1P2-Ib-4in

劣モジュラ最適化に基づいたグラフ系列のクラスタリング

Clustering a Graph Sequence based on Submodular Function Optimization

岸本 卓也 Takuya KISHIMOTO　猪口 明博 Akihiro INOKUCHI　河原 吉伸 Yoshinobu KAWAHARA　鷲尾 隆 Takashi WASHIO

大阪大学 産業科学研究所
The Institute of Scientific and Industrial Research, Osaka University

There are many real-world applications suitable to model objects by using graph sequences. For example, a human network is represented by a weighted graph where each human and each relationship between two humans correspond to a vertex and a weighted edge, respectively. If some relationships change in the human network, weights edges in the graph also change, resulting in a sequence of graphs. In this paper, we propose a method for clustering vertices in a graph sequence based on the submodular function optimization to discover changes of communities in it. In addition, we evaluate performance of the proposed method using artificial datasets.

1. はじめに

情報技術の発展により，膨大な量のデータを蓄積することが可能となった．しかし，日々肥大化するデータは人間の理解力を超えたため，有益な情報が含まれていてもそのままでは理解できなくなっている．そこで，この膨大なデータから有益な情報を発見するため，近年データマイニングに関する研究は非常に注目され，盛んに研究されている．中でもクラスタリングは，教師無し学習手法であるため，カテゴリが未知のデータを分類するのに有用である．例えば，人間関係ネットワークのクラスタリングを行うことで，隠れたコミュニティの変化を発見することができ，マーケティング戦略などへ役立てることができると考えられる．

本研究が対象とするグラフ系列マイニングでは，グラフ系列を部分グラフ系列に分割する．例えば，人間関係ネットワークにおいて，人をグラフの頂点，人と人の関係をグラフの辺で表し，その親密度に応じて辺を重み付けすると，ある時点の人間関係ネットワークを重み付きグラフにより表現することができる．さらに時間の経過と共にその構造が変化する人間関係ネットワークは，重み付きグラフの系列として表すことが可能である．このグラフ系列をクラスタリングすることにより，グラフ系列に隠れた部分グラフの変化を発見することが期待される．

本稿では，劣モジュラ関数最適化に基づいたグラフ系列のクラスタリング手法を提案した後，人工データに対して提案手法を適用し，その結果について考察を行う．

2. グラフ系列のクラスタリング

2.1 重み付きグラフ系列

V を頂点集合，E を辺集合，$w : E \to \mathbf{R}$ を辺から正の実数への関数とする．グラフのクラスタリング問題 [Von Luxburg 07] では一つの重み付きグラフ $G = (V, E, w)$ を扱うが，本稿ではグラフの時間変化を考えるために時刻 t の重み付きグラフを $G^{(t)} = (V, E, w^t)$ と表す．さらに，時間順に並べた T ステップのグラフのリスト $G^{(1)}, \ldots, G^{(T)}$ を重み付きグラフ系列と呼び，図 1 のようなグラフ系列扱う．ただし，本稿が対象とするグラフ系列は以下の仮定を満たすものとする．

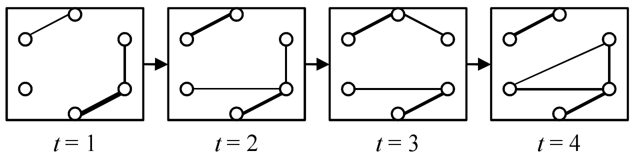

図 1　4 ステップ重み付きグラフ系列の例．辺の重みは太さで表されており，太い辺ほど大きな重み付けがされていることを表す．

仮定 1: 連続する 2 つのグラフ $G^{(t)}$ と $G^{(t+1)}$ $(1 \leq t < T)$ において対応する辺の重みの変化はわずかである．

例えば，人間関係ネットワークでは一度に大半の人間が入れ替わることは無く，実世界の多くのグラフ変化はこの仮定を満たしている．仮定 1 よりグラフの辺の重みの変化はわずかであるので，$G^{(t)}$ 及び $G^{(t+1)}$ において次節で定義するクラスタの変化もわずかであると考えることができる．

仮定 2: グラフ $G^{(t)}$ $(1 \leq t \leq T)$ の頂点数 $|V|$ を n とし，グラフ系列の中で n の値は変化しない．

実世界のグラフの頂点数は増減する場合もあるが，本稿では問題の簡単化のためこの仮定をおく．

2.2 グラフ系列のクラスタリング問題

グラフ系列のクラスタリング問題を次のように定義する．

問題 1: グラフ系列 $G^{(1)}, \ldots, G^{(T)}$ とクラスタ数 k が入力として与えられた時，グラフ構造の変化を考慮しコスト関数 $F(P)$ を最適化するクラスタリング解 P を求める．

