■12 群（電子情報通信基礎）- 2 編（離散数学）

4 章 ネットワーク最適化

（執筆者：永持　仁）［2009 年 4 月 受領］

■概要■

本章では，ネットワークとして枝や点に費用，容量などを表す重みが付された有向あるいは無向グラフを指し，ネットワーク上での最適化問題のうち効率よく解ける代表的な問題として，最小木問題，最短路問題，最大流問題，最小カット問題，最小費用流問題，最大マッチング問題について解説をする．これらの問題は，いずれもグラフ構造にちなむ線形の不等式制約のもとで線形の目的関数を最大化（最小化）させる整数値の未知変数を決定する，いわゆる整数計画問題（integer programming problem）として書き下せる．一般に未知変数が線形の不等式制約を満たす整数値を持ち得るかという問題は NP-困難であるが，未知変数を連続変数としてみなすと，多項式時間で解くことのできる線形計画問題（linear programming problem）が現れる．本章で紹介する問題は，いずれも適切に線形不等式を選択すれば，この線形計画問題が最適解として整数解をもつという性質を有している．この性質のもと，単体法などの線形計画問題を一般的に解く方法を上記の問題に特化した解法も知られているが，本章ではグラフ構造上で直接設計されたアルゴリズムを取り扱う．

【本章の構成】

本章では，最小木問題，最短路問題，最大流問題，最小カット問題，最小費用流問題，最大マッチング問題に対し，問題の定式化と問題を解くアルゴリズムのうち典型的なものについて解説する．

本章を通じて用いる記法を以下にまとめておく．グラフは点集合 $V = \{v_1, v_2, \ldots, v_n\}$, 枝集合 E の組 $G = (V, E)$ で表し， 点の数 $|V|$ を n, 枝の数 $|E|$ を m とする．無向枝はその両端点 u, v の組 $\{u, v\}$ で，有向枝は，その始点 u，終点 v の順序対 (u, v) でも表す．点に対する重みは適当なグラフ変形により枝重みとして取り扱えることが多く，本章では，容量を表す非負実数値の重み $cap(e)$ $(\forall e \in E)$，費用を表す実数値の枝重み $w(e)$ $(\forall e \in E)$ のみを考え，これらを備えたグラフをネットワーク $N = [G, cap, w]$ と書く．$w(e)$ を「枝重み」あるいは「長さ」と呼ぶ．有向（無向）枝 $e = (u, v)$ $(e = \{u, v\})$ の容量 $cap(e)$，長さ $w(e)$ は，$cap(u, v)$, $w(u, v)$ でも表す．グラフ G の点の部分集合 $X, Y \subseteq V$ に対して，$E(X, Y)$ は，G が無向グラフの場合，端点の一方を X 内に，端点の他方を Y 内に持つ無向枝の集合を表す．G が有向グラフの場合，$E(X, Y)$ は，始点を X 内に，終点を Y 内にもつ有向枝の集合を表し，特に，点 $v \in V$ から出ていく枝の集合 $E(\{v\}, V - \{v\})$，入ってくる枝の集合 $E(V - \{v\}, \{v\})$ をそれぞれ $E^+(v)$ と $E^-(v)$ とで表す．

部分グラフ $G' = (V', E')$，あるいはその枝集合 $E' \subseteq E$ に対して含まれる枝 e の重み $w(e)$ の和を $w(G'), w(E')$ と記す．

文献については個々のアルゴリズムの原著ではなく，それらを取りまとめた成書を引用した．

■12 群 - 2 編 - 4 章

4-1 最小木

（執筆者：永持　仁）[2009 年 4 月 受領]

4-1-1 最小木問題

最小木問題（minimum spanning tree problem）とは，連結な無向グラフ $G = (V, E)$ と実数値の枝重み w からなるネットワーク $N = [G, w]$ が与えられたとき，G の全域木（spanning tree）$T \subseteq E$ のなかで枝の重み和 $w(T)$ を最小にするもの（最小木）を選ぶことである．ネットワークの点が地図の地点を表し，枝が可能な通信リンクを表しているとすると，この問題は全地点を連結するために引かれるリンクの長さの総和を最小にすることに対応する．一般には負の枝重みも許したうえで，ある全域木 $T \subseteq E$ が最小木であるための必要十分条件は，任意の補木の枝 $e = \{u, v\} \in E - T$ に対し，その重み $w(e)$ が，T 上で端点 u, v 間に存在するどの枝の重みよりも小さくならないことである．ここで枝重みの役割は 2 本の大小比較を定めているだけであるので，重みによる枝の順位付けだけが最小木を決定する．

4-1-2 最小木アルゴリズム

最小木問題を解く方法としてクラスカル（Kruskal）法とプリム（Prim）法を紹介する．

クラスカル法は貪欲法に基づいている．すべての枝を重さの昇順に $w(e_1) \le w(e_2) \le \cdots \le w(e_m)$ と整列した後，空の枝集合 $T := \emptyset$ から始め，$e_1, e_2, \ldots, e_m$ の順に，T に加えていき全域木をつくる．ただし，枝 e_i を加える際 $T \cup \{e_i\}$ に e_i を通る閉路が形成される場合には，枝 e_i を T に加えずに破棄する．T が全域木となった時点で最小木が得られる．枝集合の整列はソーティングの問題であり，$m = O(n^2)$ であるので $O(m \log n)$ の手間で行える．閉路形成の判定は，枝集合 T の誘導する連結成分に対して集合合併問題のデータ構造を用いることで，全体で $O(m\alpha(m, n))$ の手間で実行できる[1]．ここで，$\alpha(m, n)$ はアッカーマン関数の逆関数であり，通常ほぼ定数とみなせる．したがって，クラスカル法の計算の手間は $O(m \log n)$ である．

