数理解析研究所講究録
第726巻 1990年 111-125

非線形最小費用流問題に対する双対ニュートン法

京都大学工学部　茨木　智　(Satoru Ibaraki)
京都大学工学部　福島雅夫　(Masao Fukushima)
京都大学工学部　茨木俊秀　(Toshihide Ibaraki)

1. はじめに

本報告では，分離可能な（微分可能とは限らない）費用関数を持つ非線形最小費用流問題に対するアルゴリズムを考察する．とくに費用関数が狭義凸関数のとき，この問題の双対問題は微分可能な目的関数をもつ制約なしの最適化問題となる．[2] では Gauss-Seidel 型の緩和法を双対問題に適用している．また [7] では主問題に対して直接ニュートン法を用いる方法が報告されている．ここではニュートン法を用いて双対問題を解くことを考察する．

節点の集合 $\mathcal{N}=\{1,2,\ldots,m\}$，枝の集合 $\mathcal{A}=\{a_1,a_2,\ldots,a_n\}$ で構成される有向グラフ $\Gamma=(\mathcal{N},\mathcal{A})$ において，枝 a_j の始点，終点がそれぞれ i,k のとき，$a_j=(i,k)$ と表し，各枝 a_j のフローを x_j，費用を $f_j(x_j): R\to R\cup\{+\infty\}$ で表す．次の最小費用流問題を考える．

$$\text{(P)}\quad \text{minimize}\quad f(x)=\sum_{j=1}^{n} f_j(x_j)$$

$$\text{subject to}\quad \sum_{j=1}^{n} e_{ij}x_j=b_i,\quad i\in\mathcal{N}-\{m\} \tag{1}$$

ただし，$x\in R^n$ は x_j を要素とするベクトルであり，e_{ij} は次式で定義される接続行列 $E\in R^{(m-1)\times n}$ の要素である．

$$e_{ij}=\begin{cases} 1 & \text{節点 } i \text{ が枝 } a_j \text{ の始点のとき} \\ -1 & \text{節点 } i \text{ が枝 } a_j \text{ の終点のとき} \\ 0 & \text{それ以外のとき．} \end{cases}$$

（P）における b_i は，各節点 $i\in N$ に対する需要（供給）量 b_i を表しており，$b_i>0$ のとき節点 i は供給点，$b_i<0$ のときは需要点，$b_i=0$ のときは通過点である．また $\sum_{i\in\mathcal{N}} b_i=0$ が成立していると仮定する．そのとき，節点 m に対する流量保存則は冗長となるので，制約条件式 (1) は取り除かれている．また，このとき行列 E は full rank である．

ここで各費用関数 $f_j(x_j)$ に対して u_j,l_j を

$$u_j = \sup\{x_j | f_j(x_j) < +\infty\}$$

$$l_j = \inf\{x_j | f_j(x_j) < +\infty\}$$

で定義しよう．この u_j および l_j をそれぞれフロー x_j の上限値，下限値と呼ぶ．($u_j = +\infty$ または $l_j = -\infty$ を許すものとする)．

一般に最小費用流問題では，費用関数 $f_j(x_j)$ はすべての x_j において有限値をとり，フロー x_j の値が枝の上下限値 u_j, l_j に対して $l_j \leq x_j \leq u_j$ を満たすという制約を考えるのがふつうである．このような場合は枝 a_j の費用関数を次式で定義すると，問題 (P) の形に定式化できる．

$$f_j(x_j) = \begin{cases} \hat{f}_j(x_j) & x_j \in [l_j, u_j] \text{ のとき} \\ +\infty & \text{それ以外のとき} \end{cases}$$

ここで，u_j および l_j は与えられたフローの上・下限値であり，$\hat{f}_j(x_j)$ はすべての実数 $x_j \in (-\infty, +\infty)$ に対して，有限の値をとるものとする．

本報告では，各費用関数 $f_j(x_j)$ に対して次の性質を仮定する．

仮定1 集合 $\{x_j | f_j(x_j) < +\infty\} \subset R$ において，関数 f_j は下半連続かつ狭義凸である．

仮定2 関数 f_j は co-finite[8] である．すなわち f_j は

$$\lim_{x_j \to \pm\infty} \frac{f_j(x_j)}{|x_j|} = +\infty$$

を満たす．

ただし，区間 $\{x_j | f_j(x_j) < +\infty\}$ が有界ならば，仮定2は自動的に満たされる．

2. 双対問題

第2節では，主問題（P）の双対問題を定式化し，その目的関数の微分可能性について考える．ラグランジュ乗数ベクトルを $p \in R^{(m-1)}$ (p_i は節点のポテンシャルとも呼ばれる) として，問題（P）のラグランジュ関数 L を

$$L(x,p) = \sum_{j=1}^{n} f_j(x_j) - \sum_{i=1}^{m-1} p_i (\sum_{j=1}^{n} e_{ij} x_j - b_i)$$