ただし，$P = \{C_1, C_2, \ldots, C_k\}$ はクラスタ C_i $(1 \leq i \leq k)$ から成る分割であり，$C_i = \left\{C_i^{(1)}, C_i^{(2)}, \ldots, C_i^{(T)}\right\}$ は各時刻のグラフのクラスタ $C_i^{(t)}$ $(1 \leq t \leq T)$ から成り，時刻 t におけるグラフ $G^{(t)}$ の頂点 V は，互いに交わりの無い k 個のクラスタ $C_i^{(t)}$ $(i = 1, \ldots, k)$ に分割される．すなわち，$\bigcup_{i=1,\ldots,k} C_i^{(t)} = V$ であり，$i \neq j$ に対して $C_i^{(t)} \cap C_j^{(t)} = \emptyset$ である．また，$F(P)$ は集合から実数への関数であり，この問題は組み合わせ最適化問題である．

連絡先: 岸本 卓也，大阪大学 産業科学研究所，567-0047 大阪府茨木市美穂ヶ丘 8-1，kishimoto@ar.sanken.osaka-u.ac.jp

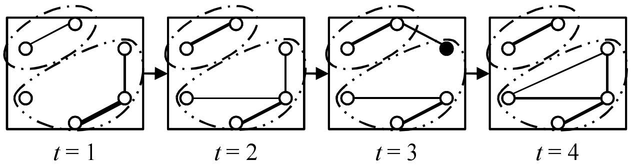

図 2　図 1 のグラフ系列について $k = 2$ で問題 1 を解いた例．時刻 $t = 3$ のグラフの黒い頂点はこのこのグラフのみをクラスタリングすると上のクラスタに分類されるが，前後の対応するクラスタの変化を考慮すると下のクラスタに分類される．

図 1 のグラフ系列について問題 1 を解くと，図 2 のようなクラスタを得る問題について考える．

3. 劣モジュラ関数

集合関数 $f : 2^V \to \mathbf{R}$ が，

$$f(S) + f(T) \geq f(S \cap T) + f(S \cup T)\ (S, T \subseteq V) \tag{1}$$

を満たす時，関数 f は劣モジュラ性を持つ，または f は劣モジュラ関数であるという [塩浦 10]．また，式 (1) は以下の式 (2) と同義である．

$$f(S' \cup \{v\}) - f(S') \geq f(S \cup \{v\}) - f(S)\ (S' \subseteq S \subseteq V, v \notin S) \tag{2}$$

離散領域における劣モジュラ関数最小化は，連続領域における凸関数最小化に対応し [河原 10]，劣モジュラ関数最適化問題の中では比較的簡単に局所解を得ることができ，また，Queyranne のアルゴリズム [Queyranne 95] や Minimum-Norm-Point [Fujishige 11] といった効率的なアルゴリズムが知られている．

4. 提案手法

グラフ系列のクラスタリング問題を解くために必要となる，膨大な組み合わせ演算による計算爆発を避けるため，効率的な計算が可能となる劣モジュラ性を持つコスト関数を定義し，劣モジュラ関数最適化アルゴリズムを利用してクラスタリング解を得る手法を提案する．

4.1 コスト関数

4.1.1 各グラフの最適なクラスタリング

2.2 節で定義した問題に基づいて各時刻のグラフ $G^{(t)}$ を k 個のクラスタに分割するためのコスト関数を $F_1(P)$ とする．$F_1(P)$ はカット関数 $cut(S) = \sum_e \{w(e) \mid e \in E(S, V \setminus S)\}\quad (S \subset V)$ の和で構成し，

$$F_1(P) = \sum_{t=1}^{T} \sum_{i=1}^{k} cut\left(C_i^{(t)}\right)$$

と定義する．ただし，$E(S, V \setminus S)$ は一方の端点が S に含まれ，他方の端点が $V \setminus S$ に含まれる辺の集合である．この関数の値が小さくなる P に分割することで，頂点間の重みが互いに大きい頂点を同一のクラスタにまとめることができる．

