5.4 容量スケーリング法

各頂点の需給量が大きい場合，逐次最短路法では増量可能路に沿って流すことのできる量が相対的に小さく，そのためにすべての頂点の超過量を解消するまでにかなりの反復を要することがある．この弱点を克服する方法として，ここでは増量可能路に**十分な量**を流す工夫 — **容量スケーリング**を説明することにしよう．これによって反復回数は大幅に減少し，最悪計算量は $O(nUS(n,m))$ から $O(m \log US(n,m))$ まで改善される[2]．まず，基本となるアイデア — **ビットスケーリング** (bit scaling) から説明しよう．

ビットスケーリング． 与えられた問題 P の最適解を，その近似問題 $\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_r$ を順に解くことで求める方法で，近似解の精度は逐次改善され，もとの問題 P の最適解へと収束する．解くべき近似問題の数 r は問題 P のデータ a のビット長，つまり $r = \lceil \log a \rceil$ 個[3]で，各 P_k では a が $\lceil a/2^{r-k} \rceil$ に丸められる．これは，データ a を r 桁の 2 進数として表したとき，大きい方から k 桁だけを取って用いることに相当する．問題 P_{k+1} を解くときには P_k の最適解を初期解として利用する．

ビットスケーリングの例として，枝容量の最大値が U のネットワーク流問題 P を考えよう．近似されるデータ a として U を用いれば，図 5.1 に示すように P から $\log\lceil U \rceil$ 個の近似問題 P_k が生成されるが，これらには次の性質を観察できる:

性質 5.10. 問題 P_{k+1} の枝容量は P_k の 2 倍プラス 0 か 1 である．

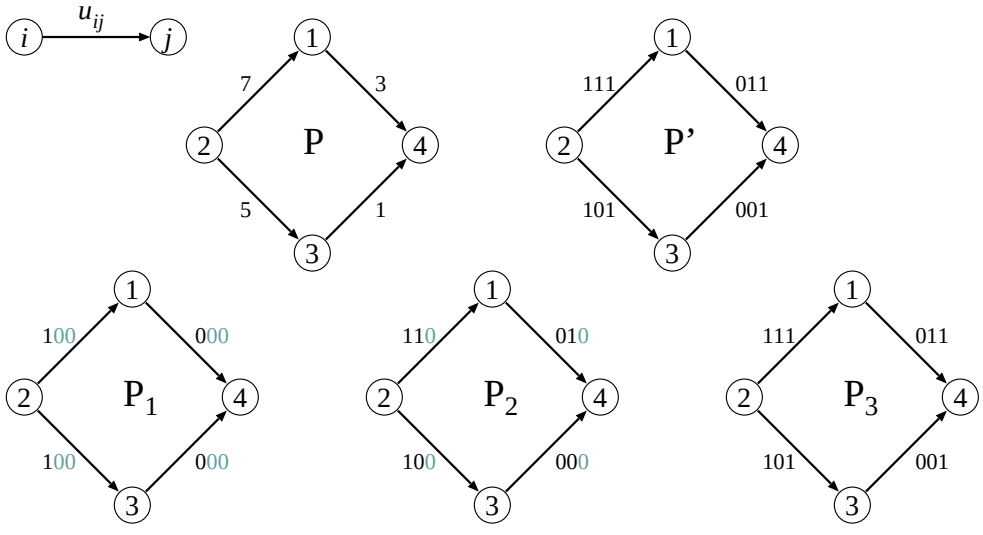

図 5.1: 問題 P とその近似問題たち．

[2] $S(n,m)$ は超点数 n，枝数 m で枝の長さがすべて非負の最短路問題を解くのに必要な計算量．
[3] $\lceil a \rceil$ は a 以上の最小整数．

このビットスケーリングは最大流や最小費用流問題のアルゴリズムの改善にきわめて有効である．その理由をいくつか挙げれば，

(a) 問題 P_1 が簡単に解ける．

(b) 問題 P_k と P_{k+1} の違いがごく小さいので，P_k の最適解から P_{k+1} の最適解は直ちに計算できる．

(c) 近似問題の数 $r = O(\log a)$ はもとの問題 P のサイズの多項式なので，P_k の最適解から P_{k+1} の最適解が多項式で求められれば，全体として多項式時間アルゴリズムとなる．

