A hybrid algorithm for the solution of a single commodity spatial equilibrium model

L. Portugal * J. Júdice †

Abstract

In this paper we propose a hybrid algorithm for the solution of a large-scale single commodity spatial equilibrium model. This model can be stated as a Linear Complementarity Problem (LCP) with a singular Symmetric Positive Semi-Definite (SPSD) matrix whose structure is closely related with the network structure of the model. The hybrid scheme is a combination of a Predictor-Corrector (PC) and a Parametric Principal Pivoting (PPP) algorithms and its implementation takes full advantage of the structure of the matrix of the LCP. We report computational experience on the solution of large-scale spatial equilibrium problems with up to 1000 regions that shows the great efficiency of the approach discussed in this paper.

1 Introduction

During the last decades, several spatial equilibrium models have been studied in the literature. Some of these models can be formulated as linear complementarity problems by assuming that the supply and demand functions are linear [2, 7, 10, 18, 19]. In this paper we discuss the single commodity spatial equilibrium model belonging to this class studied by Glassey [7] and Pang and Lee [19]. In this model there exist n separate regions which trade a certain commodity among them. In each region i, the price p_i and the net import y_i of such commodity are related by

$$p_i = a_i - b_i y_i, \tag{1}$$

where a_i is the equilibrium price in the absence of imports and exports and b_i is a positive parameter related to the elasticity of demand and supply. The net import y_i of the region i is given by

$$y_i = \sum_{j=1}^{n} x_{ji} - \sum_{j=1}^{n} x_{ij}, \tag{2}$$

where x_{ij} is the trade flow from region i to region j.

Let c_{ij} be the unit transportation cost from region i to region j. As stated in [19], the problem is to find p_i, y_i and x_{ij} satisfying

$$\begin{aligned} p_i + c_{ij} - p_j &\geq 0, \\ x_{ij} &\geq 0, \\ x_{ij}(p_i + c_{ij} - p_j) &= 0, \end{aligned} \tag{3}$$

*Departamento de Ciências da Terra, Universidade de Coimbra, 3000 - Coimbra, Portugal

†Departamento de Matemática, Universidade de Coimbra, 3000 - Coimbra, Portugal

for all $i, j = 1, \cdots, n$. We assume that the triangle inequality

$$c_{ij} \leq c_{ik} + c_{kj},$$

holds for all $i, j, k = 1, \cdots, n$ and consequently transshipment of flows does not occur. We also assume, without loss of generality, that $x_{ij} = 0$ for all $i = j$.

By eliminating the variables $y_i, i = 1, \cdots, n$ and introducing the variables z_{ij} defined by

$$z_{ij} = p_i + c_{ij} - p_j \quad i, j = 1, \cdots n,$$

we may state the conditions (1) and (3) as the monotone Linear Complementarity Problem (LCP)

$$\begin{array}{rcll} z_{ij} & = & \sum_{k=1}^{n} [b_i(x_{ik} - x_{ki}) - b_j(x_{jk} - x_{kj})] + (a_i - a_j + c_{ij}) & i, j = 1, \cdots, n, \\ z_{ij} & \geq & 0 & i, j = 1, \cdots, n, \\ x_{ij} & \geq & 0 & i, j = 1, \cdots, n, \\ x_{ij} z_{ij} & = & 0 & i, j = 1, \cdots, n. \end{array} \tag{4}$$

An important caractheristic of this model is its network structure. We may represent the model by a directed graph, where the node i represents the region i and the directed arc (i, j) represents the connection from the region i to the region j. The solution of the LCP (4) deals with a complete graph, since its formulation considers the variables x_{ij} and z_{ij} for all $i, j = 1, \cdots, n$. Pang and Lee [19] have shown that if $a_i - a_j + c_{ij} > 0$ then the equality $x_{ij} = 0$ holds. We can take this result into account to reduce the dimension of the LCP (4) to the number of elements of the set $H = \{(i, j) : a_i - a_j + c_{ij} \leq 0\}$. Hence we can reformulate the LCP (4) by introducing a directed subgraph $\Psi = (N, H)$, where N denotes the set of nodes and the set H mentioned before is a subset of directed arcs.

If $|H|$ represents the number of edges of the set H, we may restate the LCP (4) in the following form:

$$\begin{array}{rcl} z & = & G^T BGx - G^T a + c, \\ x & \geq & 0, \\ z & \geq & 0, \\ x^T z & = & 0, \end{array} \tag{5}$$

where $G^T BG$ is a singular Symmetric Positive Semi-Definite (SPSD) matrix of order $|H|$, $G \in \mathbb{R}^{n \times |H|}$ is the node-arc incidence matrix related to the flow conservation equations in the graph $\Psi = (N, H)$, $x, z, c \in \mathbb{R}^{|H|}$ are the vectors whose elements are respectively x_{ij}, z_{ij} and c_{ij} such that $(i, j) \in H$, $a \in \mathbb{R}^n$ is a vector with components a_i, and $B \in \mathbb{R}^{n \times n}$ is a diagonal matrix with diagonal elements b_i.

In 1978, Glassey [7] introduced a tree growing algorithm for the solution of the LCP (5). To our knowledge, this algorithm is the first effective technique that takes advantage of the network structure of this problem. This method attains to find a solution of the LCP (5) where the trade routes of positive flow form a forest. A solution of this type is proved to exist [7] and is also investigated by the hybrid algorithm proposed in this paper.

Pang and Lee [19] have proved that Glassey's algorithm fails to work in certain cases and proposed a Parametric Principal Pivoting (PPP) algorithm for the solution of the LCP (5). They have shown that Graves' method [9] for LCPs with SPSD matrices simplifies in this case into a simple algorithm that can be implemented by fully exploiting the network structure of the LCP (5). As we prove in the last section of this paper, this pivoting technique is slow for large-scale LCPs, since the set of active constraints is modified in exactly one element per iteration. However, the PPP algorithm has the ability to start with any basic solution. This nice property is exploited in the design of the hybrid algorithm proposed in this paper.

In 1984, N. Karmarkar [13] introduced a new polynomial time algorithm for linear programming. Later, some variants of this algorithm, known as interior point methods, were extended to the solution of linear complementarity problems [14]. Nowadays, it is well accepted that Mehrotra's [16] Predictor Corrector (PC) interior-point algorithm is an efficient procedure to solve large-scale LCPs with symmetric positive definite and semi-definite matrices [3, 5, 20, 22]. The bulk of the PC algorithm is the computation at each iteration of a search direction by solving two linear systems of normal equations. In the general case, the algorithm incorporates a sparse direct solver for this purpose. In the case of the LCP (5), this technique is not appropriate and the special structure of the problem should be exploited to design an iterative linear system solver [12].

Fernandes et al. [4] described an implementation of the predictor-corrector method to solve the LCP (5) that incorporates diagonal and block-diagonal Preconditioned Conjugate-Gradient (PCG) techniques to compute the search direction at each iteration of the PC method. The resulting implementation has shown to have great potential and to be consistently superior over the projected Sucessive Over Relaxation (SOR) algorithm discussed in [10].

In [25], the most efficient Preconditioned Conjugate-Gradient (PCG) based algorithms for linear network flow problems are described. The Incomplete QR decomposition introduced in Portugal et al. [21] is among them and has proven to be superior over the other preconditioning techniques for the solution of large-scale dense transportation problems.