プリム（Prim）法は任意に選んだ始点 s から木を成長させるように重み最小の枝を選択していく．まず，すべての点 $v \in V$ に対し $\ell(v) := w(s, v)$, $p(v) := s$ ($\{s, v\} \in E$ のとき), $\ell(v) := \infty$, $p(v) := null$ ($\{s, v\} \notin E$ のとき) 及び $S := \{s\}$, $T := \emptyset$ と初期化する．そして，$S = V$ となるまで $V - S$ のなかで最小のラベル ℓ をもつ点 v を選び，以下の操作を行えばよい．枝 $\{p(v), v\}$ はカット $E(S, V - S)$ のなかで最小の重みをもっている．この枝 $\{p(v), v\}$ を T に加え，v を S に加えた後，v と $V - S$ との間の各枝 $\{v, u\}$ に対し，$\ell(u) > w(v, u)$ であれば $\ell(u) := w(v, u)$, $p(u) := v$ と更新する．計算終了時点の T が最小木となる．全体の計算の手間は，点集合 $V - S$ をラベル値 ℓ に関するヒープで管理することで $O(m \log n)$ となり，更にフィボナッチヒープ（Fibonacci heap）というデータ構造を使えば $O(m + n \log n)$ にできる[1]．

最小木問題を解く現在最良の計算の手間は $O(m\alpha(m, n))$ である[3]．グラフ G が平面グラフである場合には $O(n)$ の手間で解ける[3]．このほか，手間の期待値が $O(m + n)$ となる確率アルゴリズム[3]や枝重みを整数に限り，数字のビット演算を許したモデルにおける線形の手間のアルゴリズムもある[3]．

有向ネットワークにおいて，始点 s から重み和 $w(T)$ を最小にする有向木 T を求めるには異なるアルゴリズムが必要であるが，$O(m + n \log n)$ の手間で解くことができる[3]．

■12 群 - 2 編 - 4 章

4-2 最短路

（執筆者：永持　仁）[2009 年 4 月 受領]

4-2-1 最短路問題

最短路問題（shortest path problem）とは，枝に長さ w をもつネットワーク $N = [G, w]$ 上の最短の経路を求める問題のことである．2 地点間の総距離あるいは総所要時間を求める実用的な問題への応用のほか，最小費用流問題などほかのネットワーク問題の部分問題としても現れる．与えられた 2 点 u,v に対して，u から v へ至る途中で同じ点を 2 度以上通らないパス P のうちでその枝の重さの和 $w(P)$ が最小であるものを u から v への最短パスという．枝の“長さ”として非負の重みだけでなく負の重みをも考えられる．枝重みの和が負となるいわゆる負の閉路（negative cycle）が存在するとき最短パスを求めることは一般に NP-困難な問題となる．有向ネットワーク N では負閉路が存在しないときに限り，点のポテンシャル（potential）$\pi(v)$（$\forall v \in V$）として，不等式 $w(u,v) + \pi(u) - \pi(v) \geq 0$ $(\forall (u,v) \in E)$ を満たすものが存在する．このようなポテンシャルを実行可能という．

4-2-2 最短路アルゴリズム

有向ネットワーク N においては指定された始点 s からほかの点へ至る最短パスの計算，あるいは負閉路の検出に動的計画法（dynamic programming）が用いられる．この単一始点最短路問題に対し，負の重みの枝が存在するときには，ムーア・ベルマン・フォード（Moore-Bellman-Ford）法を用いて，s からの最短パスか負閉路を見つけることができる．後でポテンシャルを得るため，s を始点とする十分重い枝を適当に加え s からどの点へも到達可能であるとする．各点 $v \in V$ に始点 s から点 v へ至るパスで計算中に調べたもののうち最小の重みを保持させるラベル $\ell(v)$ を用意し，まずラベル ℓ の初期化 $\ell(s) := 0$, $\ell(u) := \infty$ $(\forall u \in V - \{s\})$ を行う．そして，次の手続きを $n-1$ 回反復する．各枝 $(v,u) \in E$ に対して $\ell(v) > \ell(u) + w(u,v)$ なら更新操作 $\ell(v) := \ell(u) + w(u,v)$, $p(v) := u$ を行う．この方法は全体で $O(nm)$ の手間で実行することができる．$p(v)$ の役目は，v に至る最短パスにおいて一つ手前の点 u を記録することである．計算終了後，枝集合 $E^* = \{(u,v) \in E \mid u = p(v)\}$ が有向木でなければ，E^* には負閉路が含まれている．一方で，E^* が有向木であれば，これに沿って s からほかの点への最短パスが得られ，$\pi(v) := \ell(v)$ $(\forall v \in V)$ とおくと，π は実行可能ポテンシャルとなり，負閉路が存在しないことも分かる．ここで，実行可能ポテンシャルを使って簡約費用（reduced cost）$w_\pi(u,v) := w(u,v) + \pi(u) - \pi(v)$ $(\forall (u,v) \in E)$ を新たな重みとするネットワーク $N_\pi = [G, w_\pi]$ を考えると，簡約費用は常に非負であり，2 点間 u,v のパスの重さの変化 $\pi(u) - \pi(v)$ は一定なので，どの点を始点とする最短パスの集合も変化しない．

枝重みがすべて非負であればダイクストラ（Dijkstra）法を用いることができる．ラベル ℓ の初期化の後，ℓ の値が最小である未処理点 u を選び処理済みとし，u を始点，未処理点を終点とする各枝 $(u,v) \in E^+(u)$ を走査し，もし $\ell(v) > \ell(u) + w(u,v)$ なら更新操作 $\ell(v) := \ell(u) + w(u,v)$, $p(v) := u$ を行う．これを未処理点が存在する限り実行する．求めたい最短パスの終点 t が指定されている場合，t が処理済みとなった時点で計算を終了することもできる．全体の計算の手間は，未処理点の集合をラベル値 ℓ に関するヒープで管理することで $O(m \log n)$ となり，

