日本オペレーションズ・リサーチ学会
2004年秋季研究発表会

劣モジュラ多面体内直線探索問題に対する強多項式時間アルゴリズム

申請中　東京大学　永野 清仁　NAGANO Kiyohito

1　はじめに

本研究では劣モジュラ多面体内直線探索問題に対し，Megiddo[3] のパラメトリックサーチ法（組込み法とも呼ばれる）の枠組みを用いることで，強多項式時間アルゴリズムを与えた．劣モジュラ関数最小化の完全に組合せ的な強多項式時間アルゴリズム [1] をベースとして解法を構成しているところがポイントである．劣モジュラ多面体内直線探索問題は，最小カット問題を含んでおり，また劣モジュラ関数の理論における交換容量を求める問題の一般化としても捉えることができる．

2　問題設定

V を有限集合とする ($|V| = n$)．集合関数 $f : 2^V \to \mathbf{R}$ を劣モジュラ関数とする．つまり任意の $X, Y \subseteq V$ で

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

が成立する．ベクトル $x \in \mathbf{R}^V$ の $u \in V$ に関する成分を $x(u)$ で表すものとする．$f(\emptyset) = 0$ を満たすような劣モジュラ関数 f に対し，劣モジュラ多面体 $\mathbf{P}(f)$ を

$$\mathbf{P}(f) = \{\, x \in \mathbf{R}^V \,|\, x(X) \leq f(X)\,(\forall X \in 2^V)\,\}$$

のように定義する．ここで $x(X) = \sum_{u \in X} x(u)$ である．以下の問題を考える．

問題 1　劣モジュラ多面体内直線探索問題
入力：　劣モジュラ関数 $f : 2^V \to \mathbf{R}$,
　　初期点 $x_0 \in \mathbf{P}(f)$，方向ベクトル $a \in \mathbf{R}^V$.
出力：　$t^* = \max\{\, t \in \mathbf{R} \,|\, x_0 + t\,a \in \mathbf{P}(f)\}$.

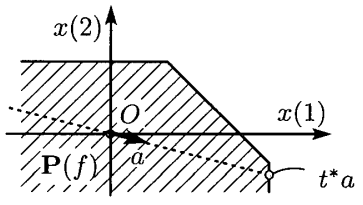

図 1: 劣モジュラ多面体内直線探索問題 ($n = 2$)

図 1 は $n = 2$ の場合の例である．もし $a \leq \mathbf{0}$ であればこの問題は解をもたないので $a \not\leq \mathbf{0}$ と仮定する．また一般性を失うことなく $x_0 = \mathbf{0}$ と仮定できる (各 $X \subseteq V$ について $f(X) := f(X) - x_0(X)$ と置きなおせばよい)．

3　Newton 法

まず，問題 1 に対するシンプルな解法である Newton 法 (Dinkelbach 法とも呼ばれる) について述べる．

関数 $h : \mathbf{R} \to \mathbf{R}$ を

$$h(t) = \min_{X \subseteq V} \{f(X) - t a(X)\}$$

で定義する．関数 h は凹関数で，$h(0) = 0$ を満たす．任意の $t \in \mathbf{R}$ について，$f - t\,a$ が劣モジュラ関数であることに注意して，関数値 $h(t)$ は劣モジュラ関数最小化アルゴリズム ([2] など) を用いて求められる．いま

$$t^* = \max\{\, t \in \mathbf{R} \,|\, h(t) = 0\,\}$$

であることから以下のような解法が構成できる．

Newton 法
S 0：　$t_1 \geq t^*$ を満たす t_1 をとってくる．$i := 1$.
S 1：　$f - t_i\,a$ を最小化する．最適解を X_i とする．
　　$(h(t_i) = f(X_i) - t_i a(X_i))$
S 2：　$h(t_i) = 0$ なら停止 $(t^* = t_i)$．$h(t_i) < 0$ なら
　　$t_{i+1} := f(X_i)/a(X_i)$，$i := i + 1$ として S 1 へ．

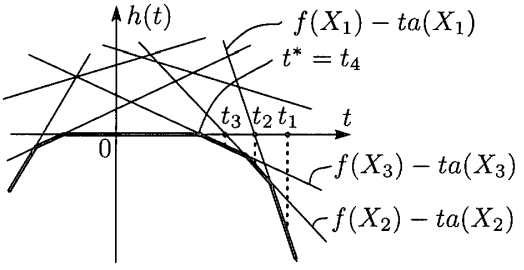

図 2: Newton 法

図 2 は Newton 法の実行例である．既存の結果を用いることで，Newton 法が問題 1 に対する弱多項式時間アルゴリズムであることは容易に導かれる．また $a \in \mathbf{R}^V_+ \backslash \{\mathbf{0}\}$ の場合は Newton 法の反復回数は高々 $n + 1$ であることが知られている．Newton 法の 1 回の反復は n の多項式で抑えられる時間で実行できるので，$a \in \mathbf{R}^V_+ \backslash \{\mathbf{0}\}$ であれば Newton 法は問題 1 に対する強多項式時間アルゴリズムとして構成可能である．しかし一般の方向ベクトル $a \in \mathbf{R}^V$ について Newton 法が問題 1 に対する強多項式時間アルゴリズムとなるかどうかはわかっていない．

4 強多項式時間アルゴリズム

問題 1 に対し，Megiddo [3] のパラメトリック・サーチ法の枠組みを用いた強多項式時間解法を与える．劣モジュラ関数を最小化する完全に組合せ的な強多項式時間アルゴリズム [1]，つまり演算として足し算・引き算・比較・関数値の呼出のみを用いたもの，をベースに解法を構成していることがポイントである．