In this paper, we show that the IQRD preconditioning technique can also be applied successfully in this case. In fact, the network associated with the model is sometimes very dense and, as in linear network programming problems, the LCP (5) has at least one solution such that the trade routes of positive flow form a forest [7]. Furthermore, we show that the $|H| \times |H|$ system of normal equations that computes the search direction in this case can be reduced into a linear system of order n with a structure similar to the structure of the system of normal equations that arises in linear transportation problems. Extensive computational experience has shown that this implementation of the PC algorithm is able to identify quickly an approximate basic feasible solution to the LCP (5) but has difficulties in achieving a solution to this problem when the dimension is quite large. In linear programming, a hybrid scheme is sometimes used to overcome this difficulty [26, 27]. In this procedure, an interior point algorithm is first applied and a switch to the simplex method is done after a feasible basis has been found. Similarly, after computing a basic feasible solution to the LCP (5) by the PC algorithm, we find a solution of this problem by using the PPP algorithm.

Before concluding this introduction, we present some notation and outline the remainder of the paper. We denote by G_I (by x_I) the submatrix of G (the subvector of x) constituted by columns with indices in the set I (by elements with indices in the set I). The number of elements of the set I is represented by $|I|$. When I denotes a set of edges and $I = \{(i,j)\}$ with i and j representing two distinct nodes, G_I is sometimes donoted by G_{ij}. In this case, the element in the row l of G_{ij} is represented by G_{lij}. We represent a vector of ones by e. Let $x \in \mathbb{R}^n$. We denote by X the $n \times n$ diagonal matrix having the components of x in the diagonal.

The organization of this paper is as follows: Section 2 introduces the PPP algorithm and briefly discusses its implementation. The PC algorithm and its implementation are presented in Section 3. Finally, the computational experience and some concluding remarks are reported in the last section of this paper.

2 The parametric principal pivoting algorithm

2.1 The algorithm

The Parametric Principal Pivoting (PPP) algorithm was first introduced by Graves [9]. Later, Pang and Lee [19] constructed a special-purpose implementation of this method for the solution of the LCP (5). It is beyond the scope of this paper to discuss in detail the general parametric principal pivoting algorithm. In this section we restrict ourselves to briefly describe the specialized algorithm and its implementation. Additionally, we report the most important features of the matrix $G^T BG$ that enable the design of an efficient implementation for the PPP algorithm.

The PPP algorithm has been designed to solve a sequence of LCPs of the form

$$\begin{aligned} z &= G^T BGx - G^T a + c + \lambda p, \\ x &\geq 0, \\ z &\geq 0, \\ x^T z &= 0, \end{aligned} \tag{6}$$

for a given vector $p > 0$ and decreasing values of λ until a solution of the LCP (6) with $\lambda = 0$ has been computed. This is obviously a solution for the LCP (5).

Let I and J be two subsets of H such that $I \cap J = \emptyset$ and $I \cup J = H$. Furthermore, consider the following partition of the matrix G:

$$G = [G_I \; G_J],$$

where $G_I \in \mathbb{R}^{n\times|I|}$, $G_J \in \mathbb{R}^{n\times|J|}$ and $|I|$ and $|J|$ are the number of columns of G_I and G_J, respectively. Based on this partition, we can rewrite the first equality in (6) as

$$\begin{bmatrix} z_I \\ z_J \end{bmatrix} = \begin{bmatrix} G_I^T BG_I & G_I^T BG_J \\ G_J^T BG_I & G_J^T BG_J \end{bmatrix} \begin{bmatrix} x_I \\ x_J \end{bmatrix} + \begin{bmatrix} q_I \\ q_J \end{bmatrix} + \lambda \begin{bmatrix} p_I \\ p_J \end{bmatrix},$$

where $x_I, z_I, q_I, p_I \in \mathbb{R}^{|I|}$, $x_J, z_J, q_J, p_J \in \mathbb{R}^{|J|}$, $x = (x_I, x_J)$, $z = (z_I, z_J)$, $q = c - G^T a = (q_I, q_J)$ and $p = (p_I, p_J)$.

Now, let $\bar{q} = (\bar{q}_I, \bar{q}_J)$ and $\bar{p} = (\bar{p}_I, \bar{p}_J)$ be two vectors satisfying

$$(G_I^T BG_I)\bar{q}_I = -q_I, \tag{7}$$
$$(G_I^T BG_I)\bar{p}_I = -p_I, \tag{8}$$
$$\bar{q}_J = (G_J^T BG_I)\bar{q}_I + q_J, \tag{9}$$
$$\bar{p}_J = (G_J^T BG_I)\bar{p}_I + p_J, \tag{10}$$

and assume that $\bar{q} + \lambda\bar{p} \geq 0$ for $\lambda = \tilde{\lambda}$. Then it is obvious that

$$x = (\bar{q}_I + \tilde{\lambda}\bar{p}_I, 0) \tag{11}$$
$$z = (0, \bar{q}_J + \tilde{\lambda}\bar{p}_J) \tag{12}$$

is a solution of the LCP (6) for all values of λ lying in the interval $[\bar{\lambda}, \tilde{\lambda}]$, where $\bar{\lambda}$ is given by

$$\bar{\lambda} = max\{-\frac{\bar{q}_{ij}}{\bar{p}_{ij}} : \bar{p}_{ij} > 0, (i,j) \in H\}. \tag{13}$$

Furthermore, if $\bar{p}_{ij} \leq 0$ for all $(i,j) \in H$, then the vectors x and z given by (11) and (12) constitute a solution of the LCP (6) for $\lambda \in]-\infty, \tilde{\lambda}]$.

In the PPP algorithm the initial vector $\bar{p}$ is chosen to be strictly positive so that the inequality $\bar{q} + \lambda\bar{p} \geq 0$ holds for some value of λ sufficiently large. Furthermore, in each iteration of the PPP algorithm we use the formulas (7)-(10) and (13) to compute the vectors $\bar{q}$ and $\bar{p}$, and to obtain the value of the parameter $\bar{\lambda}$. If $\bar{\lambda} \leq 0$ or $\bar{p}_{ij} \leq 0$ for all $(i,j) \in H$, then a solution of the LCP (5) is given by $x = (\bar{q}_I, 0)$ and $z = (0, \bar{q}_J)$ and the algorithm stops. If $\bar{\lambda} > 0$, then the sets I and J are modified according to the following rules:

$$\begin{aligned} I &= \begin{cases} I \cup \{(\alpha,\beta)\} & \text{if } (\alpha,\beta) \in J \\ I - \{(\alpha,\beta)\} & \text{if } (\alpha,\beta) \in I, \end{cases} \\ J &= H - I, \end{aligned} \tag{14}$$

where

$$(\alpha,\beta) = argmax\{-\frac{\bar{q}_{ij}}{\bar{p}_{ij}} : \bar{p}_{ij} > 0, (i,j) \in H\}. \tag{15}$$

Then another iteration of the procedure takes place.

This technique is a particular case of the pivoting procedure developed by Graves [9] for the solution of LCPs with positive semi-definite matrices. This last algorithm may require 2×2 pivot operations to converge, that is, the sets I and J may change in 2 elements in some iterations. However, Pang and Lee have proved that, in the case of the LCP (6), if the vector $\bar{p}$ associated to the initial solution is chosen to be a vector of ones, then the algorithm proposed by Graves never uses pivots of order 2 and the modifications in the sets I and J always satisfy (14). Pang and Lee have also proved that the matrix G_I is in every stage of the algorithm the node-arc incidence matrix of a forest where transshipment of flows does not occur. This property is quite important since it implies that the matrix $G_I^T BG_I$ is always nonsingular and the vectors $\bar{q}_I$ and $\bar{p}_I$ are uniquely defined in each iteration by formulas (7) and (8) (see [19] for the details).

The steps of the resulting PPP algorithm are stated below.

PPP algorithm

Let I and J be two subsets of H such that:

i) $I \cap J = \emptyset$.

ii) $I \cup J = H$.

iii) the set of nodes N together with the set of arcs I form a forest where transshipment of flows does not occur.