4.1.2 連続する 2 グラフ間の変化

仮定 1 に基づいて連続する 2 つのグラフ間 $G^{(t)}$ と $G^{(t+1)}$ で対応するクラスタの変化を考慮するためのコスト関数を $F_2(P)$ とする．$F_2(P)$ は時刻 t 及び $t+1$ のクラスタリング結果を表す指示行列の距離 $dist\left(A_i^{(t)}, A_i^{(t+1)}\right) = \sum_{v \in V} \left|a_{iv}^{(t+1)} - a_{iv}^{(t)}\right|$ で構成し，

$$F_2(P) = \sum_{t=1}^{T-1} \sum_{i=1}^{k} dist\left(A_i^{(t)}, A_i^{(t+1)}\right)$$

と定義する．ただし，$A_i^{(t)} = \left(a_{i1}^{(t)} a_{i2}^{(t)} \ldots a_{ij}^{(t)} \ldots a_{in}^{(t)}\right)$ は $G^{(t)}$ の i 番目のクラスタの要素を表すベクトルであり，v_j を j 番目の頂点 $(1 \leq j \leq n)$ とすると，

$$a_{ij}^{(t)} = \begin{cases} 1 & \text{if } v_j \in C_i^{(t)} \\ 0 & \text{if } v_j \notin C_i^{(t)} \end{cases}$$

である．この関数の値が小さくなる P に分割することで，時刻によって異なるクラスタに分類される頂点を少なくすることができる．

4.1.3 グラフ系列のコスト関数

以上のコスト関数をまとめて，グラフ系列のクラスタリングにおけるコスト関数を $\alpha \geq 0$ をパラメータとして，

$$F(P) = F_1(P) + \alpha F_2(P)$$

と定義する．α を変化させることで，$F_1(P), F_2(P)$ のどちらに重きを置くかを調整することが可能である．

4.2 コスト関数の劣モジュラ性

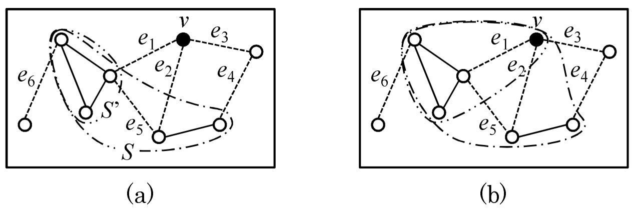

図 3　集合 S'，S，頂点 v 及び周辺の辺の関係

4.1 節で定義したコスト関数 $F(P)$ が劣モジュラ関数で構成されることを示す．まず，$F_1(P)$ の cut について，図 3(a) のようなグラフで S', S, v を定めた時，

$$\begin{aligned} cut(S') &= w(e_1) + w(e_5) + w(e_6) \\ cut(S' + \{v\}) &= w(e_2) + w(e_3) + w(e_5) + w(e_6) \\ cut(S) &= w(e_1) + w(e_2) + w(e_4) + w(e_6) \\ cut(S + \{v\}) &= w(e_3) + w(e_4) + w(e_6) \end{aligned}$$

となるので，

$$\begin{aligned} &\{cut(S' \cup \{v\}) - cut(S')\} - \{cut(S \cup \{v\}) - cut(S)\} \\ &= (-w(e_1) + w(e_2) + w(e_3)) - (-w(e_1) - w(e_2) + w(e_3)) \\ &= 2w(e_2) \geq 0 \end{aligned}$$

となり，式 (2) を満たすので cut は劣モジュラ関数であり，クラスタ C_1 及び $C_2 = V \setminus C_1$ の 2 つに分割するコスト関数 $F'_1(C_1) = F_1(\{C_1, C_2\}) = \sum_{t=1}^{T} \sum_{i=1}^{2} cut\left(C_i^{(t)}\right)$ は劣モジュラ関数である．

次に，$F_2(P)$ についてクラスタ C_1 及び $C_2 = V \setminus C_1$ の 2 つに分割するコスト関数 $F'_2(C_1) = F_2(\{C_1, C_2\}) =$

$\sum_{t=1}^{T-1}\sum_{i=1}^{2} dist\left(A_i^{(t)}, A_i^{(t+1)}\right)$ を考える．指示行列を作成すると S', S に対してそれぞれ以下に示す範囲

$$
\begin{array}{rl}
 & \qquad \overbrace{\overbrace{}^{S'}}^{S}\qquad v \\
A_1^{(t-1)} = & \left(\ 1\ \cdots\ 1\ \mid\ \cdots\ \mid\ 0\ \cdots\ 0\ \right) \\
A_1^{(t)} = & \left(\ 1\ \cdots\ 1\ \mid\ \cdots\ \mid\ a_{1v}^{(t)}\ 0\ \cdots\ 0\ \right) \\
A_1^{(t+1)} = & \left(\ 1\ \cdots\ 1\ \mid\ \cdots\ \mid\ 0\ \cdots\ 0\ \right)
\end{array}
$$

