# 劣モジュラ関数最小化と劣モジュラ流問題のアルゴリズムに関する研究

数理情報学専攻 48-096222　立木 泰樹
指導教員　室田 一雄 教授

## 1　問題設定

本研究では，劣モジュラ関数最小化問題と最小費用劣モジュラ流問題のアルゴリズムを扱った．本論文で扱う劣モジュラ関数とは，有限集合 $V = \{1, 2, \ldots, n\}$ 上の整数値集合関数 $\rho : 2^V \to \mathbb{Z}$ で，任意の $X, Y \subseteq V$ に対して劣モジュラ不等式

$$\rho(X) + \rho(Y) \geq \rho(X \cup Y) + \rho(X \cap Y)$$

を満たすものであり，劣モジュラ関数の最小化問題とは，与えられた劣モジュラ関数 $\rho$ に対して，関数値評価オラクルのもとで $\min_X \rho(X)$ の最小値を達成する $X$ を見つける問題である．最小費用劣モジュラ流問題とは，与えられた有向グラフ $G = (V, E)$ と枝容量の上下限 $\overline{c}, \underline{c} \in \mathbb{Z}^E$，枝コスト $\gamma \in \mathbb{R}^E$ と劣モジュラ関数 $\rho$ に対して，

$$\begin{aligned} &\min. \langle \gamma, \varphi \rangle \\ &\text{s.t. } \underline{c} \leq \varphi \leq \overline{c} \\ &\quad \partial_G \varphi \in \mathrm{B}(\rho) \end{aligned}$$

を達成する $\varphi \in \mathbb{Z}^E$ を見つける問題である．ここで，$\mathrm{B}(\rho)$ を

$$\mathrm{B}(\rho) = \{x \in \mathbb{R}^V \mid \forall X \subseteq V.\ x(X) \leq \rho(X) \text{ かつ } x(V) = \rho(v)\}$$

で定義し，$x(X) = \sum_{v \in X} x(v)$ と書く．

最小費用劣モジュラ流問題での $\rho$ について，仮定すべきオラクルとして自然なものは，関数値オラクルと交換容量オラクルが考えられる．関数値オラクルとは，$X \subseteq V$ に対して $\rho(X)$ を返すオラクルであり，交換容量オラクルとは，$x \in \mathrm{B}(\rho)$, $u, v \in V$ に対して

$$\sigma(x, u, v) = \max\{\alpha \geq 0 \mid x + \alpha(\vec{v} - \vec{u}) \in \mathrm{B}(\rho)\}$$

を返すオラクルである．ここで，$\vec{u} \in \mathbb{R}^V$ は第 $u$ 成分のみが 1 で，他成分が 0 のベクトルとする．関数値オラクルと交換容量オラクルは，多項式時間の意味では等価であり，多項式回のオラクル呼び出しで互いを構成できる．

最小費用劣モジュラ流アルゴリズムについての既存研究では，歴史的な経緯から，$\rho$ のオラクルとして交換容量オラクルを仮定したものがほとんど全てであり，関数値オラクルのもとでのアルゴリズムはあまり研究されていない．

本研究では，劣モジュラ最小化問題と，その双対問題を道具とし，関数値オラクルのもとでの最小費用劣モジュラ流を扱った．

## 2　方針

交換容量オラクルのもとでの既存のアルゴリズムを基に，オラクルの置き換えを施して，関数値オラクルのもとでのアルゴリズムを構成する．既知の方法として，劣モジュラ関数最小化問題を使うものがある．交換容量 $\sigma(x, u, v)$ は，別な表現として

$$\sigma(x, u, v) = \min\{\rho(X) - x(X) \mid v \in X \subseteq V \setminus u\}$$

と書けることが知られているため，劣モジュラ関数最小化アルゴリズムを使って右辺を計算することで，関数値オラクルだけから交換容量の値が得られる．

最小費用劣モジュラ流問題のアルゴリズムの標準的な設計法として，残余・交換ネットワーク上での増大路に沿って解の改善を繰り返し，最適解への到達を目指すという手法がある．

残余・交換ネットワーク $G_{x,\varphi}$ は，最小費用流などの古典的なネットワーク流問題で使われる残余ネットワークの拡張であり，以下のように定義される:

$$\begin{aligned} E_\varphi &= \{e \in E \mid \varphi(e) < \overline{c}\}, \\ \overline{E}_\varphi &= \{\overline{e} \in \overline{E} \mid \varphi(e) > \underline{c}\}, \\ E_x &= \{(u, v) \in V \times V \mid \sigma(x, u, v) > 0\}, \\ E_{x,\varphi} &= E_\varphi \cup E_\varphi \cup E_x, \\ G_{x,\varphi} &= (V, E_{x,\varphi}). \end{aligned}$$