更にフィボナッチヒープ（Fibonacci heap）というデータ構造を使えば $O(m + n \log n)$ にできる[1]．このほかの結果として，枝重みが非負整数の場合には w_{max} を最大の枝重みとしたとき，$O(n \log\log w_{max})$ や $O(n \log\log n)$ などの計算の手間で解ける[3]．平面有向ネットワークの場合には $O(n + m)$ の手間のアルゴリズムが存在する[3]．

全点間の最短パスを求めるには，始点の選択を変えて単一始点最短路問題を n 回解けばよい．負の重みの枝があっても 1 回目にムーア・ベルマン・フォード法を適用すれば，後の $n-1$ 回は簡約費用に対してダイクストラ法を使うことで，全体の計算は $O(mn+n^2 \log n)$ の手間ですむ．ほかにも全点間の最短パスを求める簡便な方法としてフロイド・ウォーシャル（Floyd-Warshall）法がある．未知変数 $\ell_k(u,v)$ $(\forall u, v \in V, k = 1, \ldots, n)$ を “点 $\{v_1, \ldots, v_k\}$ のみを通過して点 u から点 v に至る最短パスの長さ”と定義すると，これらは次の関係式に基づいて $O(n^3)$ の手間で計算することができる．$\ell_0(u,v) := w(u,v)$ $(\forall u, v \in V)$; $\ell_k(u,v) := \min\{\ell_{k-1}(u,v), \ell_{k-1}(u,v_k) + \ell_{k-1}(v_k,v)\}$ $(\forall u, v \in V,\ k = 1, \ldots, n)$．負閉路が存在しても，計算中 $\ell_k(u,u) < 0$ となる k, 点 u が見つかった時点でその存在が確認できる．現在，全点間の最短パスは $O(mn + n^2 \log\log n)$ や $O(n^3 \sqrt{\log\log n}/\log n)$ の手間で求められる[3]．

無向ネットワーク $N = [G, w]$ の場合，負の枝重みがなければ，各無向枝を 2 本の逆向きの有向枝におきかえればダイクストラ法で解くことができる．更に枝重みが非負整数である場合には線形の計算の手間のアルゴリズムが存在する[3]．負の重みの枝をもつ場合には，負閉路の検出や 2 点間の最短パスを求める問題は最小重み完全マッチングに変換するなどして多項式の計算の手間で解くことができる[2, 3]．

■12 群 - 2 編 - 4 章

4-3 最大流

(執筆者：永持　仁)[2009 年 4 月 受領]

4-3-1 最大流問題

最大流問題（maximum flow problem）とは，有向グラフ $G=(V,E)$ の各枝 e に容量として非負実数 $cap(e)$ が付与されたネットワーク $N=[G,cap]$ 及び入口の点 $s\in V$，出口の点 $t\in V$ が与えられたとき，各枝上での流量が容量以下である定常的な流れの中で入口からの流入量（=出口への流出量）を最大にするものを求めることである．入口，出口以外の各点ではそこに流入する流量の和とそこから流出する流量の和が釣り合うことが求められる．すなわち，各枝 $e\in E$ 上の流量を $f(e)$ と表したとき，最大流問題は，以下の流量保存式 (4･1)，容量制約式 (4･2) を満たす f を入口 s から出口 t への流れと呼ぶとき，このうち，流量 $\partial f(t)=\sum_{e\in E^-(t)}f(e)-\sum_{e\in E^+(t)}f(e)$ を最大にする流れ（最大流）を求めることである．

$$\sum_{e\in E^-(v)}f(e)-\sum_{e\in E^+(v)}f(e)=0\quad(\forall v\in V-\{s,t\}),\tag{4･1}$$

$$0\le f(e)\le cap(e)\ \ (\forall e\in E).\tag{4･2}$$

流れは，電流，水流，交通流，通信流などを抽象化したモデルに利用することができ，最大流問題は，交通量・通信網の解析などのほか，資源の割当て問題やシステム解析などに広く応用される．

入口 s から出口 t への流れに対する双対な概念である s,t-カットとは，$s\in S, t\in V-S$ であるような点集合 $S\subseteq V$ の与える枝集合 $E(S,V-S)$ である．ネットワーク N から s,t-カットの枝をすべて除去あるいは，それらの枝の向きを反転させると入口 s から出口 t への有向パスがなくなる．$s,t-$ カット の容量を $cap(S,V-S)=\sum_{e\in E(S,V-S)}cap(e)$ により定めると，次の最大流・最小カット定理が成り立つ．$\max\{\partial f(t)\mid$ 入口 s から出口 t への流れ $f\}=\min\{cap(S,V-S)\mid s,t-$ カット $E(S,V-S)\}$．これは，入口 s から出口 t への流れの流量の最大値がこれらの点 s,t を分離するカット容量の最小値に等しいことを意味する．このほか，すべての枝の容量が整数値の場合にはどの枝 e の流れ $f(e)$ も整数となる入口 s から出口 t への最大流が存在することも応用上重要な性質である．

4-3-2 最大流アルゴリズム

最大流を求める二つのタイプのアルゴリズムを紹介する．ネットワーク N 上の枝の流れ $f(e)\ (\forall e\in E)$ の集合に対し，各枝 $e=(u,v)\in E$ の逆向きの枝 $e_{\rm rev}$ の集合 $E_{\rm rev}$ を加え，枝容量を $cap_f(e)=cap(e)-f(e)\ (\forall e\in E)$, $cap_f(e_{\rm rev})=f(e)\ (\forall e_{\rm rev}\in E_{\rm rev})$ と定めた有向ネットワーク $N^f=[G^*=(V,E\cup E_{\rm rev}),cap_f]$ を流れ f による残余ネットワーク（residual network）という．

(1) 増加パスに基づくアルゴリズム

残余ネットワークにおいて入口から出口へ流れを増やせるパスを見つけ，そのパスに沿ってできる限り流量を増やすことを続けることで最大流を得ることができる．入口 s から出

