の平均値が他の手法と比較して最も小さかったインスタンスの数を示す．e は辺の数，d はグラフの辺密度,cc/d はグラフのクラスタ係数 cc と辺密度 d の比を表している．頂点のクラスタ係数は隣接頂点間にどの程度辺が存在するかを示しており，グラフのクラスタ係数とは全ての頂点のクラスタ係数の平均値である．現実世界の情報をもとにして作られたグラフは辺密度に対してグラフのクラスタ係数が高くなる傾向がある．

MN/TS はグラフのクラスタ係数と密度の比が小さくて比較的大規模なグラフに対して有効だった．リストヒューリスティックは全てのインスタンスに対して他の手法よりも劣っていた．Add/Swap 探索はグラフのクラスタ係数と

表 1　各手法における実験結果の比較

インスタンス			手法		
cc/d	e	d	MN/TS	リスト	シンプル
>3			12	8	25
≤ 3	$>10^5$	>0.75	104	1	12
≤ 3	$>10^5$		115	1	43
≤ 3	$<10^5$	<0.75	53	29	120
≤ 3	$<10^5$		93	31	169

密度の比が大きいグラフや比較的規模が小さいグラフに対して有効だった．

MN/TS は探索に時間はかかるが，より良い解を丁寧に見つけだすことができるので，遷移先が多すぎて質の良い解が見つけにくい大規模なグラフに対して有効だった．

一方，Add/Swap 探索は MN/TS よりも 1 回の探索で求まる解の質が少し下がるが探索は速いので，解の遷移先が少ないグラフのクラスタ係数と密度の比が大きいグラフや比較的規模が小さいグラフに対して有効だった．

リストヒューリスティックは質の悪い解が見つかり易いというデメリットをその探索の速さで補う手法であるが，各手法のベースとなる Algorithm 1 では以前に見つけたクリークの影響を強く受けて次に探索されるグラフが作られるので，1 つでも質の悪い解が途中で採用されてしまうと，最終的な解の質が悪くなってしまう．したがって，リストヒューリスティックはいずれのインスタンスに対しても他の手法より劣っていた．

4.2　計算機実験 2

WCEP と極大クリーク列挙の比較を行う．ベンチマークとして tweet グラフと RTN グラフを用いて自然言語処理におけるトピック抽出を行う．ただし，極大クリーク列挙はデータ研磨と呼ばれる操作を行った後のグラフに対して実行する．データ研磨とは，共有する隣接頂点の比率が高い 2 頂点間にのみ辺を結ぶという前処理である．データ研磨を実行することで似たような極大クリークが出力することが少なくなる．WCEP を解くために実行するアルゴリズムは 4.1 の結果から，現実世界のの情報をもとにして作られたグラフに対して性能が高い Add/Swap 探索とする．各パラメータは $d = 0.5$,$T = 7$,$L = 100$, 制限時間を 60 秒とした．グラフの大きさを鑑みて，twitter グラフは $k = 10$，RTN グラフは $k = 5$ とした．

表 2 ～ 表 5 に実験結果を示す．極大クリーク列挙よりも WCEP の方が，データの概要が把握できるような単語の集団が獲得できている．さらに WCEP はクリークの重みも出力できるので，各集団同士の注目度や重要度の違いを推測することができる．

5.　あとがき

大量の非構造データに対する数理モデルとそのモデルに対するクリーク群抽出問題という最適化問題を提案した．また，WCEP に対する 3 つの手法の計算機実験を行った．大規模なグラフのような解の遷移先が大きいグラフに対し

表 2　twitter グラフに対する WCEP で得られるクリーク (クリークの数:10)

重み	クリーク
19762	{ ミサイル発射 速報 落下 自国 }
5823	{ ミサイル 失敗 戦争 秒読み 驚く 浮上 最新情報 }
5470	{ 開発 継続 示唆 高官 }
4540	{ すぎる する トランプ 戦争 秒読み 最新情報 }
3076	{ 外交 ローマ法王 解決 問題 }
・・・	

表 3　twitter グラフに対するデータ研磨＆極大クリーク列挙で得られるクリーク (クリークの数:31055)

クリーク
{ 合わせる 邪魔 問題 ドナルド・トランプ }
{ スマホ 邪魔 外遊 ドナルド・トランプ }
{ スマホ 邪魔 森友 ドナルド・トランプ }
{ 国交 岸田 対話 ドナルド・トランプ }
{ 大切 国交 協力 }
・・・

表 4　RTN グラフに対する WCEP で得られるクリーク (クリークの数:5)

重み	クリーク
122	{attack afghanistan united_states bin_laden taliban }
41	{attack plane tuesday airliner united_states hijack people pentagon world_trade_ctr new_york washington}
13	{united_states leader world taliban washington pres_bush people afghanistan campaign terrorism american bin_laden strike country support group attack force nation war military terrorist percent target}
9	{state washington people thursday anthrax airport worker official united_states attack authority pres_bush wednesday new_york friday american time police government day fbi}
9	{monday week attack saturday country united_states pres_bush official taliban sunday bin_laden pakistan people afghanistan government report washington pentagon city man kabul day time afghan}

表 5　RTN グラフに対するデータ研磨＆極大クリーク列挙で得られるクリーク (クリークの数:450)

クリーク
{chairman united_states}
{price attack pres_bush united_states}
{ambassador government washington attack reporter united_states}
{head people new_york united_states}
{head people reporter united_states}
・・・

てはMN/TSが有効だった．現実世界のデータから作られるようなグラフや比較的小規模なグラフなどの解の遷移先が少ないグラフに対してはAdd/Swap探索が有効だった．さらに，極大クリーク列挙よりもWCEPの方が，Tweetやニュース記事から作られたグラフから上手く元のデータの概要を表すようなトピックを抽出できることを確認した．

今後はより多くの数理モデルとの比較実験を行い，我々が提案した数理モデルの有用性を示していきたい．

参考文献

[1] Wu, Q., Hao, J.-K. and Glover, F.: Multi-neighborhood tabu search for the maximum weight clique problem, *Annals of Operations Research*, Vol. 196, No. 1, pp. 611–634 (2012).
[2] Avis, D. and Imamura, T.: A list heuristic for vertex cover, *Operations Research Letters*, Vol. 35, No. 2, pp. 201–204 (2007).
[3] Challenge, T. S. D. I.: Clique Benchmark Instances, https://turing.cs.hbg.psu.edu/txn131/clique.html (1992-1993).
[4] Xu, K.: Benchmarks with Hidden Optimum Solutions for Graph Problems (Maximum Clique, Maximum Independent Set, Minimum Vertex Cover and Vertex Coloring), http://www.nlsde.buaa.edu.cn/ kexu/benchmarks/graph-benchmarks.htm.
[5] Batagelj, V.: Reuters terror news network, http://vlado.fmf.uni-lj.si/pub/networks/data/CRA/terror.htm (2011).