交換容量オラクルのもとでの既存のアルゴリズムでは，各反復ごとに $n^2$ 回だけ交換容量オラクルを使って $G_{x,\varphi}$ を構成し，$G_{x,\varphi}$ 上で解の改善を行っている．

しかし，既存のアルゴリズムの中には，交換容量が 0 か正かだけを使うものがあり，それらは交換容量の値を無視しても，計算量見積りを悪化させることなく実行できる．例えば Fujishige [2] のアルゴリズムや，Iwata [3] のアルゴリズムはこれに該当する．

また，本論文で示した定理によれば，劣モジュラ関数最小化問題の双対問題の最適解が，ある付加情報と共に得られているならば，$n^2$ ヶ所全ての交換容量の符号情報を一度に計算できる．Schrijver [6] や Orlin [5] の劣モジュラ最小化アルゴリズムでは，双対問題の解が付加情報と共に得られる．

ここから，劣モジュラ関数最小化問題の双対問題を使うことにより，$G_{x,\varphi}$ の枝全体を一度に構成する手法を提案する．

## 3 結果

提案手法を使い，交換容量オラクルのもとでの既存のアルゴリズムを基に関数値オラクルのもとでのアルゴリズムを構成した．

劣モジュラ最小化の双対問題として Orlin [5] のアルゴリズムを使い，Fujishige [2] のアルゴリズム中の残余・交換ネットワークの構成を劣モジュラ最小化の双対問題で置き換え，これをサブルーチンとして Iwata [3] のアルゴリズムを書き換えた．

このアルゴリズムは，$O(n^7 \cdot \log U)$ 回の関数値オラクルの呼び出しで最適解を出力する．一方，関数値オラクルのもとでの既知のアルゴリズムである Fleischer-Iwata [1] は $O(mn^5 \cdot \log U)$ 回の関数値オラクル呼び出しで実行できる．ここで，$U$ は枝容量の最大絶対値である．$m = \Theta(n^2)$ である密なグラフに対しては，この 2 つのアルゴリズムは同じ計算量見積りであり，多重枝により枝数がさらに多いグラフについては，提案手法は，より高速なアルゴリズムを与えている．

## 4 今後の課題

提案手法についての改良や拡張の余地として，以下の 3 つが考えられる．

- 各反復での残余・交換グラフの構成について，1 回前の反復での値を再利用すると，部分問題として解くべき劣モジュラ最小化の双対問題を，サイズの小さなものに分解できる．また，この部分問題については，最適解の値自体は既知で最適解の表現を求めるだけでよく，しかも，最適解から 1 しか離れていない点の表現が既知である．この特殊性を使って，部分問題を高速に解く方法は研究の余地がある．
- Iwata [3] のアルゴリズムでは枝コストとして分離凸関数を扱い，Iwata-Moriguchi-Murota [4] のアルゴリズムでは M 凸関数を取り入れた目的関数について，同様のアルゴリズムを適用している．提案手法によるアルゴリズムについても目的関数を拡張し，M 凸関数などに拡張することが考えられる．
- 交換容量オラクルのもとでのアルゴリズムには，費用スケーリングや負閉路消去法など，別な枠組みのアルゴリズムがある．他の枠組みのアルゴリズムについても関数値オラクルへの置き換えは工夫の余地がある．

## 参考文献


[1] Lisa Fleischer and Satoru Iwata. Improved algorithms for submodular function minimization and submodular flow. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 107–116. ACM, 2000.

[2] Satoru Fujishige. Algorithms for solving the independent-flow problems. *Journal of the Operations Research Society of Japan*, 21(2), 1978.

[3] Satoru Iwata. A capacity scaling algorithm for convex cost submodular flows. *Mathematical Programming*, 76:299–308, 1996.

[4] Satoru Iwata, Satoko Moriguchi, and Kazuo Murota. A capacity scaling algorithm for m-convex submodular flow. *Mathematical Programming*, 103(1):181–202, 2005.

[5] James B. Orlin. A faster strongly polynomial time algorithm for submodular function minimization. *Mathematical Programming*, 118(2):237–251, 2009.

[6] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal Combinatorial Theory, Series B*, 80(2):346–355, 2000.