口 t への流れ f に対する残余ネットワーク N^f において，点 s から点 t への有向パス P の容量を $\delta(P) = \min\{cap_f(e) \mid e \in E(P)\}$ により定める．ここで，$\delta(P) > 0$ であるとき，P を増加パス（augmenting path）という．入口 s から出口 t へ P に沿って流量 $\delta(P)$ を流すことができる．すなわち，流れ f を次のように更新する．$f(e) := f(e) + \delta(P)\ (\forall e \in E(P) \cap E)$, $f(e) := f(e) - \delta(P)\ (\forall e_{\mathrm{rev}} \in E(P) \cap E_{\mathrm{rev}})$．この結果，流量 $\partial f(t)$ は $\delta(P)$ だけ増加する．

アルゴリズムは，流量 0 の流れとして $f(e) := 0\ (\forall e \in E)$ と初期化する．そして残余ネットワーク N^f において増加パス P を求め，それに沿って入口 s から出口 t へ流量 $\delta(P)$ を流し，流れ f を更新することを，残余ネットワークに増加パスがなくなるまで繰り返す．最後の残余ネットワーク N^f において，入口 s から到達できる点の集合 S は $cap(S, V-S) = \partial f(t)$ を満たすので，最大流・最小カット定理より得られた流れ f は最大流であり，$E(S, V-S)$ は最小容量の s,t-カットである．常に増加パスの中で通過枝数の最小のものを選べば，最悪の場合でも $O(mn^2)$ の手間で計算が終了する[2, 3]．更に増加パスの選択の仕方に工夫を加えることなどで計算の手間の上界を $O(mn \log n)$ に減らすこともできる[2, 3]．

(2) 前流れに基づくアルゴリズム

ネットワーク N 上の枝の流れ $f(e),\ e \in E$ の集合に対し，$\partial f(v) = \sum_{e=(u,v)\in E} f(e) - \sum_{e=(v,u)\in E} f(e)\ (\forall v \in V)$ と定めるとき，容量制約及び $\partial f(v) \geq 0\ (\forall v \in V - \{s\})$ を満たす $f(e),\ e \in E$ を前流れ（preflow）といい，$\partial f(v) > 0$ である点 $v \in V - \{t\}$ を過剰点（excess vertex）という．残余ネットワーク N^f が増加パスをもたないという性質を保ちながら，流れ f を修正し，過剰点をなくすことができればやはり最大流が得られる．

距離ラベル $d(v) \geq 0\ (\forall v \in V)$ は，$d(s) = n,\ d(t) = 0$ かつ，残余容量 $cap_f(e)$ が正であるどの枝 $e = (v, v')$ に対しても $d(v) \leq d(v') + 1$ が成り立つとき有効（valid）であるという．残余ネットワーク N^f が有効な距離ラベル d をもつとき増加パスは存在しない．

過剰点 v を始点とする枝 $e = (v, v')$ は $cap_f(e) > 0$ かつ $d(v) = d(v') + 1$ であるとき使用可能枝（eligible edge）と呼び，この枝 $e = (v, v')$ を通して流量 $\delta(e) = \min\{cap_f(e), \partial f(v)\}$ だけ流れを v から v' へ送る操作をプッシュと呼び，push(e) と表す．すなわち，前流れ f を次のように更新する．$f(e) := f(e) + \delta(e)$, $\partial f(v) := \partial f(v) - \delta(e)$, $cap_f(e) := cap_f(e) - \delta(e)$, $cap_f(e_{\mathrm{rev}}) := cap_f(e_{\mathrm{rev}}) + \delta(e)$．この結果，距離ラベルの有効性が保たれる．一方，過剰点 v を始点とする枝に使用可能枝はないが，$cap_f(e) > 0$ である枝 e があるとき，v の距離ラベルを $d(v) := \min\{d(v') + 1 \mid cap_f(e) > 0, e = (v, v') \in E \cup E_{\mathrm{rev}}\}$ に更新しても距離ラベルの有効性が保たれる．この操作を再ラベルと呼び，relabel(v) と表す．

アルゴリズムは，まず前流れを $f(e) := cap(e)\ (\forall e \in E^+(s))$, $f(e) := 0\ (\forall e \in E - E^+(s))$ により，距離ラベルを $d(s) := n,\ d(v) := 0\ (\forall v \in V - \{s\})$ により初期化する．そして，過剰点が存在する限り以下の手続きを繰り返す．過剰点のうち最も初期に蓄えられた点 v を選ぶ．v を始点とする使用可能枝 $e = (v, v')$ が存在するなら，$\partial f(v) > 0$ である限り，push(e) を実行する．これにより v' が新たな過剰点になることがある．v を始点とする使用可能枝 e が存在しなくなったとき $\partial f(v) > 0$ であれば relabel(v) を施しておく．このアルゴリズムにおいて距離ラベルはその有効性が保たれつつ単調増加していき，$O(n^3)$ の手間で過剰点がなくなり計算が終了することが解析できる[2]．更に，動的木のデータ構造を使うことなどで $O(mn \log(n^2/m))$ の手間に減らすことができる[2, 3]．

■12 群 - 2 編 - 4 章

4-4 最小カット問題

（執筆者：永持　仁）[2009 年 4 月 受領]

枝容量をもつ無向ネットワーク $N=[G,cap]$ の最小カット問題とは，カット $E(S,V-S)$ の中でカット容量 $cap(S,V-S)=\sum_{e\in E(S,V-S)} cap(e)$ を最小にするもの（最小カット）を求めることである．点集合 V の空でない真部分集合 X は，空でない X の任意の真部分集合 Y に対して $cap(Y,V-Y)>cap(X,V-X)$ が成り立つとき，N の極値点集合（extreme vertex set）と呼ばれる．極値点集合の中には最小カット $E(X,V-X)$ を与えるものが必ず存在し，すべての極値点集合の族はネットワークの一つの階層的クラスタ構造を決定するもので，ネットワークの連結性に関する問題に応用をもつ．最大流アルゴリズムにより 2 点を分離する最小カットが計算できることを使うと最大流アルゴリズムを多項式回適用すれば，すべての最小カット，極値点集合を求めることができるが，本節では，流れを使わない簡便なアルゴリズムを紹介する．

