Implementation of the Network Simplex Algorithm to MATLAB by way of the shortest path problem

Naomichi AOYAMA,* Yoshihide WATANABE** and Toshiaki ITOH***

(Received February 10, 2011)

The main theme of the present paper is an implementation of the Network Simplex Algorithm for solving the minimum cost flow problem to MATLAB. The minimum cost flow problem is the problem for finding the minimum cost flow which satisfies the given capacity conditions and the demand and supply conditions. Although it is known to have a wide application to real fields, there have been known a few algorithms for solving the minimum cost flow problem. The Network Simplex Algorithm is known as the most efficient one, but it is a very complicated algorithm consisting of lots of sub-algorithms. So the implementation of the algorithm is not easy and it has varieties of combinations of sub-algorithms. In the present paper, we provide an implementation of the Network Simplex Algorithm using algorithms for the shortest path problems such as the Dijkstra algorithm or the Floyd-Warshall algorithm efficiently as sub-algorithms. We cannot say that the present implementation is quite efficient but it is easy to understand and requires a little knowledge about algorithms on graph theory.

Key words ： digraph,network, shortest path problem, minimum cost flow, network simplex algorithm,

キーワード ： 有向グラフ，ネットワーク，最短経路問題，最小費用流問題，ネットワーク・シンプレックス法

最短路問題を用いたネットワークシンプレックス法のＭＡＴＬＡＢへの実装

青 山 直 路・渡 邊 芳 英・伊 藤 利 明

1. はじめに

現在の社会では様々なタイプのネットワークが重要な役割を果している．その例として道路網、鉄道網などの交通網や，電力や水道のネットワーク，医療ネットワークや人と人との繋がりを表すもの，また電話回線やインターネット回線などの通信網などがあげられる．このようなネットワークを表すための数学的な枠組みとしてグラフ上のネットワークがある．グラフとは数学的に，頂点集合と，頂点の間を結ぶ辺の集合から構成されるものであり，頂点は道路網であれば，都市やインターチェンジなどを，コンピュータのネットワークであれば個々のコンピュータなどを表し，辺は都市間の道路網やコンピュータを繋ぐインターネット回線と考えられる．グラフ上の各辺に数値が与えられているとき，それをネットワークと呼ぶ．それらの数値は，辺が道路であれば長さを表したり，その辺を通して物を運ぶ際のコストを表したり，通信網であれば単位時間当たりの通信容量などを表す．

* Graduate School of Life and Medical Sciences, Doshisha University, Kyoto
Telephone:0774-65-6302, E-mail:dmj0102@mail4.doshisha.ac.jp
** Department of Mathematical Sciences, Doshisha University
*** Department of Biomedical Engneering, Doshisha University

このようなグラフ上のネットワークにおける最適化問題のなかで最も基本的な問題が最短路問題である．最短路問題とは，各辺の長さが与えられている時，任意の2頂点間の最短路を求める問題で，カーナビなどのナビゲーションシステムに応用されている．最短路問題には，いくつかの有名な解法アルゴリズムが知られており，最短路問題は，それ自体が重要な問題であると同時に，他のネットワーク最適化問題を解くためのパーツとしても重要な役割を果すことが知られている．

本論文の主題は，最小費用流問題であり，その解法アルゴリズムであるネットワークシンプレックス法をMATLABに実装することである．最小費用流問題とは，与えられた各辺に対する容量条件と，頂点に関する需要供給条件を満足する流れのなかで，コストが最小となるものを求める問題である．最小費用流問題の代表的な解法アルゴリズムがネットワークシンプレックス法である．ネットワークシンプレックス法のアルゴリズムは，多岐にわたる様々な部分のアルゴリズムから構成されており，実装は簡単ではない．実装にあたっては途中計算を行うアルゴリズムをどのように実装するかが鍵となる．本論文では，計算効率よりも実装の分かりやすさを重視して，最短路問題を様々な箇所に有効に使うことにより実装を行った．