が 1 であるとし，時刻 t のグラフの頂点 v を S' または S に加えた時，つまり $a_{1v}^{(t)}$ を 0 から 1 に変えた時，式 (2) を満たすかどうかを確認する．$F_2'(S' \cup \{v\})$ と $F_2'(S')$ で値が変わるのは，$a_{iv}^{(t-1)}, a_{iv}^{(t)}, a_{iv}^{(t+1)}$ の間だけであるため，

$$
\begin{aligned}
&F_2'(S' \cup \{v\}) - F_2'(S') \\
&= \sum_{i=1}^{2}\left\{\left|a_{iv}^{(t)} - a_{iv}^{(t-1)}\right| + \left|a_{iv}^{(t+1)} - a_{iv}^{(t)}\right|\right\} \\
&= \left|a_{1v}^{(t)} - a_{1v}^{(t-1)}\right| + \left|a_{1v}^{(t+1)} - a_{1v}^{(t)}\right| + \left|a_{2v}^{(t)} - a_{2v}^{(t-1)}\right| + \left|a_{2v}^{(t+1)} - a_{2v}^{(t)}\right| \\
&= 1 + 1 + 1 + 1 = 4
\end{aligned}
$$

となる．$F_2'(S \cup \{v\})$ と $F_2'(S)$ も値が変わるのは，$a_{iv}^{(t-1)}, a_{iv}^{(t)}, a_{iv}^{(t+1)}$ の間だけであるため，S' に関する計算と同様に $F_2'(S \cup \{v\}) - F_2'(S) = 4$ となる．従って，$F_2'(S' \cup \{v\}) - F_2'(S') = F_2'(S \cup \{v\}) - F_2'(S)$ が成り立つため，$F_2'(C_1)$ はモジュラ関数である．

ここで，式 (1) を満たす劣モジュラ関数 $f_1(S)$ 及び $f(S) + f(T) = f(S \cap T) + f(S \cup T)$ を満たすモジュラ関数 $f_2(S)$ の線形和 $f_3(S) = f_1(S) + f_2(S)$ について，

$$
\begin{aligned}
f_3(S) + f_3(T) &= f_1(S) + f_1(T) + f_2(S) + f_2(T) \\
&\geq f_1(S \cap T) + f_1(S \cup T) + f_2(S \cap T) + f_2(S \cup T) \\
&= f_3(S \cap T) + f_3(S \cup T)
\end{aligned}
$$

より，式 (1) を満たすので，$f_3(S)$ は劣モジュラ関数である．

従って，劣モジュラ関数とモジュラ関数の線形和は劣モジュラ関数となるため，関数 $F'(C_1) = F_1'(C_1) + F_2'(C_1)$ は劣モジュラ関数である．$F'(C_1)$ は 2 分割を行うコスト関数であるが，[Narasimhan 05] の Optimal k-clusterings と同様の手順を行うことで k 分割に拡張することが可能である．

5. 評価実験

4. 章で述べた提案手法を MATLAB で実装し，人工データへ適用させて実験を行った．実験は，Intel Xeon CPU W3565 3.20GHz のプロセッサ，12GB のメインメモリ，Windows 7 Enterprise 64bit の OS，MATLAB 7.10.0(R2010a) の環境で行い，計算時間及びエラー率を測定した．ただし，エラー率は $\frac{\text{誤分類された頂点数}}{n \times T}$ により計算した．

5.1 劣モジュラ関数最適化アルゴリズム

本稿では劣モジュラ関数最適化アルゴリズムに Minimum-Norm-Point を用い，MATLAB Toolbox for Submodular Function Optimization (v 2.0)[Krause] において実装されている関数 sfo_min_norm_point を利用した．また，クラスタリングには同じ MATLAB Toolbox の関数 sfo_greedy_splitting を利用しており，この関数では k 分割を行うために 2 分割を繰り返し貪欲的に分割する手法 [Zhao 05] が実装されている．