4-4-1 最大隣接順序を用いた最小カットの計算

無向ネットワーク $N=[G,cap]$ において異なる 2 点 $u,v\in V$ を分離するカットの最小容量 $\min\{cap(X,V-X)\mid u\in X, v\in V-X\}$ を $\lambda(u,v)$ と記す．$\lambda(u,v)=cap(\{v\},V-\{v\})$ を満たす点の順序対 (u,v) をペンダント対（pendent pair）と呼ぶ．ペンダント対 (u,v) を分離するカットの最小容量値は $cap(\{v\},V-\{v\})$ であることが分かるので，u,v を 1 点に縮約してペンダント対を分離しないカットのみを残す．ペンダント対 (u,v) を求めて縮約する操作をネットワークが 1 点になるまで繰り返し，調べたペンダント対のカット容量のなかで最小の値を選べば，元のネットワークの最小カットの値となる．ペンダント対を求めるには，任意の点を v_1 として選び，選択された点 $v_1,\ldots,v_{i-1}$ との間に最も多くの枝容量 $cap(\{v_1,v_2,\ldots,v_{i-1}\},v)$ をもつ点 $v\in V-\{v_1,v_2,\ldots,v_{i-1}\}$ を v_i として選ぶことを続ければ，最後の 2 点対 (v_{n-1},v_n) がペンダント対となることが分かっている[4]．このネットワークの点の順序づけ $(v_1,v_2,\ldots,v_n)$ を最大隣接順序（maximum adjacency ordering）という．この順序を求めるには，点ラベル $\ell(u):=0\ (\forall u\in V)$ と初期化した後，未処理の点がある限り次の手続き以下を反復すればよい．未処理の点のなかで最大のラベル $\ell(u)$ をもつ点 $v\in V-\{v_1,v_2,\ldots,v_{i-1}\}$ を選び，$v_i:=v$ とおき処理済みとし，v_i と未処理点 $u\in V-\{v_1,v_2,\ldots,v_i\}$ をつなぐ各枝 $e=\{v_i,u\}$ に対し，$\ell(u):=\ell(u)+cap(\{v_i\},\{u\})$ と更新し，枝ラベル $f(e):=\ell(u)$ をつける．この手続きはダイクストラ法と同様に，$O(m+n\log n)$ の手間で実行できるので，ペンダント対を繰り返し縮約するアルゴリズムにより $O(nm+n^2\log n)$ の手間で最小カットを求めることができる．ここで，任意の枝 $\{u,v\}\in E$ に対して $\lambda(u,v)\geq f(e)$ が成り立つことが示せるので，この不等式を使うと次の高速な実装が得られる．最小カット値の上界値を $\overline{\lambda}:=\min\{cap(\{v\},V-\{v\})\mid v\in V\}$ と初期化した後，以下の手続きをネットワークが 1 点になるまで繰り返せば最後の $\overline{\lambda}$ が元のネットワークの最小カットの値を与える．最大隣接順序を計算し，得られた枝ラベル f の値が $\overline{\lambda}$ 以上である枝の集合が誘導するネットワークの各連結成分 C に対して，C を 1 点 v_C に縮約し，上界値の更新 $\overline{\lambda}:=\min\{\overline{\lambda},cap(\{v_C\},V'-\{v_C\})\}$ を行う．ここで，V' は C を 1 点 v_C に縮約した後のネットワークの点集合を表す．

4-4-2 最小次数順序を用いた極値点集合の計算

無向ネットワーク $N = [G, cap]$ のすべての極値点集合族を $\mathcal{X}(N)$ と記す．任意の点 $v \in V$ の単一要素の集合 $\{v\}$ は常に極値点集合であり，自明な極値点集合と呼ばれる．どの二つの極値点集合 $X, Y \in \mathcal{X}(N)$ に対しても，$X \cap Y \neq \emptyset, |X| \geq |Y|$ ならば $X \supseteq Y$ であり，$|\mathcal{X}(N)| \leq 2n-2$ が成り立つ．ネットワーク N の 2 点対 $\{u, v\}$ は，$|\{u, v\} \cap X| = 1$ である非自明な極値点集合 $X \in \mathcal{X}(G)$ をもたないとき，フラット対（flat pair）と呼ばれる．最小容量次数の点を選択してはネットワークから取り除くという操作を繰り返して残る最後の 2 点対がもとのネットワークのフラット対となる性質がある[4]．この点の削除順を最小次数順序（minimum degree ordering）と呼ぶ．最小次数順序は，ダイクストラ法と同様に，$O(m + n \log n)$ の手間で求めることができる．フラット対の 2 点 u と v を単一点 z に縮約しても $\{u\}, \{v\}$ 以外の極値点集合は失われないという性質に基づくと，フラット対の縮約操作を繰り返し行い，N のすべての極値点集合を求めるアルゴリズムを得ることができる．まず，集合族を $\mathcal{X} := \{\{u\} \mid \forall u \in V\}$ と初期化した後，以下の手続きをネットワークが 1 点となるまで反復する．フラット対 $\{u, v\}$ の 2 点 u, v を単一点 z に縮約し，点 z に縮約されているすべての V の点の集合 X_z に対し $\mathcal{X} := \mathcal{X} \cup \{X_z\}$ とする．最後に，得られた族 $\mathcal{X}$ から極値点集合でないものを除く操作 $\mathcal{X}(N) := \mathcal{X} - \{X \in \mathcal{X} \mid Y \subset X$ かつ $cap(Y, V-Y) \leq cap(X, V-X)$ である $Y \in \mathcal{X}$ が存在 $\}$ を施せば，すべての極値点集合を $O(nm + n^2 \log n)$ の手間で得ることができる．