2. グラフ上のネットワーク

頂点集合 V と辺集合 E からなる有向グラフを $G = (V, E)$ で表す．有向グラフにおけるパス P とは，頂点の列 $P = (u_1, u_2, \cdots, u_n)$ であって，$(u_1, u_2) \in E, (u_2, u_3) \in E, \cdots, (u_{n-1}, u_n) \in E$ となるもののことであり，P を u_1 から u_n へのパスといい，u_1 をパスの始点，u_n をパスの終点という．パスは，それに含まれる頂点がすべて相異なるとき，単純パスと呼ばれる．ただし， 始点と終点が一致することは許し，そのような単純パスを閉路という．有向グラフ $G = (V, E)$ において，グラフ G の辺集合 E 上に実数値関数 $w : E \to \mathbb{R}$ が定義されているとき，G と w をまとめて $N = (G, w)$ と表し，N をグラフ G 上のネットワークと呼ぶ．辺集合上で定義された関数 w を重みと呼ぶ．

3. 最短路問題

3.1 最短路問題とは

有向グラフ $G = (V, E)$ 上の各辺 $e \in E$ に実数の重み $l : E \to \mathbb{R}$ が与えられているネットワークについて考える．$l(e)$ は辺 e の長さ (length) とする．一般的には $l(e)$ は正とは限らないことに注意する．最短路問題とはネットワーク $N = (G, l)$ において，ある与えられた頂点 $u \in V$ を始点とし，ある与えられた頂点 $v \in V$ を終点とするパス P のなかで，パスの長さが最小となるパス (最短路) とそのパスの長さ (最短距離) を求める問題である．この時，パス P の長さとはパスに含まれる辺の長さの総和 $\sum_{e \in P} l(e)$ である．

3.2 ダイクストラ法

最短路問題の代表的な解法の一つにダイクストラ法がある．ダイクストラ法は，すべての辺の長さ $l(e)$ が非負であるとき，ある固定された頂点を始点 s とし，それ以外の全ての頂点 v への最短路と最短距離 $d(v)$ を求めるアルゴリズムである[1),2)]．辺の長さに対する非負条件を効率的に用いたアルゴリズムであり，計算効率は非常によい．

3.3 ワーシャル・フロイド法

辺の長さが非負とは限らないときは，ダイクストラ法は使えない．そのような場合にも適用可能な最短路問題の解法アルゴリズムとして，ワーシャル・フロイド法がある[1),2)]．ワーシャル・フロイド法は，負の長さの辺があるネットワークにおいて，ダイクストラ法のようにある指定された始点からの最短路だけでなく，すべての2点間の最短路と最短距離を求めるアルゴリズムである．計算効率はダイクストラ法には劣るが，それでも十分高速なアルゴリズムである．

4. 最小費用流問題

有向グラフ $G = (V, E)$ に以下のようなデータが与えられているとする．

・下限容量 $b : E \to \mathbb{R}$

・上限容量 $c : E \to \mathbb{R}$

ここで，$b(e) \leqq c(e)$ が任意の $e \in E$ について満たされるものとする．

・コスト関数 $\gamma : E \to \mathbb{R}$

・$\sum_{v \in V} d(v) = 0$ をみたす需要関数 $d : V \to \mathbb{R}$

最小費用流問題とは以下の２つの条件

1. $b(e) \leq f(e) \leq c(e) \quad$ for $e \in E$ (容量制約)
2. $\sum_{e^+=v} f(e) - \sum_{e^-=v} f(e) = d(v) \quad$ for $v \in V$ (需要制約)

を満たす関数 $f : E \to \mathbb{R}$ のなかで $\gamma(f) = \sum_{e \in E} \gamma(e) f(e)$ で定義される f のコスト $\gamma(f)$ を最小とする f を求める問題である．

$d(v) < 0$ の頂点を供給点またはソースといい，$d(v) > 0$ の頂点を需要点またはシンクという．また $d(v) = 0$ をみたす頂点を移送点という．需要制約をみたすような $f : E \to \mathbb{R}$ をフローと呼び，需要制約と容量制約の両方をみたす f を実行可能流と呼ぶ．