Set

$\bar{p}_I = e$,
$\bar{p}_J = e$,
$p_I = -(G_I^T BG_I)\bar{p}_I$,
$p_J = \bar{p}_J - (G_J^T BG_I)\bar{p}_I$,
$\bar{q}_I = -(G_I^T BG_I)^{-1} q_I$,
$\bar{q}_J = (G_J^T BG_I)\bar{q}_I + q_J$.

Step 1. Compute $\bar{\lambda}$ by (13) and find the arc index (α, β) by (15).

Step 2. If $\bar{\lambda} \leq 0$ stop with $x = (\bar{q}_I, 0)$ and $z = (0, \bar{q}_J)$ a solution of the LCP (5).

Step 3. Update the sets I and J by (14) and compute $\bar{q}$ and $\bar{p}$ by (7)-(10). Go to Step 1.

The most important issues of the PPP algorithm are the computations (7)-(10) and the definition of the initial sets I and J. We discuss the first point in the next subsection. The most obvious choice for the initial sets I and J is $I = \emptyset$ and $J = H$. In practice, this choice seems to be suited for small intances of the LCP (5). However, when the value of $|H|$ increases the number of iterations of the algorithm grows in a quite fast way and a better approximation of the solution is required. As stated before, we propose in Section 3 an efficient technique for this purpose.

2.2 Implementation issues

We have implemented the procedure proposed in [19] to perform the computations (7)-(10). In each iteration of the PPP algorithm the vectors $\bar{q}_I$ and $\bar{p}_I$ are updated by using their corresponding values at the previous iteration and then $\bar{q}_J$ and $\bar{p}_J$ are computed from (9)-(10). Next we explain how this is done.

Let I be a subset of H and suppose that the vectors $v, r^1 \in \mathbb{R}^{|I|}$ verify the equality $(G_I^T B G_I)v = r^1$. Furthermore, let (α, β) be an element of H such that $\{(\alpha, \beta)\} \notin I$. If the matrix $G_I^T B G_I$ is nonsingular, then the solution of a linear system of the form

$$\begin{bmatrix} G_I^T B G_I & G_I^T B G_{\alpha\beta} \\ G_{\alpha\beta}^T B G_I & G_{\alpha\beta}^T B G_{\alpha\beta} \end{bmatrix} \begin{bmatrix} u^1 \\ u_2 \end{bmatrix} = \begin{bmatrix} r^1 \\ r_2 \end{bmatrix}, \tag{16}$$

can be found by the two following steps:

$$\text{(i)} \quad w_1 = (G_{\alpha\beta}^T B G_{\alpha\beta}) - (G_{\alpha\beta}^T B G_I)(G_I^T B G_I)^{-1}(G_I^T B G_{\alpha\beta}), \tag{17}$$

$$w^2 = (G_I^T B G_I)^{-1}(G_I^T B G_{\alpha\beta}), \tag{18}$$

$$\text{(ii)} \quad u_2 = \frac{r_2 - (G_{\alpha\beta}^T B G_I)v}{w_1}, \tag{19}$$

$$u^1 = v - u_2 w^2. \tag{20}$$

On the other hand, if the solution (u^1, u_2) of the linear system (16) is available, we can compute the solution v of $(G_I^T B G_I)v = r^1$ by

$$v = u^1 + u_2 w^2, \tag{21}$$

where w^2 is the vector satisfying (18).

As described in [19], w_1 and w^2 are computed by exploiting the structure of the model. At every stage of the algorithm, the graph constituted by the set N of nodes and the set I of edges is a forest. Furthermore, the forests associated to two consecutive iterations of the PPP algorithm only differ in the arc (α, β). If we wish the arc (α, β) to be moved from the set J to the set I, we add the arc (α, β) to the previous forest and connect the trees where the nodes α and β belong. We denote by $\Upsilon(N_\alpha, H_\alpha)$ and $\Upsilon(N_\beta, H_\beta)$, respectively, the two different trees that contain the nodes α and β before this operation has been performed. On the other hand, if the arc (α,β) is requested to be moved from the set I to the set J, the arc (α,β) is deleted from the previous forest and we split the tree where the nodes α and β belong to into two trees. We denote by $\Upsilon(N_\alpha, H_\alpha)$ and $\Upsilon(N_\beta, H_\beta)$, the two different trees that contain the nodes α and β after the splitting operation has been performed, respectively.

Let the nodes α and β be the roots of the trees $\Upsilon(N_\alpha, H_\alpha)$ and $\Upsilon(N_\beta, H_\beta)$, respectively. Furthermore, assume that the node γ belongs either to N_α or to N_β and let D_γ denote the set of descendents of node γ (see [19] for the definition of set of descendents of a node belonging to a rooted tree). Finally, let the real number $\bar{b}_\gamma$ be defined by

$$\bar{b}_\gamma = \frac{1}{\sum_{i\in D_\gamma}\frac{1}{b_i}},$$

for each node $\gamma \in \{N_\alpha \cup N_\beta\}$. Thus, the scalar w_1 is given by [19]

$$w_1 = \frac{1}{\bar{b}_\alpha} + \frac{1}{\bar{b}_\beta}.$$

Furthermore, if $(\gamma,\delta) \in \{H_\alpha \cup H_\beta\}$ with $\gamma \in D_\delta$, then the component of the vector w^2 associated with the arc (γ,δ) is computed by [19]

$$w^2_{\gamma\delta} = \begin{cases} -\bar{b}_\gamma/\bar{b}_\alpha & \text{if } \gamma \in D_\alpha \\ \bar{b}_\gamma/\bar{b}_\beta & \text{if } \gamma \in D_\beta. \end{cases}$$

Similarly, if $(\delta,\gamma) \in \{H_\alpha \cup H_\beta\}$ with $\gamma \in D_\delta$, then [19]

$$w^2_{\delta\gamma} = \begin{cases} \bar{b}_\gamma/\bar{b}_\alpha & \text{if } \gamma \in D_\alpha \\ -\bar{b}_\gamma/\bar{b}_\beta & \text{if } \gamma \in D_\beta. \end{cases}$$

Finally, if $(\gamma,\delta) \in I$ but $(\gamma,\delta) \notin \{H_\alpha \cup H_\beta\}$ then $w^2_{\gamma\delta} = 0$.

In the Step 3 of the PPP algorithm we first compute w_1 and w^2. Then, if we wish the arc index (α,β) to be moved from the set J to the set I, we use formulas (19) and (20) with $r_2 = -q_{\alpha\beta}$, $v = \bar{q}_I^{old}$, $u^1 = \bar{q}_I^{new}$ and $u_2 = \bar{q}_{\alpha\beta}^{new}$ to update $\bar{q}_I$ and $\bar{q}_{\alpha\beta}$. In this last operation we take into account that in the previous iteration $(\alpha,\beta) \in J$, and by formulas (9) and (10) we have $r_2 - (G^T_{\alpha\beta}BG_I)v = -q_{\alpha\beta} - (G^T_{\alpha\beta}BG_I)\bar{q}_I^{old} = -\bar{q}^{old}_{\alpha\beta}$. So, in this case we first update $\bar{q}_{\alpha\beta}$ and $\bar{q}_I$ by

$$\begin{aligned} \bar{q}^{new}_{\alpha\beta} &= -\frac{\bar{q}^{old}_{\alpha\beta}}{w_1}, \\ \bar{q}^{new}_{\gamma\delta} &= \begin{cases} \bar{q}^{old}_{\gamma\delta} - \bar{q}^{new}_{\alpha\beta} w^2_{\gamma\delta} & \text{if } (\gamma,\delta) \in I \text{ and } (\gamma,\delta) \in \{H_\alpha \cup H_\beta\} \\ \bar{q}^{old}_{\gamma\delta} & \text{if } (\gamma,\delta) \in I \text{ and } (\gamma,\delta) \notin \{H_\alpha \cup H_\beta\}, \end{cases} \end{aligned}$$

and then we set $I = I \cup \{(\alpha,\beta)\}$ and $J = J - \{(\alpha,\beta)\}$. On the other hand, if the arc index (α,β) has to be moved from the set I to the set J, we set $I = I - \{(\alpha,\beta)\}$ and $J = J \cup \{(\alpha,\beta)\}$ and update $\bar{q}_I$ by using formula (21) with $u^1 = \bar{q}_I^{old}$, $u_2 = \bar{q}^{old}_{\alpha\beta}$ and $v = \bar{q}_I^{new}$. In this case we obtain

$$\bar{q}^{new}_{\gamma\delta} = \begin{cases} \bar{q}^{old}_{\gamma\delta} + \bar{q}^{new}_{\alpha\beta} w^2_{\gamma\delta} & \text{if } (\gamma,\delta) \in I \text{ and } (\gamma,\delta) \in \{H_\alpha \cup H_\beta\} \\ \bar{q}^{old}_{\gamma\delta} & \text{if } (\gamma,\delta) \in I \text{ and } (\gamma,\delta) \notin \{H_\alpha \cup H_\beta\}. \end{cases}$$