ペンダント対, フラット対の縮約に基づくアルゴリズムは, 対象がハイパーグラフ（hypergraph）での最小カット，極値点集合を含む更に一般的な対称劣モジュラ集合関数上での最小カット，極値点集合の計算にも拡張されている[4]．

■12 群 - 2 編 - 4 章

4-5 最小費用流

（執筆者：永持　仁）[2009 年 4 月 受領]

4-5-1 最小費用流問題

最小費用流問題（minimum cost flow problem）とは，各枝 e に容量として非負実数 $cap(e)$ 及び単位量流れ当たりの費用として実数 $w(e)$ が付与された有向ネットワーク $N=[G,cap,w]$ において，流量保存式及び容量制約式を満たす流れ $f(e)\,(\forall e\in E)$ のうち最小費用のもの（最小費用流）を求めることである．すべての枝の容量が非負整数値で与えられたとき，その最大値を $c_{\max}$ とし，すべての枝の費用が整数値で与えられたとき，その絶対値の最大値を $w_{\max}$ と記す．

本節では，流量保存式の設定の仕方として，相異なる 2 点 s,t 及び入口 s から出口 t への流量として非負実数 q が指定されているものを取り扱う．すなわち，$\partial f(v)=\sum_{e\in E^-(v)} f(e)-\sum_{e\in E^+(v)} f(e)\,(\forall v\in V)$ と定めたとき，流量保存式として，$\partial f(v)=0\,(\forall v\in V-\{s,t\}),\ \partial f(s)=-q,\ \partial f(t)=q$ を考え，この流量保存式と容量制約式 (4･2) を満たす流れ $f(e)\,(\forall e\in E)$ のなかで，費用 $\sum_{e\in E} w(e)f(e)$ を最小にする問題を取り扱う．入口 s から出口 t へ流量 q の流れが存在するかは最大流問題を解くことで検証でき，以下ではそのような流れが存在すると仮定する．q 及びすべての枝の容量が整数値の場合にはどの枝 e の流れ $f(e)$ も整数となる最小費用流が存在する．

最小費用流問題にはいくつか異なる定式化の表現がある．入口 s から出口 t への流量を最大とする流れのうちで費用を最小にしたいときは一度最大流問題を解き，その最大流量を q とおけばよい．一般に，複数個の流入点，流出点を表現するために，各点 $v\in V$ の需要量 $b(v)$ として総和が 0 になる実数値が与えられたとき，$\partial f(v)=b(v)\,(\forall v\in V)$ を流量保存式と設定することもある．これも新たに入口 s，出口 t をネットワークに追加し，s から $b(v)<0$ なるすべての点へ，$b(v)>0$ なるすべての点から t へそれぞれ容量 $|b(v)|$，費用 0 の有向枝を追加することで s,t 間の最小費用流問題に変換できる．逆に，s,t 間の最小費用流問題は，容量 q 及び十分小さい費用をもつ有向枝 (t,s) を追加することで流量保存式 $\partial f(v)=0\,(\forall v\in V)$ を満たす最小費用の流れ（最小費用循環流）を求める設定にも変換できる．一方で，各点で需要量を導入する定式化では容量制約式を流量保存式に取り込むかたちでこれを消し去ることもできる．具体的には，各有向枝 $e=(u,v)\in E$ を 3 本の枝容量 ∞ の有向枝 $e_1=(u,u_e)$, $e_2=(v_e,u_e)$, $e_3=(v_e,v)$ に取り換えればよい．ここで，u_e,v_e はこの枝 e のために新たに導入した点であり，需要量を $b(u_e)=cap(e),\ b(v_e)=-cap(e)$ とし，3 本の枝費用は $w(e_1)=w(e)$, $w(e_2)=w(e_3)=0$ と定める．

最小費用流問題は，特殊な場合として最短路問題，最大流問題，2 部グラフの最大重みマッチング問題を含んでいる．最小費用流問題は実際の問題においても広範囲の応用をもち，物資の輸送経路，スケジュール作成などの解決に数多く適用されている．

4-5-2 最小費用流アルゴリズム

ネットワーク $N=[G=(V,E),cap,w]$ 上の枝の流れ $f(e)\,(\forall e\in E)$ の集合に対し，各枝 $e=(u,v)\in E$ の逆向きの枝 e_{rev} の集合 E_{rev} を加え，枝容量，枝費用を $cap_f(e)=cap(e)-f(e)$,

$w_f(e) = w(e)\ (\forall e \in E), cap_f(e_{\rm rev}) = f(e), w_f(e) = -w(e)\ (\forall e_{\rm rev} \in E_{\rm rev})$ と定めた有向ネットワーク $N^f = [G^* = (V, E \cup E_{\rm rev}), cap_f, w_f]$ を残余ネットワーク（residual network）という．残余ネットワーク N^f において有向パス（有向閉路）P の容量を $\delta(P) = \min\{cap_f(e) \mid e \in E(P)\}$ で，P の費用を $w_f(P) = \sum_{e \in E(P)} w_f(e)$ により定める．有向パス（有向閉路）P に沿って流量 $\delta(P)$ だけを流れを送ることができる．すなわち，流れ f を次のように更新する．$f(e) := f(e) + \delta(P)\ (\forall e \in E(P) \cap E)$, $f(e) := f(e) - \delta(P)\ (\forall e_{\rm rev} \in E(P) \cap E_{\rm rev})$．この結果，費用は $w_f(P)\delta(P)$ だけ増加する．