5. ネットワークシンプレックス法

本節では，Dieter Jungnickel の教科書 [2] に従って，ネットワークシンプレックス法について簡単な解説を行う．

5.1 木解と認容木構造

有向グラフ G 上の最小費用流問題を考える．まず，木解について説明する．T が $G = (V, E)$ の全域木であるとは，T が G の閉路を含まない連結部分グラフであり，頂点集合が G の頂点全体と一致するときををいう．実行可能流 f が T に関する木解であるとは，$e \notin T$ に対して，$f(e)$ の値が下限容量 $b(e)$ または上限容量 $c(e)$ に等しくなるものである．f を最小費用流問題 P の実行可能流とする．そのとき $b(e) < f(e) < c(e)$ を満たす辺 e をフロー f に関する自由辺という．f が全域木 T に関する木解であるための必要十分条件は G の全域木 T がフロー f の自由辺をすべて含むことである．P が実行可能流を持つなら，最適な木解をもつことが示されている．よって，最小費用流（最適流）を求めるには，木解の中から探せばよい．

有向グラフ $G = (V, E)$ 上で定義された最小費用流問題の全域木 T に関する木解 f において，$E \setminus E(T)$ の辺の中から，$f(e) = b(e)$ を満たす辺集合を L，$f(e) = c(e)$ を満たす辺集合を U とおくことで，分割 $E \setminus E(T) = L \cup U$ が得られる．そのとき，(T, L, U) を木解 f に付随する認容木構造という．木解 f に付随する認容木構造 (T, L, U) が与えられたとき，f に関する自由辺は全て T に含まれているが，全ての T の辺が自由辺であるわけではない．

5.2 変形コストと木解の最適性の判定

次に認容木構造が最適解の木構造となるための条件を調べるために，変形コスト関数の概念を導入する．

ポテンシャル関数 $\pi : V \to \mathbb{R}$ を用いて，コスト $\gamma : E \to \mathbb{R}$ に対して，ポテンシャル π により変形されたコスト関数 γ_π を

$$\gamma_\pi(uv) = \gamma(uv) + \pi(u) - \pi(v) \quad \text{for} \quad uv \in E$$

と定義する．このとき，コスト関数 γ に関するフロー f のコスト $\gamma(f)$ と、変形コスト関数 γ_π に関するフロー f のコスト $\gamma_\pi(f)$ の差は

$$\gamma f - \gamma_\pi(f) = \sum_{v \in V} \pi(v) d(v)$$

となり，フロー f によらない定数であるから，フロー f の最適性の判定を変形コスト γ_π を用いて行うことができる．実際，最小費用流問題の認容木構造 (T, L, U) が最適木構造であるための必要十分条件は，以下の条件を満たすポテンシャル $\pi : V \to \mathbb{R}$ が存在することであることがわかる．

$$\gamma_\pi(e) \begin{cases} = 0 & \forall e \in T \\ \geq 0 & \forall e \in L \\ \leq 0 & \forall e \in U \end{cases}$$

(T, L, U) を最小費用流問題に付随する木構造とする．$x \in V$ を任意にとるとき，

$$\pi(x) = 0 \quad \text{and} \quad \gamma_\pi(e) = 0 \quad \forall e \in T$$

を満たすポテンシャル $\pi : V \to \mathbb{R}$ は，$e \in T$ に対して

$$uv \in T \text{ なら} \gamma(uv) + \pi(u) - \pi(v) = 0$$

を用いて計算することができる．T は G の全域木であるから，頂点 x と任意の頂点 v を結ぶパスが T 内に 1 本だけあることに注意すれば，ポテンシャルは一意的に決定される．以上により，認容木構造 (T, L, U) が与えられたとき，V のある固定された頂点 x のポテンシャルの値を 0 で定めれば，$e \in T$ に対して，π によって変形されたコスト関数 $\gamma_\pi(e)$ が 0 になることより，ポテンシャル π が一意的に決定される．そのとき，そのポテンシャルを用いて，L と U 上で γ_π の値を計算することにより，認容木構造 (T, L, U) の最適性を判定することができる．

5.3 初期木解

有向グラフ $G = (V, E)$ 上の最小費用流問題 P の最適木解を求める際，最初に問題となるのは，初期の認容木解 (初期木解) を見つけることである．そのため初期木解を簡単に求めるために，有向グラフ G に 1 つの頂点 x と，x と G の全ての頂点を結ぶ辺を付け加えて有向グラフ $G' = (V', E')$ をつくり，$G' = (V', E')$ 上の補助的な最小費用流問題 P' を以下のように定義する．