で定義すると，双対問題（D）は次のようになる．

(D) minimize $q(p)$

ただし，

$$\begin{aligned} q(p) &= -\inf_x L(x,p) \\ &= \sup_x \sum_j (x_j \sum_i e_{ij} p_i - f_j(x_j)) - \sum_i b_i p_i \end{aligned}$$

簡便のために以下では，枝 $a_j = (l,k)$ のテンション t_j を

$$t_j = \sum_i e_{ij} p_i = p_l - p_k, \quad \forall j = 1,2,\ldots,n$$

で定義する．さらに f_j の共役関数を

$$f_j^*(t_j) = \sup_{x_j} \{x_j t_j - f_j(x_j)\} \tag{2}$$

で定義すると，上の双対関数は

$$\begin{aligned} q(p) &= \sum_j f_j^*(\sum_i e_{ij} p_i) - \sum_i b_i p_i \\ &= \sum_j f_j^*(t_j) - \sum_i b_i p_i \end{aligned} \tag{3}$$

と書き換えられる．仮定2より，関数 f_j^* は凸であり，かつ任意の t_j に対して有限な値をとる．(2) の右辺の最大を与える x_j の値を $\bar{x}_j$ とすれば，仮定1より $\bar{x}_j$ は一意的に定まる．そのとき関数 f_j^* は微分可能であり，

$$f_j^{*\prime}(t_j) = \bar{x}_j, \quad \forall j \tag{4}$$

が成り立つ [8]．さらに，(2)-(4) より，双対問題（D）の目的関数 $q(p)$ も微分可能であり，その勾配 $\nabla q(p)$ の各成分は

$$\frac{\partial q(p)}{\partial p_i} = \sum_j e_{ij} f_j^{*\prime}(t_j) - b_i = \sum_j e_{ij} \bar{x}_j - b_i, \quad \forall i \in \mathcal{N} - \{m\}$$

と表される．ゆえに，関数の勾配ベクトルは

$$\nabla q(p) = E\bar{x} - b$$

で得られる．したがって，双対問題（D）は微分可能な目的関数を持つ，制約なしの凸最小化問題となる．

ここではさらに関数 $q(p)$ の2次微係数について考察する．関数 $q(p)$ が (3) で表されていることから，すべての j に対して $f_j^{*\prime\prime}(t_j)$ が存在すると仮定すれば，$q(p)$ のヘッセ行列 $\nabla^2 q(p)$ は

$$\nabla^2 q(p) = EH(t)E^T \tag{5}$$

と表せる．ただし $t = E^T p$ であり，$H(t)$ は次式で定義される対角行列である．

$$H(t) = \mathrm{diag}(f_j^{*\prime\prime}(t_j)) \tag{6}$$

次の定理は，ある与えられたt_jに対して$f_j^{*\prime\prime}(t_j)$が存在しかつその値が正であるための条件を表している．

定理 1 任意のt_jに対して，*(4)*により定まる$\overline{x}_j$において，$f_j''(\overline{x}_j)$が存在し，かつ$f_j''(\overline{x}_j) > 0$と仮定する．このとき，その共役関数$f_j^*(t_j)$も2回微分可能であり，

$$f_j^{*\prime\prime}(t_j) = \frac{1}{f_j''(\overline{x}_j)} > 0$$

が成り立つ．

以下で述べるアルゴリズムでは，$q(p)$や$f_j^*(t_j)$を陽に関数の形で表現する必要はなく，与えられたpに対して関数の値を評価するだけで十分である．

ところで，一般に問題(D)の最適解は必ずしも唯一ではないが，次の定理の条件のもとでは最適解の唯一性が保証される．

定理 2 $\overline{p}$を双対問題(D)の任意の最適解とし，$\overline{t} = E^T\overline{p}$とする．また，すべての$j$に対して$f_j^{*\prime\prime}(\overline{t}_j)$が存在すると仮定する．このとき，

$$\hat{\mathcal{A}} = \{a_j \in \mathcal{A} | f_j^{*\prime\prime}(\overline{t}_j) > 0\}$$

で定義される枝の集合$\hat{\mathcal{A}}$と節点の集合$\mathcal{N}$で構成されるΓの部分グラフ$(\mathcal{N}, \hat{\mathcal{A}})$が連結ならば，$\overline{p}$は問題(D)の唯一解である．