5.2 人工データ

人工データは次のように作成した．まず，混合ガウス分布の $k = 3$ 個のガウス分布の平均を，原点中心，半径 $r = 4$ の円周上に等間隔に配置した．また，時刻ステップを経る毎に各平均が原点に近づいたり離れたりするようにして混合ガウス分布を生成した．分散 $var = 1.0$，n/k 個の要素から成る各ガウス分布が一つのクラスタに対応し，その中心は初期位相 $\varphi = 0$，振幅 $A = 1.0$，周期 $p = \pi/4$ で正弦的に振動する．なお，頂点数 n の既定値は 24 個である．このようにして生成した各時刻の頂点座標を用いて，2 頂点間のユークリッド距離の逆数を辺の重みとし，時刻ステップ数 $T = 8$ のグラフ系列を生成した．

5.3 実験結果

以下の全ての実験において α の値は $\{22.4, 25.1, 28.2, 31.6\}$ に変化させて計測した．

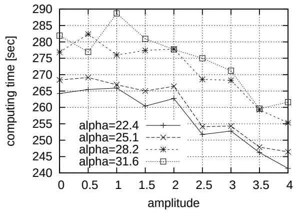

図 4　振幅に対する計算時間の変化

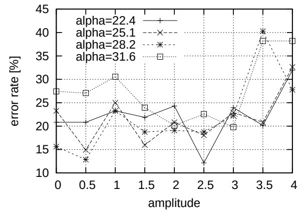

図 5　振幅に対するエラー率の変化

図 4，5 は振幅 A を 0 から 4 まで増加させた時の計算時間及びエラー率である．半径 $r = 4$ であるため，振幅が 4 に近づくに従ってクラスタが原点に近づいた時に他のクラスタと交差する可能性が高くなり，クラスタが交差するグラフの個数も増加するため，エラー率が上昇している．

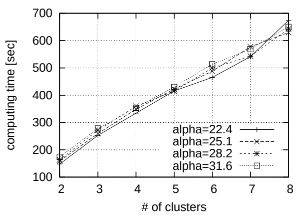

図 6　クラスタ数に対する計算時間の変化

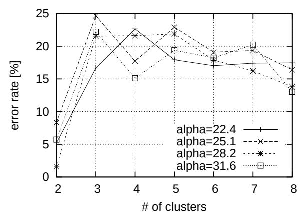

図 7　クラスタ数に対するエラー率の変化

図 6，7 はクラスタ数 k を 2 から 8 まで増加させた時の計算時間及びエラー率である．貪欲的に k 分割する手法を用いているため，計算時間はクラスタ数に対して線形に増加している．エラー率は全体的にはクラスタ数を増加させるに従い減少する傾向となっているが，$k = 2$ において最小となっている．これは，$k = 2$ においては誤った分類がなされた時，分類される他のクラスタの候補が一つしか無いためである．

図 8，9 は半径 r を 0.1 から 15.9 まで増加させた時の計算時間及びエラー率である．振幅 $A = 1$ であるため，$r = 1$ を超えると異なるクラスタ同士が交差する可能性が下がり，分割しやすくなるため誤分類される頂点が減り，エラー率が減少し始めている．

図 10，11 は時刻ステップ数 T を 4 から 20 まで増加させた時の計算時間及びエラー率である．グラフ系列のクラスタリングでは，nT 個の頂点のクラスタリングを試行する必要がある上，プログラム内部のループ処理も T に比例してループ回数が増加

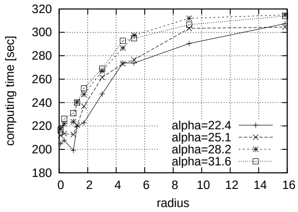

図 8　半径に対する計算時間の変化

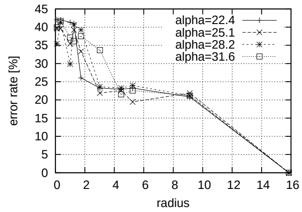

図 9　半径に対するエラー率の変化

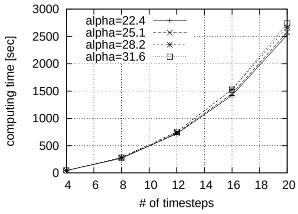

図 10　時刻ステップ数に対する計算時間の変化

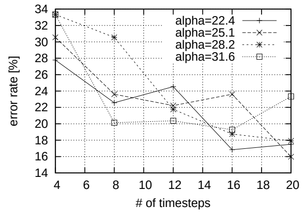

図 11　時刻ステップ数に対するエラー率の変化

するため，計算時間が指数的に増加している．また，人工データの各クラスタの中心は正弦的に振動するため，時刻ステップ数が増えるとクラスタの時間変化を制御するコスト関数 F_2 の影響が大きくなるため，誤分類される頂点が減り，エラー率が減少している．