頂点集合：$V' = V \cup \{x\}$

需要関数：$d'(v) = d(v) \quad \text{for } v \in V, \quad d'(x) = 0$

辺集合：$E' = E \cup \{xv : d(v) - \sum_{e^+=v} b(e) + \sum_{e^-=v} b(e) \geq 0\} \cup \{vx : d(v) - \sum_{e^+=v} b(e) + \sum_{e^-=v} b(e) < 0\}$

下限容量：$b'(e) = b(e) \quad \text{for } e \in E, \quad b'(xv) = 0 \quad \text{for } xv \in E', \quad b'(vx) = 0 \quad \text{for } vx \in E'$

上限容量：$c'(e) = c(e) \quad \text{for } e \in E, \quad c'(xv) = d(v) - \sum_{e^+=v} b(e) + \sum_{e^-=v} b(e) + 1 \quad \text{for } xv \in E', \quad c'(vx) = -d(v) + \sum_{e^+=v} b(e) - \sum_{e^-=v} b(e) + 1 \quad \text{for } vx \in E'$

目的関数：$\gamma'(e) = \gamma(e) \quad \text{for } e \in E, \quad \gamma'(xv) = M \quad \text{for } xv \in E', \quad \gamma'(vx) = M \quad \text{for } vx \in E'$

$M := 1 + \frac{1}{2}|V| \max\{|\gamma(e)| : e \in E\}$

このとき，P' には必ず最適解 f が存在し，さらに以下の 2 つのどちらかが成り立つ．

1. ある $xv \in E'$ に対して，$f(xv) > 0$ となるならば，P には実行可能流は存在しない．
2. 全ての $xv \in E'$ に対して，$f(xv) = 0$ ならば f の E への制限は P の最適解である．

P' を P から作られた補助的な最小費用流問題として，

$$T = \{xv | xv \in E'\} \cup \{vx | vx \in E'\}, \quad L = E, \quad U = \emptyset$$

とおく．そのとき，(T, L, U) は P' の認容木構造となる．これを P' の初期木解という．この初期木解から出発して，ネットワークシンプレクス法の手続きにより，木解の更新を行い，最終的に得られた P' の最適木解に対して，$e \in E' \setminus \quad E$ について，$f(e) = 0$ なら，得られた P' の最適木解を E に制限したものが，元の問題 P の最適解である．

5.4 ネットワークシンプレックス法のアルゴリズム

P を最小費用流問題とし，その補助問題を P' とする．

Step.1 初期化

$$T = E' \backslash E \quad L = E \quad U = \emptyset$$
$$f(e) = b(e) \quad \text{for} \quad e \in E \quad \text{and}$$
$$f(e) = c'(e) - 1 \quad \text{for} \quad e \in E' \backslash E$$
$$\pi(x) = 0$$
$$\pi(v) = M \quad \text{for} \quad (v \in V, xv \in E'),$$
$$\pi(v) = -M \quad \text{for} \quad (v \in V, vx \in E')$$

Step.2 最適化条件の検証

$e \in L \cup U$ に対して，

$(*) \quad \gamma_\pi(e) \geq 0 \quad \text{for } e \in L, \quad \gamma_\pi(e) \leq 0 \quad \text{for } e \in U$

が成り立つかどうか調べる．

この条件が成り立つなら，次に $f(e) > 0$ が成り立つ辺 $e \in E' \backslash E$ があるかどうか調べて，そのような辺があれば，P に実行可能流は存在しない．任意の辺 $e \in E' \setminus E$ について $f(e) = 0$ ならば，f を E に制限したものが，最適解（最小費用流）となる．

Step.3 プライシング

最適化の条件 $(*)$ が満たされなければ，$\gamma_\pi(e) < 0$ となる辺 $e \in L$ または $\gamma_\pi(e) > 0$ となる辺 $e \in U$ が存在する．T は全域木であるから $T \cup \{e\}$ は唯一つの閉路を含むのでその閉路を $C_e(T)$ とおく．