最適値 t^* との比較

強多項式時間アルゴリズムを構成する準備として，任意の $t \in \mathbf{R}_+$ と t^* の大小判定を考える (図 3 参照)．

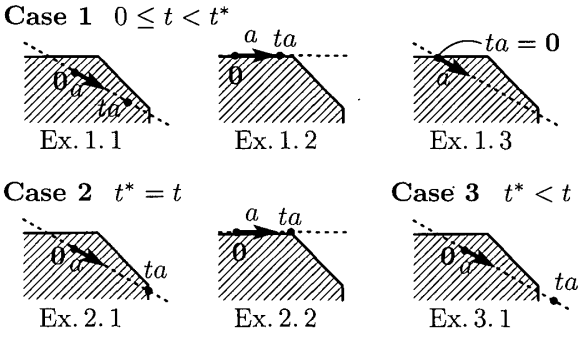

図 3: 最適値 t^* との比較

まず $t^* < t$ かどうかの判定は

$$t^* < t \iff \min_{X \subseteq V} \{f(X) - ta(X)\} < 0$$

から $f - ta$ の最小化により可能である．あとは $t \le t^*$ の条件下で $t^* = t$ or $t < t^*$ の判定を考える．

いま $f(X) = ta(X)$ を満たす集合 $X \subseteq V$ の全体を $\arg\min(f - ta)$ で表す (必ず $\emptyset$ を含む)．図 3 を参考にして，$\max\{\, a(X) \mid X \in \arg\min(f - ta)\}$ の値が正でなら $t^* = t$，そうでなければ $t < t^*$ とわかる．集合族 $\arg\min(f - ta)$ を構成するには，集合として最小な最小化元を必ず出力する劣モジュラ関数最小化アルゴリズム ([2], [1] は少し工夫すればこのようなアルゴリズムとなる) を n 回実行すればよい．また a の最大化は最大流問題に帰着できる．結局 t と t^* の比較は n の多項式で抑えられる時間で実行できるとわかる．

$t^* = t$ が成り立つ場合に $f(X) = ta(X)$ かつ $a(X) > 0$ を満たす集合 X は重要である．n の多項式で抑えられる時間で $t \in \mathbf{R}_+$ と t^* の大小を比較し，$t^* = t$ のときは $f(X) = ta(X)$ かつ $a(X) > 0$ を満たす集合 X を出力するような手続きを COV(t) とする．手続き COV は上記のようにして構成できる．手続き COV における劣モジュラ関数最小化を，完全に組合せ的な強多項式時間アルゴリズム [1] で行うものを手続き L-COV とする．

本研究で与えた強多項式時間アルゴリズムは，簡単にいえば「手続き L-COV の実行中に手続き COV を繰り返し何度も呼び出す」ような構成になっている．

強多項式時間アルゴリズム

t^* の値を知っている状態で L-COV(t^*) を実行すれば，$f(X) = t^* a(X)$ かつ $a(X) > 0$ を満たすような集合 X が出力される．もし仮に t^* の値を知らない状態で L-COV(t^*) を無理やり実行できたとする．これによりやはり $f(X) = t^* a(X)$ かつ $a(X) > 0$ を満たすような集合 X が出力され，$f(X)/a(X)$ によって t^* の値を得ることができる．問題は，いかにして「t^* の値を知らない状態で L-COV(t^*) を無理やり実行」するかであり，これは Megiddo のパラメトリック・サーチ法の枠組みを用いて実現される．以下この概略を示す．

t^* の値を知らない状態で L-COV(t^*) を無理やり実行するときにどのような困難が伴うかについて考察する．L-COV(t^*) において行われる演算はその構成法から足し算・引き算・比較・f の関数値呼出，それと各 $v \in V$ について $t^* a(v)$ の値を得るための n 回の掛け算のみである．つまり ($t^* a(v)$ を得るため以外には)L-COV(t^*) の中においてデータどうしの掛け算や割り算が行われない．よって L-COV(t^*) の実行中に現れるすべての値を t^* の一次式 ($p - q\,t^*$ の形，p, q は既知) で表すことが可能である．すべての値を t^* の一次式で表したとして，各演算について考える．

足し算・引き算については

$$(p_1 - q_1\,t^*) \pm (p_2 - q_2\,t^*) \mapsto (p_1 \pm p_2) - (q_1 \pm q_2)t^*$$

より手間は 2 倍となるが計算時間のオーダは変わらない．

比較演算は足し算・引き算ほど簡単ではない．任意の既知の実数 $p, q \in \mathbf{R}$ について $p - q\,t^*$ の符号を判定できれば比較演算は可能だが，いま t^* の値はわかっていない．COV(0) を実行すれば $t^* = 0$ か $t^* > 0$ か判定できるので $t^* > 0$ と仮定する．もし $pq \le 0$ であれば符号の判定は容易である．$pq > 0$ の場合は符号はすぐにはわからないが，COV(p/q) を実行することで符号を判定することができる．

以上のように，L-COV(t^*) を無理やり実行し，比較演算が現れるたびに (必要ならば) 手続き COV を実行するという，手続き L-COV の中に手続き COV を組込んだようなアルゴリズムは強多項式時間で実行できる．

参考文献

[1] S. Iwata: A fully combinatorial algorithm for submodular function minimization. *Journal of Combinatorial Theory (B)*, **84** (2002), pp. 203–212.

[2] S. Iwata, L. Fleischer, and S. Fujishige: A combinatorial, strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, **48** (2001), pp. 761–777.

[3] N. Megiddo: Combinatorial optimization with rational objective functions. *Mathematics of Operations Research*, **4** (1979), pp. 414–424.