In both operations $\bar{p}_I$ and $\bar{p}_{\alpha\beta}$ are updated by using similar formulas.

The computation of $\bar{q}_J$ is easy to perform. Since it does not occur transshipment of flows in the forest constituted by the set N of nodes and the set I edges, we may consider in each iteration the following three sets of nodes which form a partition of the set N:

- $N_1 = \{i : i \in N$ and i has no adjacent arcs in $I\}$
- $N_2 = \{i : i \in N$ and i only has adjacent incoming arcs in $I\}$
- $N_3 = \{i : i \in N$ and i only has adjacent outgoing arcs in $I\}$

Now, it follows from (9) that

$$\bar{q}_{\gamma\delta} = q_{\gamma\delta} + (\sum_{(i,\gamma)\in I} b_\gamma \bar{q}_{i\gamma} - \sum_{(\gamma,i)\in I} b_\gamma \bar{q}_{\gamma i}) + (\sum_{(\delta,i)\in I} b_\delta \bar{q}_{\delta i} - \sum_{(i,\delta)\in I} b_\delta \bar{q}_{i\delta}),$$

for all $(\gamma, \delta) \in J$. Then, $\bar{q}_J$ can be obtained without computing the elements $\bar{q}_{\gamma\delta}$ for which $\gamma \in N_1$ and $\delta \in N_1$, since in this case $\bar{q}_{\gamma\delta} = q_{\gamma\delta}$. Furthermore, when it is necessary to compute $\bar{q}_{\gamma\delta}$ we only have to consider the arcs in the set I that are adjacent to the nodes γ and δ. Finally, if both arcs (γ, δ) and (δ, γ) belong to the set J, it is easy to prove that once $\bar{q}_{\gamma\delta}$ has been computed we can obtain $\bar{q}_{\delta\gamma}$ by $\bar{q}_{\delta\gamma} = c_{\delta\gamma} + c_{\gamma\delta} - \bar{q}_{\gamma\delta}$. The vector $\bar{p}_J$ is computed by using a procedure similar to that employed for finding $\bar{q}_J$.

As final remark, we note that the sets of nodes N_1, N_2 and N_3 can be considered to simplify the computation of the parameter $\bar{\lambda}$ and the arc index (α, β) in the Step 1 of the algorithm. Since in each iteration the set of nodes N together with the set of arcs I constitute a forest where transshipment of flows does not occur, we only have to consider in such a computation the element (γ, δ) of the set J to be moved to the set I if $\gamma \in \{N_1 \cup N_3\}$ and $\delta \in \{N_1 \cup N_2\}$.

3 The predictor-corrector interior-point algorithm

3.1 The algorithm

The Predictor Corrector (PC) algorithm has been discussed in the literature as the most efficient interior point algorithm for monotone LCPs [3, 5, 20, 22]. This method attains to find a solution of the LCP (5) by solving the nonlinear system of equations

$$\begin{bmatrix} ZXe \\ c - G^T a + G^T BGXe - Ze \end{bmatrix} = 0,$$

so that the conditions $x \geq 0$ and $z \geq 0$ hold in each iteration. Here, $X = diag(x)$, $Z = diag(z)$ and e is a vector of ones.

The procedure generates a sequence of points $\{(x^k, z^k)\}$ by

$$(x^{k+1}, z^{k+1}) = (x^k, z^k) + \theta_k (u^k, v^k), \tag{22}$$

where (u^k, v^k) is a direction and θ_k is a positive stepsize. The direction (u^k, v^k) is computed by first solving the linear system

$$\begin{bmatrix} Z^k & X^k \\ G^T BG & -I_{|H|} \end{bmatrix} \begin{bmatrix} \hat{u}^k \\ \hat{v}^k \end{bmatrix} = \begin{bmatrix} -Z^k X^k e \\ -G^T BGX^k e + Z^k e - c + G^T a \end{bmatrix}, \tag{23}$$

where $I_{|H|}$ is the identity matrix of order $|H|$. Next (u^k, v^k) is found by

$$(u^k, v^k) = (\hat{u}^k, \hat{v}^k) + (\bar{u}^k, \bar{v}^k),$$

where $(\bar{u}^k, \bar{v}^k)$ satisfies

$$\begin{bmatrix} Z^k & X^k \\ G^T BG & -I_{|H|} \end{bmatrix} \begin{bmatrix} \bar{u}^k \\ \bar{v}^k \end{bmatrix} = \begin{bmatrix} -\hat{U}^k \hat{V}^k e + \mu_k e \\ 0 \end{bmatrix},$$

with $\hat{U}^k = diag(\hat{u}^k)$, $\hat{V}^k = diag(\hat{v}^k)$, and μ_k the so-called central parameter.

The direction (u^k, v^k) may also be computed by solving the linear system

$$\begin{bmatrix} Z^k & X^k \\ G^T BG & -I_{|H|} \end{bmatrix} \begin{bmatrix} u^k \\ v^k \end{bmatrix} = \begin{bmatrix} -Z^k X^k e - \hat{U}^k \hat{V}^k e + \mu_k e \\ -G^T BGX^k e + Z^k e - c + G^T a \end{bmatrix}. \tag{24}$$

We follow the recomendations stated in Carpenter et al. [3] to compute the central parameter μ_k and the stepsize θ_k. Hence μ_k is given by

$$\mu_k = \frac{(x^k + \alpha_k \hat{u}^k)^T (z^k + \alpha_k \hat{v}^k)}{|H|^2}, \tag{25}$$

where $\alpha_k = 0.99995 \times \min\{\alpha_k^1, \alpha_k^2\}$ and

$$\begin{aligned} \alpha_k^1 &= \min\{-\frac{x_{ij}^k}{\hat{u}_{ij}^k} : (i,j) \in H \text{ and } \hat{u}_{ij}^k < 0\}, \\ \alpha_k^2 &= \min\{-\frac{z_{ij}^k}{\hat{v}_{ij}^k} : (i,j) \in H \text{ and } \hat{v}_{ij}^k < 0\}. \end{aligned}$$

On the other hand, the stepsize θ_k is computed by $\theta_k = 0.99995 \times \min\{\theta_k^1, \theta_k^2\}$, where

$$\begin{aligned} \theta_k^1 &= \min\{-\frac{x_{ij}^k}{u_{ij}^k} : (i,j) \in H \text{ and } u_{ij}^k < 0\}, \\ \theta_k^2 &= \min\{-\frac{z_{ij}^k}{v_{ij}^k} : (i,j) \in H \text{ and } v_{ij}^k < 0\}. \end{aligned}$$

The steps of the resulting PC algorithm are stated below.

PC algorithm

Let $k = 0$ and $(x^0, z^0) > 0$.