Step.4 増加

$e \in L$ なら $C_e(T)$ の向きを e の向きに一致させ，$e \in U$ なら $C_e(T)$ の向きを e の逆の向きにとる．そのとき，閉

路 $C_e(T)$ において，上記で定めた $C_e(T)$ の向きにフローを追加することができる．増加可能量の最小値を δ とすれば，δ だけフローを追加すると，$C_e(T)$ に含まれる辺のどれかが自由辺ではなくなり，上限容量または下限容量に達する．その辺を a とする．また $\delta > 0$ である限り，$a = e$ となることも許容する．

Step.5 更新

$$T := (T \backslash \{a\}) \cup \{e\}$$

$$L := \begin{cases} (L \backslash \{e\}) \cup \{a\} & (f(a) = b(a)) \\ (L \backslash \{e\}) & (f(a) = c(a)) \end{cases}$$

$$U := E' \backslash (T \cup L)$$

とおいて，新たな木構造 (T, L, U) を定義し，この木構造に付随したポテンシャル π を再計算し，**Step.2** へ戻る．

5.5 最終閉鎖辺選択規則

ネットワークシンプレックス法において，**Step.3** で選ばれる辺 e を入辺といい，**Step.4** で選ばれる辺 a を出辺という．また，$C_e(T) \subset T \cup \{e\}$ をピボット閉路という．辺 e を入辺とし，ピボット閉路 $C_e(T) \subset T \cup \{e\}$ をとる．e の向きに応じて $C_e(T)$ に向き付けを行い，閉路に可能な限りフローを増加させる．そのとき，フローの上限または下限値に達して，フローの追加が不可能になる辺ができる．このような辺 $a \in C_e(T) \setminus \{e\}$ を閉鎖辺という．一般的には閉鎖辺は一つではない．複数の閉鎖辺があるとき，そのなかから，適当に閉鎖辺を選んでそれを出辺とすると，アルゴリズムに循環が起きる場合があることが知られている．循環とは，ある認容木構造から出発して，フローの追加がないままに，何回かのステップで最初の木構造に戻ることをいう．このような循環をさけるためには，出辺の選択において次で説明する最終閉鎖辺選択規則と呼ばれる規則に従えばよいことが知られている．

1) グラフのある頂点を固定し，全域木 T を，その頂点を根とする根付き木と考える．さらに根から最も近い $C_e(T)$ に含まれる頂点を選びそれを向点と呼ぶ．

2) 向点から出発して，$C_e(T)$ の向きに沿って進み，最も最後に現れる閉鎖辺を出辺とする．

6. 実装にあたって

6.1 実装にあたっての留意点

本論文において実装したネットワークシンプレックス法の関数は MathWorks のユーザーコミュニティに投稿しており，誰でも簡単にダウンロードして使用することができる．(http://www.mathworks.com/matlabcentral/fileexchange/29665)

本実装においては，下限容量 $b(e)$ は 0 としていることを断っておく．

今回の実装にあたり，最短路問題のアルゴリズムを用いた箇所を説明する．

(I) ネットワークシンプレックス法のアルゴリズムの **Step.3** の操作で，グラフ $G = (V, E)$ の全域木 T に一つの辺 $e = (u, v)$ を付け加えるとただ一つの閉路 $C = C_e(T)$ ができる．本実装ではダイクストラ法を用いてこのような閉路を求める．

まず，新たな有向グラフ $G_e(T) = (V, E_e)$ を考える．ここで $G_e(T)$ の頂点集合は G の頂点集合と同じでであり，辺集合 E_e は $E(T) \cup \{e\}$ の有向辺全体と $E(T) \cup \{e\}$ の辺の逆向きの辺全体の和集合からなるものとする．ここで，$E(T)$ は全域木 T の辺集合を表す．

このグラフ $G_e(T)$ の各辺に新たな距離関数 $r : E \to \mathbb{R}$ を導入し，新たなネットワーク $N = (G_e(T), r)$ を定義する．この長さ r は

$$r(e) = \begin{cases} 1 & \text{for} \quad T \text{ の辺およびその逆向きの辺} \\ \infty & \text{for} \quad e = (u, v) \qquad \text{or} \quad e = (v, u) \end{cases}$$