特に，$\delta(P) > 0$ かつ $w_f(P) < 0$ である有向閉路 P を負の閉路（negative cycle）という．ある指定された流量 q の流れのなかで流れ f が最小費用流であるための必要十分条件は，N^f に負閉路が存在しないことである．

最小費用流問題を解く手間が n と m のみの多項式で抑えられるアルゴリズムを得るにはやや込み入った議論が必要になる．この種の手間で現在最善のものは $O((m + n\log n)m\log m)$ である[2, 3]．以下では二つの基本的なアルゴリズムのみを紹介する．

（1）最小費用増加パスに基づくアルゴリズム

一つ目のアルゴリズムは，流れの流量を目標値 q に達するまで費用の最小性を達成しながら順々に増加させている方法である．初期化として流量 0 の流れ $f(e) := 0\ (\forall e \in E)$ を設定する．このとき N^f には負閉路は存在しないものとする．以後，流量が q に達するまで，以下の手続きを反復する．N^f において入口 s から出口 t への有向パス P で $\delta(P) > 0$ であるもののうち最小費用のものを求め，この有向パス P に沿って流量 $\delta(P)$ だけ流す．手続きの各反復後の流れ f はその流量 $q' = \partial f(t)$ と同じ流量をもつ s から t への流れのうち費用最小のものであることが帰納的に示される．N^f において最小費用の有向パスを求めることは $cap_f(e) = 0$ なる有向枝 e をすべて削除すれば最短路問題となる．各反復後，この最短路問題には負の重みの枝も現れ得るが，前の反復において入口 s から他の各点 v に至る最短路長 $\ell(v)$ が計算されていれば，次の反復では，$w_f(u,u')$ の代わりに点のポテンシャル $\pi(v) := \ell(v)$ による簡約費用 $w_\pi(u,u') := w_f(u,u') + \pi(u) - \pi(u') \geq 0$ にダイクストラ法を適用することができる．すべての枝容量及び指定流量 q が整数であるときこの方法の全体の手間は $O((m + n\log n)mc_{\max})$ である．

（2）負閉路解消アルゴリズム

入口 s から出口 t への流量 q の流れ f をはじめに用意し，残余ネットワーク N^f に負閉路が存在しなくなるように流量 $\partial f(t)$ を q に保ちつつ，流れ f を修正していくことができれば最小費用流が得られる．もし，N^f に負閉路が存在すれば，ムーア・ベルマン・フォード（Moore-Bellman-Ford）法により $O(nm)$ の手間でその一つを求めることができ，得られた負閉路 P に沿って流量 $\delta(P)$ だけ流れを送る．この結果，流れの費用が $\delta(P)w_f(P)$ だけ減少し，得られた流れ f の残余ネットワーク N^f において P はもはや負閉路ではない．すべての枝容量，枝費用，指定流量 q が整数のとき，$O(mc_{\max}w_{\max})$ 回負閉路を解消すれば最小費用流が得られ，この方法の全体の手間は $O(nm^2c_{\max}w_{\max})$ である．負閉路 P のうち平均長 $w_f(P)/|E(P)|$ を最小にするものは動的計画法により $O(nm)$ の手間で求めることができ，常に最小平均長の負閉路を解消していけば，枝重みの整数性を仮定しなくても $O(m^2n\log n)$ 回負閉路を解消すれば最小費用流が得られることが示されている[3]．

■12 群 - 2 編 - 4 章

4-6 最大マッチング

（執筆者：永持　仁）[2009 年 4 月 受領]

4-6-1 最大マッチング問題

最大マッチング問題（maximum matching problem）とは，無向グラフ $G = (V, E)$ が与えられたとき，マッチング（matching）と呼ばれるどの 2 本の枝も端点を共有しない枝部分集合 $M \subseteq E$ でサイズ（枝数）が最大となるもの（最大マッチング）を求めることである．マッチング M のある枝の端点になっている点をマッチ点，そうでない点を未マッチ点と呼ぶ．各枝 e に実数値重み $w(e)$ が与えられているとき，含まれる枝の重み和 $w(M)$ を最大（最小）にするマッチング M を最大（最小）重みマッチングという．

グラフ G の点集合 V が二つの互いに素な集合 V_1, V_2 に分割（すなわち，$V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$）され，E の枝がすべて V_1 の 1 点と V_2 の 1 点を結ぶようなグラフを 2 部グラフ（bibartite graph）といい，$(V_1, V_2; E)$ で表す．2 部グラフのマッチングは次の割当問題（assignment problem）の実行可能解に対応する．割当問題とは，n_1 人と n_2 個の仕事があり，i 番目の人が仕事 j をこなすことにより得られる利得が $w_{i,j}$ で与えられているとき，各人に一つずつ相違なる仕事を割り当てて，利得の全体での和を最大にしようという問題のことである．

マッチング M に含まれている枝と含まれない枝を交互に通る単純なパスを交互パス（alternating path）と呼ぶ．特に，マッチされていない 2 点を結ぶ交互パスは，パス上の枝集合においてマッチング枝とマッチングでない枝の役割を入れ替えるとマッチングのサイズを一つ増大させることができるので，増大パスという．G が増大パスをもたないマッチングは最大であることが示せる．与えられたグラフの最大マッチングを求めるには，空のマッチングから始めて，増大パスを見つけてマッチングのサイズを一つずつ増やすことを，増大パスが存在しなくなるまで繰り返すことができればよい．