Repeat

Step 1. Compute $(\hat{u}^k, \hat{v}^k)$ by solving the linear system (23).

Step 2. Let μ_k be given by (25). Compute the direction (u^k, v^k) by solving (24).

Step 3. Compute (x^k, z^k) by (22) and set $k = k + 1$.

Until stopping criterion is verified.

The most important issues of the PC algorithm are the solution of the linear systems (23) and (24) and the choice of a stopping criterion to the PC algorithm that is appropriate in this particular instance. These points are addressed next.

3.2 Implementation issues

In this section we discuss the implementation of the Predictor Corrector (PC) interior-point algorithm. It follows from its description that the main computational effort relies on the solution of the linear systems (23) and (24). These linear systems can both be restated in the following form

$$\begin{bmatrix} Z^k & X^k \\ G^TBG & -I_{|H|} \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix} = \begin{bmatrix} -Z^kX^ke + \bar{\mu}_k e \\ -G^TBGX^ke + Z^ke - c + G^Ta \end{bmatrix}, \tag{26}$$

where $\bar{\mu}_k e = 0$, $s = \hat{u}^k$, $t = \hat{v}^k$ in (23) and $\bar{\mu}_k e = \mu_k e - \hat{U}^k\hat{V}^k e$, $s = u^k$, $t = v^k$ in (24).

The solution of the linear system (26) is usually obtained in two major steps. First, s is computed by solving the system of normal equations

$$(G^TBG + (X^k)^{-1}Z^k)s = -G^TBGX^ke + \bar{\mu}_k(X^k)^{-1}e - c + G^Ta, \tag{27}$$

where $G^TBG + (X^k)^{-1}Z^k$ is a Symmetric Positive Definite (SPD) matrix of order $|H|$. In a second step t is found by

$$t = -Z^ke + \bar{\mu}_k(X^k)^{-1}e - (X^k)^{-1}Z^ks.$$

Until the end of this section we investigate the use of an iterative solver for the solution of the system (27) and discuss an appropriate stopping criterion for the PC algorithm.

3.2.1 The reduced system

The use of a direct method for the solution of the system of normal equations (27) is not recommended when $|H|$ is large. We apply to a conjugate-gradient technique to obtain the solution of (27) indirectly by solving a linear system of order n. The next results establish the equivalence between this reduced linear system and the system of normal equations (27).

Theorem 1 Let $\hat{s}$ be the solution of the linear system

$$(GX^k(Z^k)^{-1}G^T + B^{-1})\hat{s} = d, \tag{28}$$

where

$$d = G(-X^ke + (X^k)(Z^k)^{-1}c - \bar{\mu}_k(Z^k)^{-1}e) + B^{-1}a.$$

Then the solution of the normal equations system (27) is given by

$$s = X^k(Z^k)^{-1}(G^T\hat{s} - c) + (Z^k)^{-1}\bar{\mu}_k e. \tag{29}$$

Proof. The normal equations system (27) is equivalent to the following damped linear least-squares problem:

$$\min_{s\in\mathbb{R}^{|H|}} \frac{1}{2}\left\| \begin{bmatrix} B^{\frac{1}{2}}G \\ (X^k)^{-\frac{1}{2}}(Z^k)^{\frac{1}{2}} \end{bmatrix} s - \begin{bmatrix} -B^{\frac{1}{2}}GX^ke + B^{-\frac{1}{2}}a \\ (X^k)^{-\frac{1}{2}}(Z^k)^{-\frac{1}{2}}\bar{\mu}_k e - (X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}}c \end{bmatrix} \right\|_2^2,$$

where $(X^k)^{\frac{1}{2}}$ and $(Z^k)^{\frac{1}{2}}$ are diagonal matrices whose diagonal elements are the square roots of the components of the vectors x^k and z^k. We may reformulate this least-squares problem as

$$\min_{\bar{s}\in\mathbb{R}^{|H|}} \frac{1}{2}\left\| \begin{bmatrix} B^{\frac{1}{2}}G(X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}} \\ I_{|H|} \end{bmatrix} \bar{s} - \begin{bmatrix} \bar{d} \\ 0 \end{bmatrix} \right\|_2^2,$$

where $\bar{s}$ and $\bar{d}$ are defined by

$$\bar{s} = (X^k)^{-\frac{1}{2}}(Z^k)^{\frac{1}{2}}s - (X^k)^{-\frac{1}{2}}(Z^k)^{-\frac{1}{2}}\bar{\mu}_k e + (X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}}c, \tag{30}$$

and

$$\bar{d} = B^{\frac{1}{2}}G(-X^k e + (X^k)(Z^k)^{-1}c - \bar{\mu}_k(Z^k)^{-1}e) + B^{-\frac{1}{2}}a.$$

This latter least-squares problem is equivalent to the following optimization problem:

$$\begin{array}{ll} & \min_{\bar{s},\tilde{s}} \| \bar{s} \|_2^2 + \| \tilde{s} \|_2^2 \\ \text{subject to} & B^{\frac{1}{2}}G(X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}}\bar{s} + I_n\tilde{s} = \bar{d}. \end{array}$$

The optimal solution of this problem can be found by

$$(B^{\frac{1}{2}}GX^k(Z^k)^{-1}G^T B^{\frac{1}{2}} + I_n)\tilde{s} = \bar{d}, \tag{31}$$

and

$$\bar{s} = (X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}}G^T B^{\frac{1}{2}}\tilde{s}. \tag{32}$$

If we set $\tilde{s} = B^{-\frac{1}{2}}\hat{s}$ and $\bar{d} = B^{\frac{1}{2}}d$ then the linear systems (31) and (28) become equivalent. Furthermore, since $\bar{s}$ is given by

$$\bar{s} = (X^k)^{\frac{1}{2}}(Z^k)^{-\frac{1}{2}}G^T\hat{s}, \tag{33}$$

we obtain the formula (29) by combining the equalities (30) and (33). □

In the implementation of the PC algorithm it is quite useful to exploit the relationship between the model and an undirected graph $\Phi = (N, \bar{H})$, where $\bar{H}$ denotes a set of undirected edges. The undirected edge (i, j) with $i < j$ belongs to $\bar{H}$ if either the directed edge (i, j) or the directed edge (j, i) belong to H. Now, we can define a matrix $\bar{G}$ (a diagonal matrix W) by establishing a one to one correpondence between edges of $\Phi = (N, \bar{H})$ and columns of G (and diagonal elements of W). If $(i, j) \in \bar{H}$, then the column $\bar{G}_{ij}$ of $\bar{G}$ exists and its elements $\bar{G}_{lij}$ are given by

$$\bar{G}_{lij} = \begin{cases} 0 & \text{if} \quad l \neq i \quad \text{and} \quad l \neq j, \\ 1 & \text{if} \quad l = i, \\ -1 & \text{if} \quad l = j. \end{cases}$$

Furthermore, if $(i, j) \in \bar{H}$, at iteration k of the PC algorithm the diagonal entries w_{ij} of W that are associated to the arcs (i, j) of $\Phi = (N, \bar{H})$ are given by

$$w_{ij} = \begin{cases} \frac{x_{ij}^k}{z_{ij}^k} + \frac{x_{ji}^k}{z_{ji}^k} & \text{if} \quad (i,j) \in H \quad \text{and} \quad (j,i) \in H, \\ \frac{x_{ij}^k}{z_{ij}^k} & \text{if} \quad (i,j) \in H \quad \text{and} \quad (j,i) \notin H, \\ \frac{x_{ji}^k}{z_{ji}^k} & \text{if} \quad (i,j) \notin H \quad \text{and} \quad (j,i) \in H. \end{cases}$$

The next result establishes the equivalence between the linear system (28) and a linear system with the matrices $\bar{G}$ and W.