とする．ここで，ダイクストラ法を用いて，N 上で u, v 間の最短路を求める．得られた u から v への最短パスにおいて，パスに含まれる辺の向きとして T において元来付けられていた向きを採用し，そのパスに辺 $e = (u, v)$ を付け加えることにより，求めるべき閉路 $C_e(T)$ が得られる．

(II) ネットワークシンプレックス法では，各点に与えられるポテンシャルを木構造が更新されるたびに再計算す

るが，その際のポテンシャルの計算方法に，最短路問題を用いる．

ポテンシャルが満たすべき条件は，ポテンシャルによって変形されたコスト γ_π により，

$$\pi(x) =, 0 \;\; \gamma_\pi(uv) = \gamma(uv)+\pi(u)-\pi(v) = 0 \; (e = uv \in T)$$

と与えられる．

上記の規則に従ってポテンシャルの値を計算するために，新たな有向グラフ $G_T = (V', E'')$ 上のネットワークを考える．ここで，有向グラフ G_T の頂点集合 V' は補助問題 P' が定義されている有向グラフ $G' = (V', E')$ の頂点集合 V' であり，G_T の辺集合 E'' は G' の全域木 T の辺集合と T の辺集合の逆向きの辺全体の和集合とする．このような有向グラフ $G_T = (V', E'')$ 上に新たな距離関数 $r' : E'' \to \mathbb{R}$ を次のように定義する．

$$r'(uv) = \begin{cases} \gamma(uv) & \text{for} \quad uv \in T \\ -\gamma(vu) & \text{for} \quad vu \in T \\ \infty & \text{for} \quad uv \notin T \quad \text{and} \quad vu \notin T \end{cases}$$

T は全域木であるから，ここで定義したグラフ G_T の基礎無向グラフにおいて，$x \in V$ から他の頂点 v までの無向パスがただ一つ存在する．そこで，x を始点とし，新たな距離関数 r' において各点までの最短路の長さをワーシャル・フロイド法で求める．距離関数 r' に関する頂点 x から頂点 v への最短路の長さを求めると，ポテンシャルに対する条件から，最短路の値が頂点 v におけるポテンシャル $\pi(v)$ となる．

(III) 最終閉鎖辺選択規則の実装にも最短路問題を応用している．最終閉鎖辺選択規則とは，すでに説明したように，ネットワークシンプレックス法のアルゴリズムの **Step.4,5** において，閉路 $C_e(T)$ に閉鎖辺が複数現れたとき，その中から特定の閉鎖辺をアルゴリズムにおける出辺として選ぶ規則であり，1) 向点を選ぶ規則と　2) 閉路 $C_e(T)$ 上で，向点から出発して，$C_e(T)$ の向きに進み，もっとも最後に現れる閉鎖辺を選ぶという規則からなる．まずは，1) の向点を求める方法について説明する．新たなグラフ $G_a = (V', E_a)$ を考える．このグラフ G_a の頂点集合 V' は補助問題 P' が定義されている有向グラフ $G' = (V', E')$ の頂点集合 V' で，G_a の辺集合 E_a は G' の辺集合 E' と E' の辺の逆向きの辺全体の和集合とする．このような有向グラフ $G_a = (V', E_a)$ 上に新たな距離関数 $r_a : E \to \mathbb{R}$ を次のように定義する．

$$r_a(uv) = 1 \quad \text{for} \quad uv \in E' \quad \text{or} \quad vu \in E'$$

ここで新たな距離関数 r_a に関して，点 x から閉路 $C_e(T)$ 上の各点への最短路を求める．そのとき，x からの距離が最短である $C_e(T)$ 上の点が向点である．最短距離が同じ頂点が複数あった場合は頂点番号が一番小さいものを向点に選ぶ．

続いて 2) の操作に移る．(I) で説明したようにダイクストラ法で求められた閉路 $C_e(T)$ は頂点番号の並びによって与えられている．そこで，閉路の表示を向点が最初に来るように頂点番号を並びかえる．こうすることにより，簡単に最後に現れる閉鎖辺を選ぶことが可能となる．