3. 方法

問題（D）は完全に制約のない問題であるので，目的関数の勾配を用いた降下法によって解くことができる．しかし本報告では，関数の2次の情報を利用した，より効率のよいアルゴリズムを提案する．

3.1. アルゴリズム

基本アルゴリズム

ステップ0： 初期解$p \in R^{(m-1)}$を定める．

ステップ1： 正定値対称行列$Q(p) \in R^{(m-1)\times(m-1)}$を選び，連立1次方程式

$$Q(p)s = -\nabla q(p) \tag{7}$$

を解いて，降下条件$\nabla q(p)^T s < 0$を満たす探索方向$s \in R^{(m-1)}$を求める．

ステップ2 : 次のδに関する1次元の最小化問題を（近似的に）解いてステップ長$\delta > 0$を求める.

$$\min_{\delta>0} q(p+\delta s) \tag{8}$$

ステップ3 : $p := p + \delta s$と更新して，ステップ1に戻る.

ステップ1の方向sが降下条件を満たすことから，反復を繰り返すことにより目的関数値は単調減少する．各pにおいて関数$q(p)$が2回微分可能であれば，そのヘッセ行列を$Q(p)$とすることにより，(7)から得られる方向sはニュートン探索方向となる．また，$Q(p) \equiv I$とおけば方向sは最急降下方向となる．またステップ2において，$q(p+\delta s) < q(p)$を満たすようなステップ長$\delta > 0$が必ず求まる．しかし厳密にいうと，上で述べた条件だけでは基本アルゴリズムの大域的収束性は必ずしも保証されない．以下では，探索方向の計算および直線探索に対して，基本アルゴリズムの大域的収束性を保証する条件および具体的な計算法について述べる.

3.2. 探索方向の計算

(7)で得られる方向sを用いるとき，基本アルゴリズムが収束するための条件の1つは，任意のベクトル$y \neq 0$に対して

$$0 < \lambda \leq \frac{y^T Q(p) y}{y^T y} \leq \Lambda \tag{9}$$

を満たす，pに独立な正定数λおよびΛが存在することである[3]．本報告では，f_j^*の2次の情報をできるだけ利用して条件(9)を満たすような行列$Q(p)$を構成する．(5)および(6)を考慮して，$H(t)$を対角行列

$$H(t) = \mathrm{diag}(h_j(t_j))$$

とし，これを用いて，行列$Q(p)$を

$$Q(p) = EH(t)E^T \tag{10}$$

で与えることを考えよう．このとき行列Eはfull rankなので，もし行列$H(t)$の各対角成分$h_j(t_j)$があるある$\bar{c} > \underline{c} > 0$に対して，$\underline{c} \leq h_j(t_j) \leq \bar{c}$を満たすならば，条件(9)は満足される.

具体的な$h_j(t_j)$の値は次のようにして与えた．まず各t_jに対して$f_j^{*\prime\prime}(t_j)$が存在するとき，定数$\underline{c}$および$\bar{c}$に対して，

$$h_j(t_j) = \begin{cases} \underline{c} & f_j^{*\prime\prime}(t_j) < \underline{c} \text{のとき} \\ f_j^{*\prime\prime}(t_j) & \underline{c} \leq f_j^{*\prime\prime}(t_j) \leq \bar{c} \text{のとき} \\ \bar{c} & \bar{c} < f_j^{*\prime\prime}(t_j) \text{のとき} \end{cases}$$

とする．また$f_j^{*\prime\prime}(t_j)$が存在しないとき，$h_j(t_j)$は区間$[\liminf_{z\to t_j} h_j(z), \limsup_{z\to t_j} h_j(z)]$内の任意の数と定める．このとき$Q(p)$は正定値対称行列となり，条件(9)を満足する.

本報告では$Q(p)$が(10)で与えられているとき，連立1次方程式

$$EH(t)E^T s = -\nabla q(p) \tag{11}$$

に対して，共役勾配法 (CG) を用いて探索方向 s を計算している．なぜなら，大規模な問題に対して直接 (11) を解くには多くの記憶領域を必要とするからである．実際には，$Q(p)$ を (10) で表される積の形で扱い，ネットワークのデータ構造を利用すると，$Q(p)$ とベクトルのかけ算は容易に実行される．