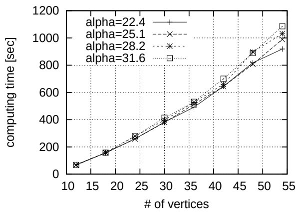

図 12　頂点数に対する計算時間の変化

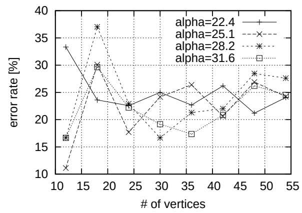

図 13　頂点数に対するエラー率の変化

図 12，13 は頂点数 n を 12 から 54 まで増加させた時の計算時間及びエラー率である．先にも述べたとおり，nT 個の頂点のクラスタリングを試行する必要があるため，n が増加すると計算時間は線形に増加している．一方で頂点数を増やしても他のパラメーターによって決まる頂点の分布やクラスタ同士の交差具合は変化しないため，エラー率はほとんど変わらない．

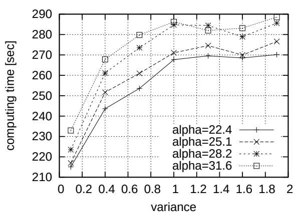

図 14　分散に対する計算時間の変化

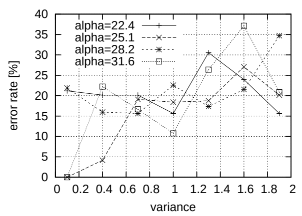

図 15　分散に対するエラー率の変化

図 14，15 は分散 *var* を 0.1 から 1.9 まで増加させた時の計算時間及びエラー率である．分散を増加させると頂点が各クラスタの中心から離れた位置に散在しやすくなり，異なるクラスタ同士が交差する可能性が増えるため分割が難しくなり，エラー率及び計算時間は共に増加している．

以上の全ての結果について α の値の変化による違いに注目すると，計算時間に対しては α の値を変化させても同じような傾向を示しているが，エラー率については時折他と異なる値を示している点がある．これは，全てのパラメーターの組み合わせ毎に最適な α の値が存在するが，その値を求めることはできないため，必ずしも良い α で実験できていないためである．

6.　まとめ

本稿では，劣モジュラ関数最適化に基づいたグラフ系列のクラスタリング手法を提案し，人工データを用いてその評価実験を行った．劣モジュラ関数は局所解を比較的簡単に求めることができ，最適化には既存の効率的なアルゴリズムを利用可能であることから，提案手法はそのコスト関数が劣モジュラ性を持つように定義した．

人工データによる実験において，計算時間は主にクラスタ数，時刻ステップ数，及び頂点数の増加に従って長くなり，特に時刻ステップ数に対しては指数的に増加しており計算時間に与える影響が大きい．従って，今後この改善が課題であり，また，人工データだけではなく実データによる手法の評価を行う必要がある．

参考文献

[Fujishige 11] Fujishige, S. and Isotani, S.: A Submodular Function Minimization Algorithm Based on the Minimum-Norm Base, *Pacific Journal of Optimization*, Vol. 7, pp. 3–17 (2011)

[Krause] Krause, R. A.: Matlab Toolbox for Submodular Function Optimization, `http://www.cs.caltech.edu/~krausea/sfo/`

[Narasimhan 05] Narasimhan, M., Jojic, N., and Bilmes, J.: Q-Clustering, *Advances in Neural Information Processing Systems 18* (2005)

[Queyranne 95] Queyranne, M.: A combinatorial algorithm for minimizing symmetric submodular functions, in *Proceedings of the sixth annual ACM-SIAM symposium on Discrete algorithms*, SODA '95, pp. 98–101, Philadelphia, PA, USA (1995), Society for Industrial and Applied Mathematics

[Von Luxburg 07] Von Luxburg, U.: A tutorial on spectral clustering, *Statistics and Computing*, Vol. 17, No. 4, pp. 395–416 (2007)

[Zhao 05] Zhao, L., Nagamochi, H., and Ibaraki, T.: Greedy splitting algorithms for approximating multiway partition problems, *Math. Programming*, Vol. 102, pp. 102–167 (2005)

[塩浦 10] 塩浦 昭義：劣モジュラ関数の最大化, `http://www.kurims.kyoto-u.ac.jp/~takazawa/coss2010/shioura-1.pdf` (2010)

[河原 10] 河原 吉伸：機械学習における劣モジュラ性の利用, `http://www.nec.co.jp/rd/datamining/project/nec_datamining_seminar8.pdf` (2010)