Theorem 2 The linear system (28) is equivalent to

$$(\bar{G}W\bar{G}^T + B^{-1})\hat{s} = d. \tag{34}$$

Proof. If we suppose without loss of generality that the directed graph $\Psi = (N, H)$ is complete, then G can be ordered to $(\bar{G} \;\; -\bar{G})$ and the proof follows immediately. □

The matrix of the system (34) is symmetric positive definite and has a special structure that appeals to an Incomplete QR Decomposition (IQRD) technique [21] to construct a preconditioning matrix. This preconditioning technique has been used to solve other network problems and proved to be quite efficient [21]. Next we introduce the PCG algorithm and the IQRD preconditioning technique that is used to improve its convergence.

3.2.2 The preconditioned conjugate gradient algorithm

The conjugate-gradient method for solving linear systems was first introduced by Hestenes and Stiefel [11]. The preconditioned variant of this algorithm attains to find the solution of the linear system

$$M^{-1}(\bar{G}W\bar{G}^T + B^{-1})\hat{s} = M^{-1}d, \tag{35}$$

where M is the so–called preconditioning matrix. The pseudo-code for the Preconditioned Conjugate Gradient (PCG) algorithm is presented below.

PCG algorithm

$\hat{s}^0 = M^{-1}d$

$r^0 = d - (\bar{G}W\bar{G}^T + B^{-1})\hat{s}^0.$

$y^0 = M^{-1}r^0.$

$p^0 = y^0.$

Repeat

$\quad q^i = (\bar{G}W\bar{G}^T + B^{-1})p^i.$

$\quad \alpha_i = (y^i)^T r^i/(p^i)^T q^i.$

$\quad \hat{s}^{i+1} = \hat{s}^i + \alpha_i p^i.$

$\quad r^{i+1} = r^i - \alpha_i q^i.$

$\quad y^{i+1} = M^{-1}r^{i+1}.$

$\quad \beta_i = (y^{i+1})^T r^{i+1}/(y^i)^T r^i.$

$\quad p^{i+1} = y^{i+1} + \beta_i p^i\ .$

$\quad i = i + 1.$

until stopping criterion is verified.

The main issues of this algorithm are the choice of the preconditioning matrix M, the computation of the product $q^i = (\bar{G}W\bar{G}^T + B^{-1})p^i$ and the stopping criterion. Next we discuss these three points.

The product $q^i = (\bar{G}W\bar{G}^T + B^{-1})p^i$ is computed as follows:

for all $\gamma : \gamma \in N$ **do**

$\quad q^i_\gamma = 0.$

for all $(\gamma, \delta) : (\gamma, \delta) \in \bar{H}$ **do**

$\quad \pi = w_{\gamma\delta}(p^i_\gamma - p^i_\delta).$

$q^i_\gamma = q^i_\gamma + \pi.$

$q^i_\delta = q^i_\delta - \pi.$

for all $\gamma : \gamma \in N$ **do**

$q^i_\gamma = q^i_\gamma + p^i_\gamma / b_\gamma.$

This technique is recommended in this instance since it avoids the explicit computation of the matrix $\bar{G}W\bar{G}^T + B^{-1}$, takes full advantage of the sparsity of the matrices $\bar{G}$, W and B, and does not require additional memory space. This procedure requires $O(\bar{H})$ sums and multiplications and $O(n)$ divisions to compute q^i.

In [23] a stopping criterion based on the norm of the residual vector r^i is proposed for network linear programming problems. This criterion is very efficient in practice and enjoys good theoretical properties. Unfortunately, it cannot be applied in this instance since we solve the reduced system (34) instead of solving the system of normal equations (27) directly. The criterion $(y^i)^T r^i < \varepsilon$, where ε is a small tolerance, has been used in [8] to stop the PCG algorithm. This criterion has been compared with other stopping criteria for the solution of linear transportation problems using the same kind of approach and proved to be quite robust [21] for this type of application. In the computational experience reported in the last section of this paper, we employ this stopping criterion with $\varepsilon = 10^{-8}$.

The preconditioning matrix M should be chosen so that the condition number of $M^{-1}(\bar{G}W\bar{G}^T + B^{-1})$ is smaller than that of $\bar{G}W\bar{G}^T + B^{-1}$. Furthermore, the matrix M should be simple so that the systems $My^{i+1} = r^{i+1}$ are easy to solve. Regarding these conditions one can conclude that M should approximate as much as possible the matrix $R^T R$, where $R^T R$ is the Cholesky decomposition of $\bar{G}W\bar{G}^T + B^{-1}$. The use of an Incomplete QR Decomposition (IQRD) technique to construct a preconditioning matrix has been proposed to solve other network problems [21] and proved to be quite efficient when compared with other approaches. Next we describe how to employ such preconditioning for the solution of the linear system (34).

Let the elements of the matrix W represent weights of the edges of the undirected graph $\Phi = (N, \bar{H})$. Furthermore, let $\hat{H}$ be a subset of $\bar{H}$ and let $\Upsilon = (N, \hat{H})$ represent a maximum spanning tree (forest) of $\Phi = (N, \bar{H})$. As before, we can define a matrix $\hat{G}$ (a diagonal matrix $\hat{W}$) as a submatrix of $\bar{G}$ (of W), by establishing a one to one correspondence between the edges of the tree $\Upsilon = (N, \hat{H})$ and columns of $\hat{G}$ (and diagonal elements of $\hat{W}$). So we can write

$$\begin{aligned}
\bar{G}W\bar{G}^T + B^{-1} &= \begin{bmatrix} B^{-\frac{1}{2}} & \hat{G} & \tilde{G} \end{bmatrix} \begin{bmatrix} I_n & & \\ & \hat{W} & \\ & & \tilde{W} \end{bmatrix} \begin{bmatrix} B^{-\frac{1}{2}} \\ \hat{G} \\ \tilde{G} \end{bmatrix} \\
&= \begin{bmatrix} B^{-\frac{1}{2}} & \hat{G}\hat{W}^{\frac{1}{2}} & \tilde{G}\tilde{W}^{\frac{1}{2}} \end{bmatrix} \begin{bmatrix} B^{-\frac{1}{2}} \\ \hat{W}^{\frac{1}{2}}\hat{G}^T \\ \tilde{W}^{\frac{1}{2}}\tilde{G}^T \end{bmatrix},
\end{aligned}$$

where $\tilde{G}$ and $\tilde{W}$ denote the submatrices of $\bar{G}$ and W, respectively, associated to the edges of $\Phi = (N, \bar{H})$ belonging to the set $\bar{H} - \hat{H}$.

The Cholesky factor R of $\bar{G}W\bar{G}^T + B^{-1}$ can be obtained by computing the QR factorization of the matrix

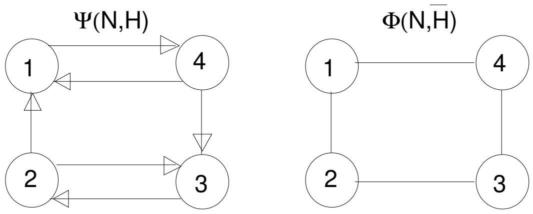
$$A = \left[\begin{array}{c} B^{-\frac{1}{2}} \\ \hat{W}^{\frac{1}{2}}\hat{G}^T \\ \tilde{W}^{\frac{1}{2}}\tilde{G}^T \end{array}\right].$$

In this case this procedure is not recommended because it is expensive to carry out and destroys the sparsity of the matrix A. We propose instead an incomplete QR decomposition of A obtained by applying Givens rotations [6] to this matrix. We consider two phases in this procedure. First the elements of the matrix $\hat{W}^{\frac{1}{2}}\hat{G}^T$ are turned to zero by using the diagonal elements of $B^{-\frac{1}{2}}$. At this stage we only allow fill-in outside the nonzero structure of the diagonal matrix $B^{-\frac{1}{2}}$. In the second phase, the elements of the matrix $\tilde{W}^{\frac{1}{2}}\tilde{G}^T$ are annihilated by using the diagonal elements of the matrix obtained after the first stage. No fill–in is allowed in this last operation. After the incomplete QR decomposition we have

$$M = FDF^T,$$

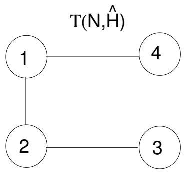
where D is a diagonal matrix and F is a matrix with unit diagonal elements that can be reordered into a triangular form. Next we introduce an illustrating example and then we present a pseudo-code for this procedure.