4-6-2 2 部グラフにおけるマッチング問題

2 部グラフ上のマッチング問題は，ネットワークフロー問題に帰着させて解くことができる．2 部グラフ $G = (V_1, V_2; E)$ に対して，新たに 2 点 s, t を加え，s から V_1 の各点に容量 1 の有向枝を，V_2 の各点から t に容量 1 の有向枝を付け，E の各枝を V_1 の点から V_2 の点へ向きをつけた容量 1 の有向枝に置き換えたネットワーク N を考えると，G における最大マッチング問題は，N 上でどの枝 e の流れ $f(e)$ も整数である s から t への最大流を求める問題と等価になる．最大流アルゴリズムにおける増加パスがマッチングに対する増大パスに対応し，長さの短い増加パスに基づく最大流アルゴリズムにより 2 部グラフの最大マッチングは $O(\sqrt{n}m)$ の手間で求まる[2, 3]．現在この手間は $O(n\sqrt{nm/\log n})$，$O(m\sqrt{n}\log(n^2/m)/\log n)$ の手間に下げられている[3]．また，2 部グラフ上の最小重みマッチング問題は，上記の帰着において，E の枝重みはそのままとし，s, t に接続する枝の重みを 0 とおくことで，s から t への最小費用流問題と等価となる．

4-6-3 一般グラフにおける最大マッチングアルゴリズム

2 部グラフの場合と異なり，一般のグラフの場合は，未マッチ点から交互パスを延ばしていく際，奇数長の閉路を形成し別の未マッチ点にうまく到達できなくなることがある．増大パスを失わないよう奇数長閉路を処理するエドモンズ（Edmonds）法を説明する．

グラフ G のマッチング M に関する花（blossom）とは G の奇数長 $2k+1 \geq 3$ の閉路で，閉路上に $k \geq 1$ 本の M のマッチング枝を含むものである．花の含む唯一の未マッチ点をこの花の基点といい，基点とある未マッチ点を結ぶ長さ偶数の交互パスがこの花のほかの点を通らずに取れるとき，この花を偶の花という．偶の花 B の点の集合を 1 点 v_B に縮約しても，元のグラフに増大パスが存在すれば，縮約後のグラフ G' に残っている M のマッチング枝の集合 M' に関する増大パス P' が存在することがいえる．そのような P' から G における増大パス P を得ることは，必要なら花 B 内で基点を移動させることで容易に行える．

現在のマッチング M に関する未マッチ点 u^1 を選び，u^1 を始点とする増大パスは以下の手続き Search(u^1) により発見できる．ただし，そのような増大パスがないときには，u^1 を始点とする偶数長の交互パスにより到達できる偶の花を繰り返し縮約したグラフ G^1 及び G^1 において u^1 からの交互パスにより到達できる点の集合 W^1 が出力される．

Search(u^1) ステップ 1：$v_0 := u^1$, $P = (v_0)$, $k := 0$ と初期化する．

ステップ 2：現在の交互パス $P = (v_0, v_1, \ldots, v_{2k})$（$P$ の点はすべて未処理）に対して，P の終点 v_{2k} に未走査の枝が接続していないときは，v_{2k} を処理済みとし，v_{2k} に未走査の枝が接続するか，$k = 0$ となるまで $k := k - 1$ を繰り返す．$k = 0$ となれば現在のグラフを G^1，処理済みの点集合を W^1 として出力し終了する．そうでなければステップ 3 へ．

ステップ 3：終点 v_{2k} に接続する未走査の枝 $\{v_{2k}, u\}$ を任意に 1 本選んで以下を実行し，(iv) の場合以外はステップ 2 へ．

(i) u が処理済みか，u が P 上の奇数添字の点 $u = v_{2j+1}$ であるときは枝 $\{v_{2k}, u\}$ を走査済みにする．

(ii) u が P 上の偶数添字の点 $u = v_{2j}$ であるときは，パス P に枝 $\{v_{2k}, v_{2j}\}$ を加えて得られる基点 $u = v_{2j}$ をもつ偶の花 $B = (v_{2j}, v_{2j+1}, \ldots, v_{2k-1}, v_{2k}, v_{2j})$ を 1 点 v_B に縮約したのち，$v_{2j} := v_B$, $k := j$ とする．

(iii) u が P 上にない未処理のマッチ点であるとき，P に枝 $\{v_{2k}, u\}$ と u に接続しているマッチング枝 $\{u, u'\}$ を付け加え，$v_{2k+1} := u$, $v_{2k+2} := u'$, $k := k + 1$ とする．

(iv) u が P 上にない未処理の未マッチ点であるとき $(v_0, v_1, \ldots, v_{2k}, u)$ は増大パスである．現在のグラフには偶の花が繰り返し縮約されている場合があるが，すべての縮約点を順に元の花に展開しながら元のグラフの増大パスを求めて終了する．

Search(u^1) の (iv) において増大パスが求まったら，これを使ってマッチングのサイズを大きくし，未マッチ点からの増大パスの探索を続ける．一方，u^1 を始点とする増大パスが見つからなかったとき，G^1 の構造から，元のグラフに増加パスは未処理の点 $V - W^1$ のみしか通らないことが示される．よって，以後，未処理の未マッチ点 u を選び，Search(u) を適用し，最後まで増大パスが見つからなければその時点でのマッチングは最大である．

現在，最大マッチングは $O(m\sqrt{n}\log(n^2/m)/\log n)$ の手間で求まることが知られている[3)]．最大重みマッチングを求めるアルゴリズムは少し複雑となるが計算の手間が $O(nm + n^2 \log n)$ であるものが知られている[2, 3)]．

■参考文献

1) 浅野孝夫, “情報の構造 上 情報数学セミナー　ネットワークアルゴリズムとデータ構造,” 日本評論社, 1993.
2) 浅野孝夫, “情報の構造 下 情報数学セミナー　ネットワークアルゴリズムとデータ構造,” 日本評論社, 1993.
3) B. コルテ（著）, J. フィーゲン（著）, 浅野孝夫（翻訳）, 平田富夫（翻訳）, 小野孝男（翻訳）, 浅野泰仁（翻訳）, “組合せ最適化-理論とアルゴリズム,” シュプリンガー・フェアラーク東京, 2005.
4) H. Nagamochi and T. Ibaraki, “Algorithmic Aspects of Graph Connectivities,” Encyclopedia of Mathematics and Its Applications, Cambridge University Press, 2008.