またふつう共役勾配法の効率をあげるために，前処理 (precondition) を行うが，この前処理に使われる行列 M を，ここでは $Q(p)$ の対角成分からなる行列と定めた．実際には行列 M の各対角成分は

$$M_{ii} = \sum_{a_j=(i,k)} h_j(t_j) + \sum_{a_j=(k,i)} h_j(t_j)$$

で求められる．

CG algorithm
begin
$k := 0; s_0 := 0; r_0 := -\nabla q(p)$
while a convergence criterion is not satisfied **do**
begin
begin
$\tilde{r}_k := M^{-1} r_k;$
$k := k + 1$
end
if $k = 1$ **then**
$\tilde{s}_1 := \tilde{r}_0$
else
begin
$\beta_k := r_{k-1}^T \tilde{r}_{k-1} / \tilde{s}_k^T Q(p) \tilde{s}_k;$
$\tilde{s}_k := \tilde{r}_{k-1} + \beta_k \tilde{s}_{k-1}$
end
end if
begin
$\alpha_k := r_{k-1}^T \tilde{r}_{k-1} / \tilde{s}_k^T Q(p) \tilde{s}_k;$
$s_k := s_{k-1} - \alpha_k \tilde{s}_k;$
$r_k := r_{k-1} - \alpha_k Q(p) \tilde{s}_k$
end
end
$s := s_k$
end

理論的には共役勾配法は有限回の反復で解を得る．しかし，よく知られているように実際には計算誤差のため有限性が失われるので，以下に示す数値実験では残差 r_k が

$$\|r_k\| < \epsilon_{CG}\|r_0\| \tag{12}$$

を満たしたとき反復を停止させた．ただし r_0 は初期残差，$\epsilon_{CG} > 0$ は定数である．

3.3. 直線探索

この節では，基本アルゴリズムの大域的収束性を保証するステップ長δの決定方法を考える．表記を簡単にするために，関数$\phi(\delta)$ を

$$\phi(\delta) = q(p + \delta s)$$

で定義する．そのとき微係数$\phi'(\delta)$ は

$$\phi'(\delta) = \nabla q(p + \delta s)^T s$$

となり，降下条件は$\phi'(0) < 0$ と等価である．

[3] によると，ステップ長δが条件

$$\phi(\delta) \leq \phi(0) + \delta\rho\phi'(0) \tag{13}$$

および

$$\phi'(\delta) \geq \sigma\phi'(0), \tag{14}$$

を満足するとき最適解への収束性が保証される．ただし$\rho \in (0, \frac{1}{2})$ と$\sigma \in (\rho, 1)$ はあらかじめ定められた定数である．条件 (13) は点 $(0, \phi(0))$ を通り，傾きが$\rho\phi'(0)$ である直線と曲線$\phi(\delta)$ との関係を表している．また (14) は傾き$\phi'(\delta)$ が$\sigma\phi'(0)$ 以上であるための条件である．

以下にステップ長δを見つけるための反復解法を述べる．この方法は，まず (13),(14) を満たすようなδの値を含む区間 (bracket) を見つける手続き (bracketing phase) と，次にこの bracket を分割して最終的に適当なステップ長δを求める手続き (sectioning phase) から成り立っている．

[Bracketing Phase]

choose $\rho \in (0, \frac{1}{2}), \sigma \in (\rho, 1), \gamma \in (1, +\infty)$ and $\delta_1 > 0$;
for $i := 1, 2, \ldots$ **do**
begin
  evaluate $\phi(\delta_i)$;
  if $\phi(\delta_i) \geq \phi(0) + \rho\delta_i\phi'(0)$ **then**
    begin
      $\delta_u := \delta_i$; terminate Bracketing Phase; go to Sectioning Phase
    end
  end if
  evaluate $\phi'(\delta_i)$;
  if $\phi'(\delta_i) \geq \sigma\phi'(0)$ **then** terminate;

set $\delta_{i+1} := \gamma\delta_i$
end
[Sectioning Phase]
set $\delta_0 := 0$;
for $j := i, i+1, \ldots$ **do**
begin
choose $\delta_j \in [\delta_l, \delta_u]$
evaluate $\phi(\delta_j)$;
if $\phi(\delta_j) > \phi(0) + \rho\delta_j\phi'(0)$ **then** $\delta_u := \delta_j$;
else
begin
evaluate $\phi'(\delta_j)$;
if $\phi'(\delta_j) \geq \sigma\phi'(0)$ **then** terminate;
set $\delta_l := \delta_j$
end
end if
end

ただし 'terminate Bracketing Phase' は bracket を見つける手続きの終了を示しており，それが起きた場合，初期区間 $[\delta_l, \delta_u]$ を用いて，ただちに Sectioning Phase にゆく．それに対して，'terminate' は基本アルゴリズムの直線探索の終了を表すもので，得られたステップ長δ_jを用いた基本アルゴリズムは収束する [3]．また Sectioning Phase のδ_jの選択法はさまざまな方法が考えられるが，ここでは関数値やその傾きを利用した 3 次補間法を用いている．