例えば，先のネットワーク流問題 P が最大流問題であったとしよう．近似問題 P_k の最大流を $\mathbf{x}^k$，その値を v^k とすれば，性質 5.10 から $2\mathbf{x}^k$ は P_{k+1} の実行可能流であり，その値は $2v^k$ である．さらに，P_{k+1} の最大流 $\mathbf{x}^{k+1}$ の値 v^{k+1} は，すべての枝の容量が P_k の 2 倍プラス 1 として，高々 $2v^k + m$ にすぎない．つまり，$\mathbf{x}^{k+1}$ を得るには実行可能流 $2\mathbf{x}^k$ から m 単位の流量を増加させるだけでよく，第 4.2 節のアルゴリズム AUGMENTING_PATH を用いれば $O(m)$ 回の反復で実現できる．このように最大流問題に対してビットスケーリングを用いれば，最短路増量可能路法とは異なる多項式時間アルゴリズムを設計することができる．

最小費用流問題への応用. ビットスケーリングを最小費用流問題に応用する場合，どのデータを近似するかによって複数のバリエーションが考えられる: 枝容量と頂点の需給量

$$U = \max\{\max\{|b_i| \mid i \in N\},\ \max\{u_{ij} \mid (i,j) \in E \text{ かつ } u_{ij} < \infty\}\}$$

を近似する**容量スケーリング法** (capacity scaling algorithm)，枝費用

$$D = \max\{c_{ij} \mid (i,j) \in E\}$$

を近似する**費用スケーリング法** (cost scaling algorithm)，この両者を近似する**２重スケーリング法** (double scaling algorithm)，どの方法によっても (弱) 多項式時間で最適流を求めることができるが，ここでは逐次最短路法を利用する容量スケーリング法を紹介しよう．

容量スケーリング法でも計算途中に保持されるのは被約費用最適性条件（定理 5.5，補題 5.7）を満たす疑似流 $\mathbf{x}$ であり，超過量

$$e(i) = b_i + \sum_{\{j|(j,i)\in E\}} x_{ji} - \sum_{\{j|(i,j)\in E\}} x_{ij}$$

をすべての頂点 i でゼロにするように計算が進められる．各近似問題では枝容量と要点の需給量が近似されるが，これはパラメータ Δ を用いて次のように行われる:

まず，最初の近似問題では $\Delta = 2^{\lfloor \log U \rfloor}$ とし，各増量可能路にはちょうど Δ 単位だけ流す．どの頂点の超過量の絶対値 $|e(i)|$ も Δ 未満になったら，Δ の値を半分にした次の近似問題を解く．

最終的に $\Delta = 1$ となるが，そのときネットワークを流れる疑似流は実行可能流であり，被約費用最適性条件が常に満たされているため，最適流であることが保証される．

パラメータ Δ に対して

$$S(\Delta) = \{i \in V \mid e(i) \geq \Delta\}, \quad T(\Delta) = \{i \in V \mid e(i) \leq -\Delta\}$$

と定義すれば，各増量可能路の始点は $S(\Delta)$ の頂点，終点は $T(\Delta)$ の頂点でなければならない．さらに，その増量可能路に含まれる枝の残余容量も Δ 以上でなければならない．そこで，補助ネットワークが必要となる:

Δ 残余ネットワーク (Δ-residual network). 残余ネットワーク $N(\mathbf{x}) = (G(\mathbf{x}) = (V, E(\mathbf{x})), \mathbf{c}', \mathbf{r})$ と Δ に対し，

$$E(\mathbf{x}, \Delta) = \{(i, j) \in E(\mathbf{x}) \mid r_{ij} \geq \Delta\}$$

を枝集合とする有効グラフ $G(\mathbf{x}, \Delta) = (V, E(\mathbf{x}, \Delta))$ と，費用 $\mathbf{c}' = (c'_{ij})$，容量 $\mathbf{r} = (r_{ij})$ を合わせて Δ 残余ネットワークとよぶ.