6.2 関数の使い方

本論文におけるネットワークシンプレックス法の MATLAB への実装は，メイン関数を定義している一つの m-file(simplex.m) と，そのメイン関数で使われるサブモジュールとなる関数を定義している複数の m-file で構成されている．このメイン関数における入力や出力，使い方を説明する．

メイン関数を定義する m-file のファイル名（関数名）は simplex.m である．この関数は，グラフ上の各辺の容量 a，各頂点上の需要関数 d，各辺のコスト関数 g を入力として，実行可能流が存在すれば最小費用流を返す．具体的な入力は以下の 3 つである．

1. 上限容量は重みつき隣接行列 a で与える．すなわち，頂点数が n のグラフでは，a は $n \times n$ の行列で a の (i, j) 成分は辺 (i, j) の上限容量である．要素 $a(i, j)$ は非負の整数と仮定している．下限容量はすべての辺で 0 としている．
2. 需要関数 d は n 次元のベクトルで表され，$d(v)$ は頂点 v における需要関数を表している．ここで，頂点

集合 $V = \{1, 2, \ldots, n\}$ としている．需要関数についても，すべての $d(v)$ は整数であると仮定する．

3. コスト関数も重みつき隣接行列 g で定義される．すなわち，g は $n \times n$ の行列で $g(i, j)$ は辺 (i, j) のコストを表している．コストについても，すべての $g(i, j)$ は非負の整数であると仮定する．

出力は，実行可能流が存在すれば minf という $n \times n$ 行列が返される．minf(i, j) は非負の整数で，最小費用流となるときの，辺 (i, j) のフローを表している．
もし元のネットワークに実行可能流が存在しなければ minf $= 0$ というスカラーを返す．

また，この simplex.m という関数は与えたネットワークが最小費用流の各辺のフローの値を返すだけで，そのときのコストは表示しない．最小費用流のコストを計算する際は，value.m という関数 (m-file) を使用する．

以下に具体的な最小費用流問題をあげ，本研究で作成した m-file を用いて MATLAB 上で最小費用流を求める．

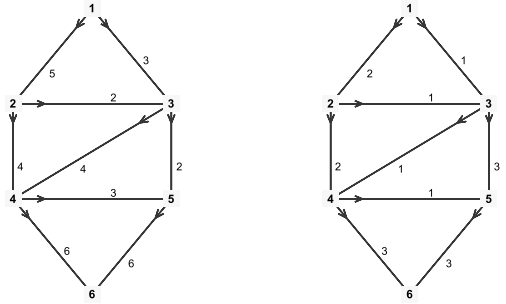

Fig. 1. An example of Minimal Cost Flow Problem. (the left:capacities; the right: costs)

Fig. 1 のようなネットワークが与えられたと仮定する．上記の２つの図は同じグラフであり，左の図が各辺の容量を，右の図が各辺のコストを表している．ここで各頂点上の需要関数を $d(1) = -4, d(2) = -1, d(3) = -2, d(4) = -1, d(5) = 0, d(6) = 8$ とする．このネットワーク上の最小費用流問題を解く．

はじめに MATLAB にネットワークの容量 a，需要関数 d，コスト関数 g を入力する．容量とコスト関数は $n \times n$ 行列，需要関数は n 次元ベクトルであり，以下のように入力する．

```
a=[0 5 3 0 0 0;0 0 2 3 0 0;0 0 0 4 2 0;0 0 0 0 3 6;
   0 0 0 0 0 6;0 0 0 0 0 0]
d=[-4 -1 -2 -1 0 8]
g=[0 2 1 0 0 0;0 0 1 2 0 0;0 0 0 1 3 0;0 0 0 0 1 3;
   0 0 0 0 0 3;0 0 0 0 0 0]
```

次に，

```
minf=simplex(a,d,g)
```

と入力すれば，以下のような出力を得る．

```
minf =
     0     1     3     0     0     0
     0     0     0     2     0     0
     0     0     0     4     1     0
     0     0     0     0     1     6
     0     0     0     0     0     2
     0     0     0     0     0     0
```

これがネットワークシンプレックス法を用いて最小費用流を求めた時の各辺のフローである．この最適解は，与えられたネットワーク上の各辺の容量制約と，各頂点上の需要制約を満たしていることは簡単に確認できる．