4. 数値実験

前節で述べた方法を FORTRAN77 でコード化し，京都大学大型計算機センターの FACOM M780-30 を用いて倍精度で計算実験を行った．

実験に用いたネットワークは，Fig.1 で与えられるような格子状のもので，左端の列の各節点が供給点 ($b_i > 0$)，右端の列の各節点が需要点 ($b_i < 0$)，それ以外の節点はすべて通過点 ($b_i = 0$) である．これらの需要（供給）量 b_iは，区間 [1,10] 上において一様乱数によって定める．(ただし$\sum_{i\in N} b_i = 0$ である．)

以下の数値実験で用いた問題では費用関数 $f_j(x_j)$ を

$$f_j(x_j) = \begin{cases} c_j x_j + \frac{1}{2} d_j x_j^2, & x_j \in [0, u_j] \text{ のとき} \\ +\infty & \text{それ以外のとき} \end{cases} \tag{15}$$

$$f_j(x_j) = \begin{cases} c_j x_j + \frac{1}{3} d_j x_j^3, & x_j \in [0, u_j] \text{ のとき} \\ +\infty & \text{それ以外のとき} \end{cases} \tag{16}$$

$$f_j(x_j) = \begin{cases} c_j x_j - \mu \log x_j - \mu \log(u_j - x_j), & x_j \in (0, u_j) \text{ のとき} \\ +\infty & \text{それ以外のとき} \end{cases} \tag{17}$$

のいずれかで与えている．ただし，すべての枝 $a_j \in A$ のフロー x_jの下限 $l_j = 0$ としてあり，実数 c_j, d_jおよびフロー x_jの上限 u_jは，それぞれ $c_j \in [1, 20], d_j \in [0.1, 2]$ および $u_j \in [5, 10]$ の一様乱数を用いて定めた．

4.1. 結果

ここでは，基本アルゴリズムの収束判定をベクトル $\nabla q(p)$ のノルムを用いて行い，十分小さい正定数ϵに対して，条件

$$\frac{\|\nabla q(p)\|}{\|\nabla q(p_0)\|} < \epsilon \tag{18}$$

が満たされたとき反復を停止した．ただし，p と p_0は現在および初期の双対変数 (ポテンシャル) の値である．表1，2および3は費用関数がそれぞれ (15),(16) および (17) で与えられる問題に対する実験結果である．どの場合に対しても，初期点 $p_0 = 0$ とし，(18) の$\epsilon = 10^{-3}$としている．前節で述べたように，探索方向を決定する (11) の解 s の精度は，条件 (12) のϵ_{CG}に依存する．表1，2に $\epsilon_{CG} = 10^{-1}$および$\epsilon = 10^{-3}$のときの結果が記されている．それらによると，各反復ごとに (11) の高精度の解を求めれば，基本アルゴリズムの反復回数を減少させるが，全 CPU 時間は逆に増加することがわかる．また表3によると，費用関数が (17) で与えられる問題に対しては，他の2つの場合より多くの CPU 時間を要することがわかる．実際μが小さい場合，上下限からはなれた点では，関数 (17) はほとんど線形であり，ゆえにその傾き $f_j'(x_j)$ は定数となるので，その逆関数である $f_j^{*\prime}$はステップ関数となる．言い替えれば，目的関数 $q(p)$ は数値的には，ほとんど微分不可能となり，収束するまでに多くの反復を要する．

また問題 1-11,12，2-11,12 に対して，表4および5で項目

（a）精度ϵが $10^{-1}, 10^{-2}, 10^{-3}$および 10^{-4}に達するまでの，各反復関数／累積反復回数

（b）精度ϵが $10^{-1}, 10^{-2}, 10^{-3}$および 10^{-4}に達するまでの，各 CPU 時間／累積 CPU 時間

に対する詳しい結果を記す．これから，最適解の近くでは収束の速度が非常に早いことがわかる．これはニュートン法の典型的な特徴であり，他の問題に対しても同様の結果がみられた．

また [10] で提案されている Gauss-Seidel 型の緩和法もコード化して比較実験を行ったが (Fig.2)，ここで提案した方法の方がかなり少ない計算時間で最適解を見いだすことが確認できた．さらに [6] で提案されている，主問題に対するニュートン法とも比較したが (Fig.3)，ここで提案した方法の方が最適解に速く収束することが観察された．この方法は内点法であり，(17) のようなバリア関数を用いて変換された問題を解いているが，初期実行可能解を求めるための人為的な問題を解かねばならず，それに相当する計算を初めに要するからである．

5. おわりに

本報告では，非線形最小費用流問題に対して，大域的収束性を持つニュートン法を提案した．この方法は双対問題直接解く方法であり，主問題の費用関数が凸で co-finite であることを仮定している．数値実験を行った結果，比較的大規模な問題（節点数 1024，枝数 2976）に対しても効率的に解を得ることができた．