Let $\Psi = (N, H)$ and $\Phi = (N, \bar{H})$ be the following directed and undirected graphs, respectively, associated with a model with 4 regions:



Now suppose that the maximum spanning tree $\Upsilon = (N, \hat{H})$ and the matrix A are

$$A = \left[\begin{array}{cccc} \frac{1}{\sqrt{b_1}} & & & \\ & \frac{1}{\sqrt{b_2}} & & \\ & & \frac{1}{\sqrt{b_3}} & \\ & & & \frac{1}{\sqrt{b_4}} \\ \cdots & \cdots & \cdots & \cdots \\ \sqrt{w_{12}} & -\sqrt{w_{12}} & & \\ \sqrt{w_{14}} & & & -\sqrt{w_{14}} \\ & \sqrt{w_{23}} & -\sqrt{w_{23}} & \\ \cdots & \cdots & \cdots & \cdots \\ & & \sqrt{w_{34}} & -\sqrt{w_{34}} \end{array}\right] \begin{array}{l} \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} B^{-\frac{1}{2}} \\ \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\} \hat{W}^{\frac{1}{2}}\hat{G}^T \\ \left.\begin{array}{l} \\ \end{array}\right\} \tilde{W}^{\frac{1}{2}}\tilde{G}^T \end{array}$$



We start the procedure by eliminating the element $(5,1)$ using the first diagonal element of $B^{-\frac{1}{2}}$. Note that the element $(5,2)$ is not transformed in this operation (matrix 1).

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}}} & & \boxed{\text{matrix 1}} \\
 & \frac{1}{\sqrt{b_2}} & & \\
 & & \frac{1}{\sqrt{b_3}} & \\
 & & & \frac{1}{\sqrt{b_4}} \\
0 & -\sqrt{w_{12}} & & \\
\sqrt{w_{14}} & & & -\sqrt{w_{14}} \\
 & \sqrt{w_{23}} & -\sqrt{w_{23}} & \\
 & & \sqrt{w_{34}} & -\sqrt{w_{34}}
\end{bmatrix}$$

Then we turn to zero the element $(5,2)$ of matrix 1 by using the element $(2,2)$ (matrix 2).

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}}} & & \boxed{\text{matrix 2}} \\
 & \sqrt{b_2^{-1}+w_{12}} & & \\
 & & \frac{1}{\sqrt{b_3}} & \\
 & & & \frac{1}{\sqrt{b_4}} \\
0 & 0 & & \\
\sqrt{w_{14}} & & & -\sqrt{w_{14}} \\
 & \sqrt{w_{23}} & -\sqrt{w_{23}} & \\
 & & \sqrt{w_{34}} & -\sqrt{w_{34}}
\end{bmatrix}$$

The next step consists of eliminating the element $(6,4)$ of matrix 2 by employing the same procedure. In this operation we do not transform the element $(6,1)$ (matrix 3).

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}}} & & \boxed{\text{matrix 3}} \\
 & \sqrt{b_2^{-1}+w_{12}} & & \\
 & & \frac{1}{\sqrt{b_3}} & \\
\frac{w_{14}}{\sqrt{b_4^{-1}+w_{14}}} & & & \sqrt{b_4^{-1}+w_{14}} \\
0 & 0 & & \\
\sqrt{w_{14}} & & & 0 \\
 & \sqrt{w_{23}} & -\sqrt{w_{23}} & \\
 & & \sqrt{w_{34}} & -\sqrt{w_{34}}
\end{bmatrix}$$

After all the elements of $\hat{W}^{\frac{1}{2}}\hat{G}^T$ are annihilated, the first phase of the procedure is completed (matrix 4).

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}+w_{14}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}+w_{14}}} & & \boxed{\text{matrix } 4} \\
 & \sqrt{b_2^{-1}+w_{12}+w_{23}} & \frac{w_{23}}{\sqrt{b_2^{-1}+w_{12}+w_{23}}} & \\
 & & \sqrt{b_3^{-1}+w_{23}} & \\
\frac{w_{14}}{\sqrt{b_4^{-1}+w_{14}}} & & & \sqrt{b_4^{-1}+w_{14}} \\
0 & 0 & & \\
0 & & & 0 \\
 & 0 & 0 & \\
 & & \sqrt{w_{34}} & -\sqrt{w_{34}}
\end{bmatrix}$$

The second phase of the procedure starts with the elimination of the element $(8,3)$ of matrix 4 by using the element $(3,3)$. We note that we do not allow fill-in in the place $(3,4)$ and do not transform the element $(8,4)$ in this operation (matrix 5).

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}+w_{14}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}+w_{14}}} & & \boxed{\text{matrix } 5} \\
 & \sqrt{b_2^{-1}+w_{12}+w_{23}} & \frac{w_{23}}{\sqrt{b_2^{-1}+w_{12}+w_{23}}} & \\
 & & \sqrt{b_3^{-1}+w_{23}+w_{34}} & \\
\frac{w_{14}}{\sqrt{b_4^{-1}+w_{14}}} & & & \sqrt{b_4^{-1}+w_{14}} \\
0 & 0 & & \\
0 & & & 0 \\
 & 0 & 0 & \\
 & & 0 & -\sqrt{w_{34}}
\end{bmatrix}$$

After turning to zero the element $(8,4)$ of matrix 5 the entire procedure is completed and the following matrix $D^{\frac{1}{2}}F^T$ is obtained:

$$\begin{bmatrix}
\sqrt{b_1^{-1}+w_{12}+w_{14}} & \frac{w_{12}}{\sqrt{b_1^{-1}+w_{12}+w_{14}}} & & \boxed{\text{matrix } D^{\frac{1}{2}}F^T} \\
 & \sqrt{b_2^{-1}+w_{12}+w_{23}} & \frac{w_{23}}{\sqrt{b_2^{-1}+w_{12}+w_{23}}} & \\
 & & \sqrt{b_3^{-1}+w_{23}+w_{34}} & \\
\frac{w_{14}}{\sqrt{b_4^{-1}+w_{14}+w_{34}}} & & & \sqrt{b_4^{-1}+w_{14}+w_{34}}
\end{bmatrix}$$

So for this example

$$D = \begin{bmatrix}
b_1^{-1}+w_{12}+w_{14} & & & \\
 & b_2^{-1}+w_{12}+w_{23} & & \\
 & & b_3^{-1}+w_{23}+w_{34} & \\
 & & & b_4^{-1}+w_{14}+w_{34}
\end{bmatrix}$$

and

$$F^T = \begin{bmatrix} 1 & \frac{w_{12}}{b_1^{-1}+w_{12}+w_{14}} & & \\ & 1 & \frac{w_{23}}{b_2^{-1}+w_{12}+w_{23}} & \\ & & 1 & \\ \frac{w_{14}}{b_4^{-1}+w_{14}+w_{34}} & & & 1 \end{bmatrix}$$

We note that in order to avoid square root operations the matrices D and F are obtained without computing $D^{\frac{1}{2}}F^T$. Suppose that $\Upsilon = (N, \hat{H})$ is a rooted maximum spanning tree (in the example above the root node is node 3) and let $pred(\gamma)$ represent the predecessor node of node γ in $\Upsilon = (N, \hat{H})$. Then, the procedure used to compute the nonzero elements of matrix D and the off-diagonal elements of matrix F can be presented as follows:

for all $\gamma : \gamma \in N$ **do**

$d_{\gamma\gamma} = b_\gamma^{-1}$.

for all $(\gamma, \delta) : (\gamma, \delta) \in \bar{H}$ **do**

$d_{\gamma\gamma} = d_{\gamma\gamma} + w_{\gamma\delta}$.
$d_{\delta\delta} = d_{\delta\delta} + w_{\gamma\delta}$.

for all $(\gamma, \delta) : (\gamma, \delta) \in \hat{H}$ **do**

if $\gamma = pred(\delta)$ **then** $f_{\gamma\delta} = w_{\gamma\delta}/d_{\delta\delta}$, **else** $f_{\delta\gamma} = w_{\gamma\delta}/d_{\gamma\gamma}$.