また value.m という関数は現在のフローの総コストを計算する関数であり，

```
value(minf,g)
```

と入力すると

```
ans =
    41
```

という出力が返される．これは求めた最小費用流の総コストが 41 であるということを意味している．

7. 結論

本論文の目的は，最小費用流問題の代表的なアルゴリズムであるネットワークシンプレックス法のアルゴリズムを MATLAB に実装することであった．このネットワークシンプレックス法は様々なグラフ理論のアルゴリズムの集合体のようなもので，実装に際しては各計算ステップをどのようなグラフ理論の問題と捉え，どのような手法で解くかという自由度が非常に高いアルゴリズムである．本論文ではこの実装に際して効率よりも，分かり易

さを重視し，様々なところで最短路問題を用いて実装を行った．その結果，比較的簡便な 300 ステップ程度のネットワークシンプレックス法の実装が可能となった．この実装した関数では頂点数が 100 個程度のネットワークであれば，殆どストレスなく最小費用流を計算できることがわかり，実装の有用性が確かめられた．しかし，このアルゴリズムを用いて，実際の最小費用流問題を解くという実験は十分に行うことが出来なかった．また，今後の検討課題として，本論文の実装がどれくらいのサイズ (頂点数や辺の数) の問題の計算に耐えられるかを検証する必要がある．そのような限界を把握した上で，より大きな問題を解けるような改良を加えることが必要であると考えている．改善すべき点の一つとして，問題の表現方法の工夫があげられる．本実装では，入力として与えられる辺の上限容量，コストなどのネットワークのデータを，重みつき隣接行列で与えている．このような隣接行列を用いたグラフの表現は，見た目が分かりやすく，コーディングが容易であるという利点があるが，一方でその形でデータをコンピュータに蓄えるには，頂点数の平方の記憶容量を必要とする．しかし実際には，問題にもよるが，多くの場合殆どの隣接行列の要素が 0 になっており，記憶容量の節約という点では無駄が多いし，入力も大変である．従って，本論文で採用した問題の表現は大きなサイズの問題には向いていない．

計算機実験をスムーズに行うには，問題作成の自動化も大切である．そのためには，例えば MATLAB の乱数を発生させる機能を使って最小費用流問題を自動生成する関数を作成することが必要である．しかし，辺の容量，辺のコスト，需要関数など，全てのパラメータを乱数を使って自動生成すると，問題のサイズが大きくなれば大きくなる程，生成されたネットワークにおいて実行可能流が存在しない可能性が大きくなってしまうことが予想される．従って，問題を自動生成の鍵は，以下にして実行可能流が存在するネットワークをたくさん生成できるかどうかにかかっている．

本実装では，アルゴリズムを構成する各部分で，様々な形で最短路問題を使う仕様になっている．その際，可能なら最も高速なアルゴリズムであるダイクストラ法を，重みが負の辺が存在するときは，ワーシャル・フロイド法を用いて最短路を求めている．最短経路問題は簡単な問題であり，その解法アルゴリズムは十分に高速であるとはいえ，部分アルゴリズムとして，最短経路問題の解法を採用することが最も効率がよいとは言えない．計算効率を上げるために，各計算ステップを見直して，各ステップの計算を最短路問題として定式化するのではなく，もっと効率よく解ける問題として定式化してその問題の解法を適用する必要がある．

また，ポテンシャルの更新や木構造の更新に際して，最初から再計算する方法を採用しているが，計算の効率を上げるためには更新のルールを明確化し，現状の値を利用してポテンシャルや木構造を更新するという工夫も必要になる．

以上残された課題は多いが，今回グラフ上のネットワーク理論に関する様々な予備知識を必要とせず，この複雑なアルゴリズムであるネットワークシンプレックス法を実際に動かすことができたことは有意義なことであると考えており，この改善は将来の課題である．

参 考 文 献

1) Bernhard Korte and Jens Vygen. *Combinatorial Optimization third edition*. Springer,(2005).

2) Dieter Jungnickel. *Graphs, Networks and Algorithms second edition*. Springer, (2005).