参考文献

[1] M.S.Bazaraa and J.J.Jarvis, *Linear Programming and Network Flows*, Wiley (1977)

[2] D.P.Bertsekas, P.A.Hosein and P.Tseng, "Relaxation method for network flow problems with convex arc costs", *SIAM J. on Control and Optimization* 25, pp.1219-1243 (1987)

[3] R.Fletcher, *Practical Methods of Optimization*, Second Edition, Wiley (1987)

[4] 福島雅夫，新井直子，茨木俊秀，"非線形最小費用流問題に対する内点法"，システムと制御，31巻 pp.837-843 (1987)

[5] 茨木智，"非線形最小費用流問題に対する共役勾配法を用いたニュートン法"，京都大学工学部数理工学科特別研究報告書（昭和 63 年）

[6] R.Katsura, M.Fukushima and T.Ibaraki, "Interior methods for nonlinear minimum cost network flow problems", *J. of the Operations Research Society of Japan* 32, pp.174-199 (1989)

[7] J.G.Klincewicz, "A Newton method for convex separable network flow problems", *Networks* 13, pp.427-442 (1983)

[8] R.T.Rockafeller, *Convex Analysis*, Princeton University Press (1970)

[9] R.T.Rockafeller, *Network Flows and Monotropic Programming*, Wiley (1983)

[10] P.Tseng and D.P.Bertsekas, Relaxation method for problems with strictly convex separable costs and linear constraints", *Mathematical Programming* 38 (1987)

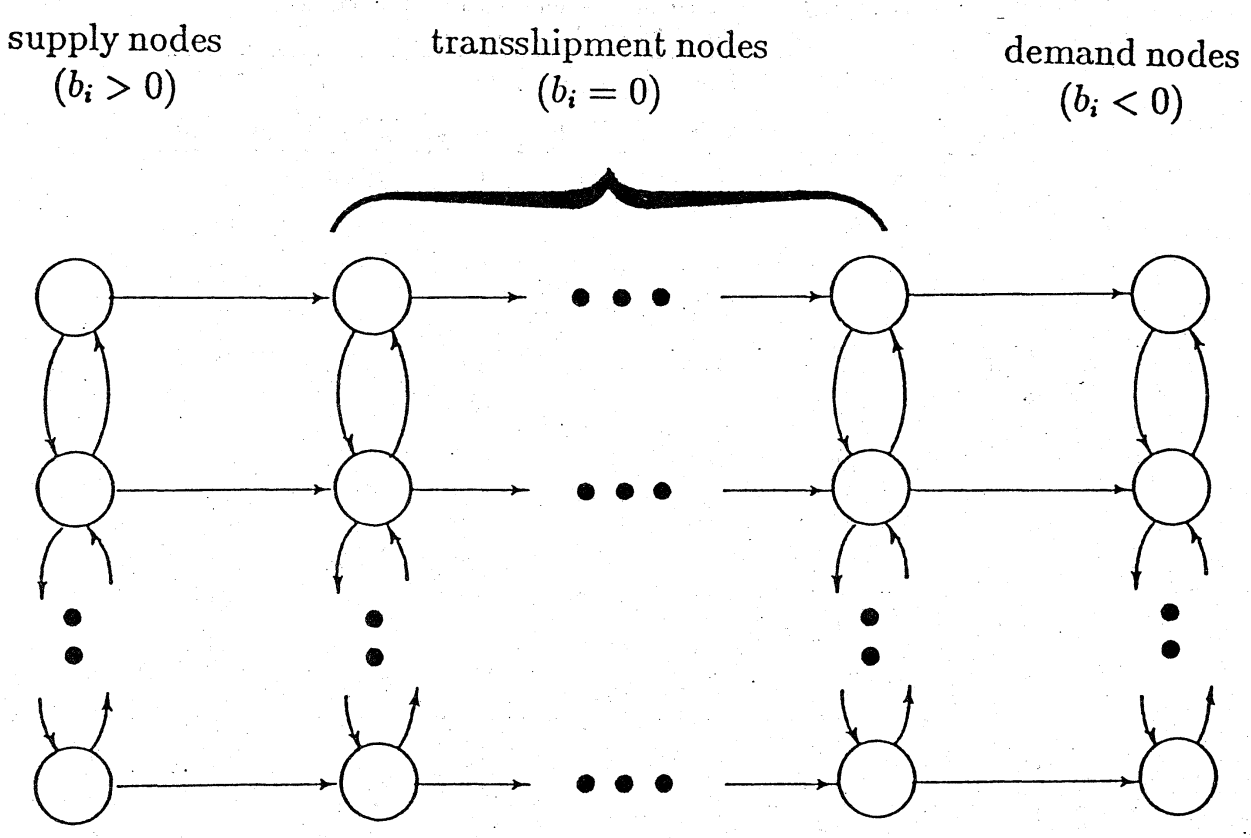

Fig. 1 Lattice network

Table 1: Results for test problems with cost functions defined by (7.1): $\epsilon = 10^{-3}$