このように定義すれば，$N(\mathbf{x}, \Delta)$ 上で $S(\Delta)$ の頂点から $T(\Delta)$ の頂点までの最短路を計算することで，各近似問題における増量可能路を求められることがわかる．アルゴリズムの途中では，$G(\mathbf{x}, \Delta)$ の枝に対して常に被約費用最適性が満たされる; つまり，

$$c'_{ij}(\mathbf{y}) = c'_{ij} - y_i + y_j \geq 0, \quad \forall (i, j) \in E(\mathbf{x}, \Delta), \tag{5.12}$$

を満たす $\mathbf{y}$ が保持される．しかし，$\mathbf{y}$ は枝 $(i, j) \in E(\mathbf{x}) \setminus E(\mathbf{x}, \Delta)$ に対しては必ずしもこの条件を満足しない.

algorithm CAPACITY_SCALING
begin
  $\mathbf{x} := \mathbf{O}$; $\mathbf{y} := \mathbf{O}$; $\Delta := 2^{\lfloor \log U \rfloor}$;
  while $\Delta \geq 1$ **do begin**
    for 各枝 $(i, j) \in E(\mathbf{x})$ **do**
      if $r_{ij} \geq \Delta$ **and** $c'_{ij}(\mathbf{y}) < 0$ **then**
        r_{ij} 単位を枝 (i, j) に流し，x_{ij} と超過量 $e(i)$, $e(j)$ を更新する;
    $S(\Delta) := \{i \in V \mid e(i) > \Delta\}$; $T(\Delta) := \{i \in V \mid e(i) < -\Delta\}$;
    while $S(\Delta) \neq \emptyset$ **and** $T(\Delta) \neq \emptyset$ **do begin**
      頂点 $s \in S(\Delta)$ と $t \in T(\Delta)$ を選ぶ;
      Δ 残余ネットワーク $N(\mathbf{x}, \Delta)$ における頂点 s から他のすべての要点 i への $\mathbf{c}'(\mathbf{y})$ を長さとする最短距離 $d(i)$ を求める;
      各頂点 i に対して $y_i := y_i - d(i)$ とする;

頂点 t への最短路 $\pi \in N(\mathbf{x}, \Delta)$ に沿って流れを Δ だけ増加させ，疑似流 $\mathbf{x}$ を更新する;
各頂点 $i \in V$ における超過量 $e(i)$ を更新し，$S(\Delta)$ と $T(\Delta)$, $N(\mathbf{x}, \Delta)$ を更新する
end;
$\Delta := \Delta/2$
end
end;

容量スケーリング法の正当性と計算量. 外側の **while** ループでは，各反復の終わりに $S(2\Delta) = \emptyset$ か，あるいは $T(2\Delta) = \emptyset$，つまり，

$$e(i) < 2\Delta, \quad \forall i \in V; \quad e(i) > -2\Delta, \quad \forall i \in V$$

のいずれかが満たされることに注意しよう．これは，この時点での正の超過量の総和（= 負の超過量の総和）が $2n\Delta$ でおさえられることを意味する．

次の反復ではパラメータの値 Δ が半分になるので，$\Delta \leq r_{ij} < 2\Delta$ を満たす枝 $(i, j) \in E(\mathbf{x})$ が新たに Δ 残余ネットワーク $N(\mathbf{x}, \Delta)$ に加わる．これらの被約費用が $c'_{ij}(\mathbf{y}) \geq 0$ を満たすかどうかは反復の始めにチェックされ，$c'_{ij}(\mathbf{y}) < 0$ ならば残余容量 r_{ij} に等しい量を流すことで，枝 (i, j) は残余ネットワークと Δ 残余ネットワークから除去される．代わりに逆向きの枝 (j, i) が生じるが，その被約費用は

$$c'_{ji}(\mathbf{y}) = -c'_{ij} + y_i - y_j = -c'_{ij}(\mathbf{y}) > 0$$

となって条件 (5.12) が満たされる．同時に，枝 (i, j) の始点 i と終点 j の超過量 $e(i)$, $e(j)$ も更新されるが，$r_{ij} < 2\Delta$ であることから，その変化量は高々2Δ である．したがって，条件 (5.12) を回復したのちの正の超過量の総和は高々