Problem	Number	Number	$\epsilon_{CG} = 10^{-1}$		$\epsilon_{CG} = 10^{-3}$	
number	of nodes	of arcs	CPU(s)	iteration	CPU(s)	iteration
1-1	30	73	0.06	16	0.09	16
1-2	30	73	0.08	16	0.08	15
1-3	56	146	0.24	26	0.29	18
1-4	56	146	0.17	17	0.38	21
1-5	121	330	1.34	33	2.44	34
1-6	121	330	1.04	30	2.17	33
1-7	256	720	4.54	42	9.63	38
1-8	256	720	4.84	38	11.7	44
1-9	529	1518	15.7	53	46.9	55
1-10	529	1518	15.2	47	41.5	47
1-11	1024	2976	69.9	77	224	74
1-12	1024	2976	64.2	61	181	60

Table 2: Results for test problems with cost functions defined by (7.2): $\epsilon = 10^{-3}$

Problem	Number	Number	$\epsilon_{CG} = 10^{-1}$		$\epsilon_{CG} = 10^{-3}$	
number	of nodes	of arcs	CPU(s)	iteration	CPU(s)	iteration
3-1	30	73	0.04	14	0.06	13
3-2	30	73	0.06	17	0.08	15
3-3	56	146	0.19	24	0.24	20
3-4	56	146	0.21	24	0.32	22
3-5	121	330	0.96	35	1.83	32
3-6	121	330	0.60	26	1.00	19
3-7	256	720	2.90	37	6.12	31
3-8	256	720	2.51	38	5.74	28
3-9	529	1518	8.75	47	21.5	34
3-10	529	1518	10.7	55	26.7	42
3-11	1024	2976	23.4	48	98.5	43
3-12	1024	2976	26.7	57	88.0	48

Table 3: Results for test problems with cost functions defined by (7.3): ($\epsilon = 10^{-3}$)

Problem	Number	Number	$\epsilon_{CG} = 10^{-1}$	
number	of nodes	of arcs	CPU time(s)	ite.
3-1	30	73	0.23	47
3-2	30	73	0.46	32
3-3	56	146	0.98	62
3-4	56	146	1.01	64
3-5	121	330	7.85	107
3-6	121	330	6.01	86
3-7	256	720	58.0	166
3-8	256	720	38.6	117
3-9	529	1518	308	210
3-10	529	1518	334	216

Table 4: Detailed results for problems 1-11 and 1-12

	problem 1-11		problem 1-12	
	$\epsilon_{CG} = 10^{-1}$	$\epsilon_{CG} = 10^{-3}$	$\epsilon_{CG} = 10^{-1}$	$\epsilon_{CG} = 10^{-3}$
(a) iteration				
$\epsilon = 10^{-1}$	55	65	46	53
10^{-2}	11/66	4/69	8/54	3/56
10^{-3}	11/77	5/74	7/61	4/60
10^{-4}	2/79	2/76	3/65	2/62
(b) CPU time(s)				
$\epsilon = 10^{-1}$	60.7	215.9	58.9	175.9
10^{-2}	4.6/65.3	3.4/219.3	3.4/62.3	2.6/178.5
10^{-3}	4.6/69.9	5.4/224.7	1.9/64.2	2.3/180.8
10^{-4}	0.4/70.3	1.8/226.5	0.4/64.6	1.7/182.5

Table 5: Detailed results for problems 2-11 and 2-12

	problem 2-11		problem 2-12	
	$\epsilon_{CG} = 10^{-1}$	$\epsilon_{CG} = 10^{-3}$	$\epsilon_{CG} = 10^{-1}$	$\epsilon_{CG} = 10^{-3}$
(a) iteration				
$\epsilon = 10^{-1}$	24	27	26	29
10^{-2}	13/37	9/36	11/37	10/39
10^{-3}	11/48	6/43	20/57	9/48
10^{-4}	12/60	3/46	14/71	5/52
(b) CPU time(s)				
$\epsilon = 10^{-1}$	18.2	80.9	20.9	72.0
10^{-2}	2.6/20.8	9.9/90.8	1.7/22.6	8.0/80.0
10^{-3}	2.6/23.4	7.7/98.5	4.1/26.8	7.9/87.9
10^{-4}	2.2/26.7	3.1/101.6	4.7/31.4	3.7/91.7

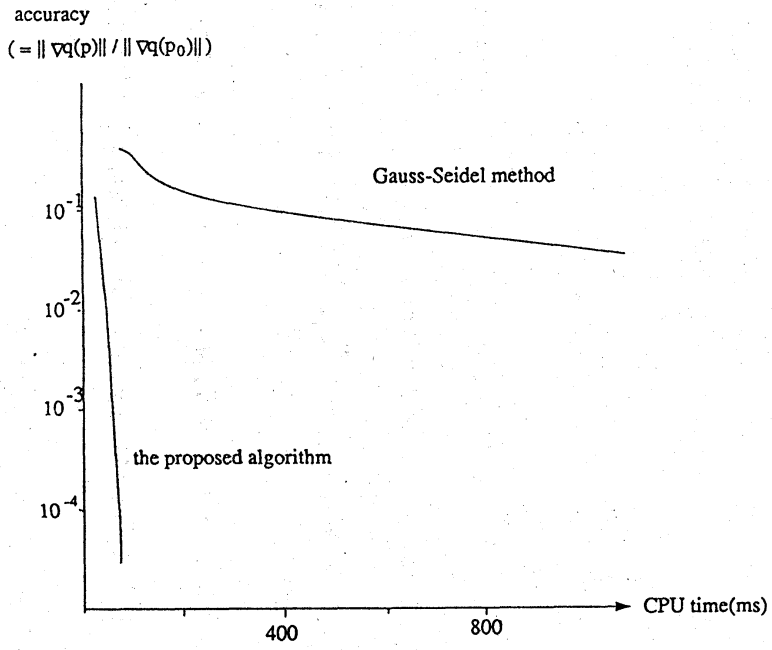

Fig. 2 Comparison of the proposed algorithm and Gauss-Seidel method for test problem 2-1.

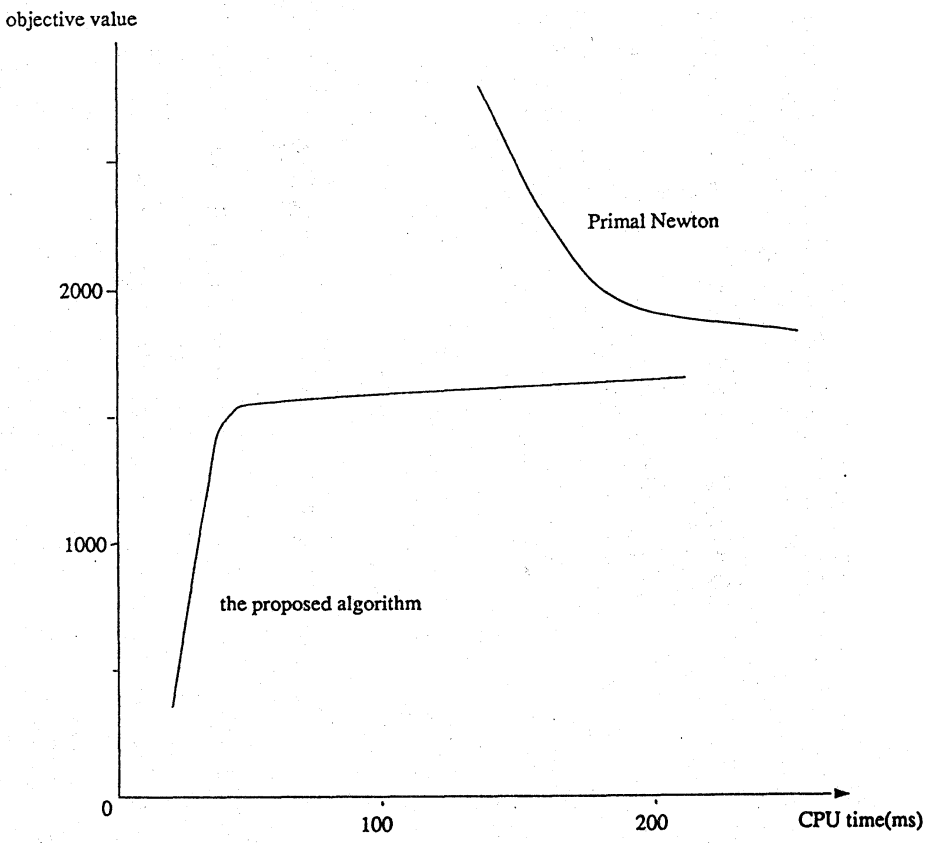

Fig. 3 Comparison of the proposed algorithm with the primal Newton method for test problem 1-1.