$$2n\Delta + 2m\Delta = 2(n + m)\Delta.$$

内側の **while** ループは Δ **スケーリング段階** (Δ-scaling phase) とよばれ，総供給量が高々 $2(n + m)\Delta$ の近似問題を解くプロセスに相当する．ここでは毎反復，増量可能路 $\pi \in N(\mathbf{x}, \Delta)$ に沿って頂点 $s \in S(\Delta)$ から $t \in T(\Delta)$ へ Δ 単位が流される．補題 5.7, 5.8 により，増量操作ののちも条件 (5.12) は保たれる．集合 $S(\Delta)$ か $T(\Delta)$ が空となれば，この **while** ループを抜け，Δ の値を半分にしてから次の Δ スケーリング段階に入る．したがって，$O(\log U)$ 回の Δ スケーリング段階を経たのちにパラメータ Δ の値は 1 となる．問題 (5.1) では定数がすべて整数なので，最後の Δ スケーリング段階（つまり，1 スケーリング段階）にはどの頂点の超過量も解消される．このとき，$N(\mathbf{x}, \Delta) = N(\mathbf{x})$，しかも条件 (5.12) が保たれているので，定理 5.5 から $\mathbf{x}$ はもとの問題の最適流である．

各 Δ スケーリング段階に入るときの正の超過量の総和は $2(n + m)\Delta$ でおさえられており，また 1 回の増量操作では頂点 $s \in S(\Delta)$ の超過量が Δ だけ減少する．このことから，各 Δ スケーリング段階が行う増量操作は最大でも $2(n + m)$ 回にすぎない．以上から次の定理が導かれる:

定理 5.11. アルゴリズム CAPACITY_SCALING が要する最悪計算量は $(m \log US(n,m))$ である.

参考文献

最小費用流問題の研究は Ford と Fulkerson によって 1950 年代の後半から始まったが，この節で紹介した閉路削除法は

[1] M. Klein, "A primal method for minimal cost flows with application to the assignment and transportation problems", *Management Science* **14** (1967), 205 – 220.

によるもので，逐次最短路法は

[2] M. Iri, "A new method of solving transportation-network problems", *Journal of the Operations Research Society of Japan* **3** (1960), 27 – 87.

による．残余ネットワークの枝長を非負にすることで逐次最短路法の計算量は著しく改善されるが，これを示したのは

[3] N. Tomizawa, "On some techniques useful for solution of transportation network problems", *Networks* **1** (1972), 173 – 194.

[4] J. Edomonds and R.M. Karp, "Theoretical improvement in algorithmic efficiency for network flow problems", *Journal of ACM* **19** (1972), 248 – 264.

後者には，ビットスケーリングに基づく最初の (弱) 多項式時間アルゴリズムも示されている．またも Edomonds！であるが，彼のアイデアを踏まえ，強多項式時間アルゴリズム — $O(m^4)$ の開発に初めて成功したのは女性研究者の Tardos である:

[5] E. Tardos, "A strongly polynomial minimum cost circulation algorithm", *Combinatorica* **5** (1985), 247 – 255.

その後も緻密な改良が加えられ，現在，最速とされてるのは Orlin による $O((m \log n)(m + n \log n))$ のアルゴリズムである:

[6] J.B. Orlin, "A faster strongly polynomial minimum cost flow algorithm", *Procedings of the 20th ACM Symposium of the Theory of Computing* (1988), 377 – 387.

ただし，最速というのはあくまで最悪計算量での話，経験的な解析によれば単体法のネットワーク版，**ネットワーク単体法** (network simplex algorithm) の方が強力である:

[7] W.H. Cunningham, "A network simplex method", *Mathematical Programming* **11** (1976), 105 – 116.

単体法では最適解に収束しない**巡回** (cycling) 現象の起きる恐れがあるが，これを防ぐためにネットワーク単体法には「あっ!」と驚くような工夫が施されている．その詳細は，また別の機会に紹介しよう.