At each iteration of the PC algorithm, we apply the Fibonacci heap variant of Prim's [24] algorithm described in [1] to compute the maximum spanning tree $\Upsilon = (N, \hat{H})$. This implementation of Prim's algorithm finds a maximum spanning tree in $O(|\bar{H}| + nlog(n))$ time.

The construction of the matrices W, F and D is done once in each iteration of the PC algorithm and requires $O(|\bar{H}|)$ sums and $O(n)$ divisions wich is not too meaningful. For example, it is smaller than the computation of the product $q^i = (\bar{G}W\bar{G}^T + B^{-1})p^i$ that is required in each iteration of the PCG algorithm.

We employ the following procedure to solve the system $My^{i+1} = r^{i+1}$ in the PCG algorithm:

$$Fv = r^{i+1},$$
$$\bar{v} = D^{-1}v,$$
$$F^T y^{i+1} = \bar{v}.$$

This technique, requires $O(n)$ divisions, $O(n)$ multiplications and $O(n)$ subtractions to compute y^{i+1} since D is a diagonal matrix and F can be permuted into a triangular matrix with unit diagonal elements.

3.2.3 stopping criterion

In this section we discuss the stopping criterion for the PC algorithm. The ideia is to stop this interior-point method as soon as it is able to identify a solution of the LCP (5). We note that at least one solution of this problem exists where the trade routes of positive flow are represented by a forest where transshipment of flows does not occur. Thus, when the value of the gap function $z^T x$ becomes sufficiently small, we try to guess such a forest by using a variant of Kruskal's algorithm [15]. In this procedure, we first sort the arcs $(i, j) \in H$ by decreasing value of the parameter x_{ij}^k / z_{ij}^k. Then, we pick-up the arcs one by one and test whether they can belong to the forest or not. To do so, when an arc is considered we check either if a cycle will be formed or transshipment of flows will occur by adding the arc to the forest constituted by the arcs already selected. If so, we discard the arc. Otherwise we add the arc to this forest.

Our computational experience shows that in almost all the problems (and particulary in the larger ones) the PC algorithm generally cannot achieve a point from wich the above technique is able to identify the correspondent forest. However, even when the PC algorithm is stopped earlier, this procedure generally provides a good forest for starting the parametric principal pivoting algorithm. These conclusions lead us to a hybrid algorithm in which the PC algorithm is first used until the relaxed stopping criterium $z^T x \leq 10^{-1}$ is satisfied. Then we compute a forest that is the starting solution of the PPP algorithm. This pivoting procedure is then used to obtain a solution for the LCP (5).

4 Computational experience

In this section we compare the efficiency of the hybrid method introduced in this paper with the Parametric Principal Pivoting (PPP) algorithm with initial sets $I = \emptyset$ and $J = H$.

All the experiments were performed on a SUN SPARCstation 10-52 with 64 Mbytes of RAM. The codes are written in FORTRAN and were compiled with the Sun Fortran Compiler (f77) using flags **-O4 -cg89 -libmil -native**. All the CPU times reported are excluded from input and output time and were measured with the internal function **etime**.

We use a similar technique to that proposed in [19] to generate the test problems. The elements of the vector of costs c are randomly generated in the interval $[0, 80]$ in such a way that the triangular inequality $c_{ij} \leq c_{ik} + c_{ki}$ holds for all $i, j, k = 1, \cdots, n$. The elements of the vectors a and b are also generated to be random numbers in the interval $[0, 100]$.

We have considered test problems with $n = 25, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000$. The results of this study are illustrated in Table 1. The symbols **iter**, **pcgit**, **cpu** and **tcpu** represent the number of iterations required by the algorithms, the average PCG iterations per system solved in the PC algorithm, the CPU time in seconds required by the algorithms, and the total CPU time in seconds spent by the hybrid scheme, respectively.

We have achieved the following conclusions:

- The PPP algorithm is appropriate to solve the LCP (5) if $|H|$ is not too large. In fact, since the computational effort per iteration of the algorithm is small and the algorithm requires a small number of iterations, the total amount of work spent by the algorithm to obtain the solution of the LCP (5) is in those cases acceptable. However, when $|H|$ is large the PPP algorithm requires a huge number of iterations that leads to a substantial amount of work to reach a solution of the LCP (5).

- The IQRD preconditioning together with the transformation of the system of normal equations, constitute a suited technique to solve the linear systems in the PC method. In fact, considering that the order of the original system of normal equations is $|H|$, the PCG algorithm takes on average $0.099|H|$ and $0.0002|H|$ iterations to solve this system in the test problems with 25 and 1000 regions, respectively. This number appears to decrease with n.

Table 1: Computational results.

test problems		PPP algorithm		hybrid algorithm					
				PC phase			PPP phase		
n	$\|H\|$	iter	cpu	pcgit	iter	cpu	iter	cpu	tcpu
25	127	43	0.03	12.56	9	0.12	0	0.00	0.12
50	462	144	0.47	15.28	9	0.44	0	0.00	0.44
100	2117	290	3.85	18.83	11	2.37	3	0.03	2.40
200	7763	874	45.51	23.76	13	11.16	31	0.94	12.10
300	17921	2201	262.47	31.89	14	29.21	84	5.68	34.89
400	30730	2478	539.06	28.45	21	71.23	101	12.31	83.54
500	48931	3036	1060.88	31.12	21	142.39	240	46.69	189.08
600	70033	4037	2046.48	35.00	20	172.99	625	169.26	342.25
700	95337	6690	4687.48	30.41	17	187.79	1076	386.73	574.52
800	125356	6058	5592.20	33.73	20	310.36	607	295.34	605.70
900	157051	7638	8879.73	33.95	20	387.62	673	417.86	805.48
1000	188856	8909	12815.40	39.23	24	642.20	817	620.24	1262.44

- The hybrid algorithm is appropriate to solve the LCP (5) if $|H|$ is large. In fact, in the largest test problems we can spare about 90% in CPU time if, instead of using the PPP algorithm with initial sets $I = \emptyset$ and $J = H$, we first employ the PC method to guess near optimal sets I and J and then switch to the PPP algorithm.

These results clearly show that the hybrid algorithm presented in this paper is recommended for solving this single commodity spatial equilibrium problem. We believe that similar type of methodology can also be useful in the solution the multicommodity spatial equilibrium models that have been presented in the literature [2, 10, 17, 18].

References

[1] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[2] R. Asmuth, B. Eaves, and E. Peterson. Computing economic equilibria on affine networks with Lemke's algorithm. *Mathematics of Operations Research*, 4:209–214, 1979.

[3] T. Carpenter, I. Lustig, J. Mulvey, and D. Shanno. Higher order predictor-corrector interior point methods with applications to quadratic objectives. Technical report, Princeton University, Princeton, New Jersey, USA, 1990.

[4] L. Fernandes, M. Coelho, J. Júdice, and J. Patricio. On the solution of a spatial equilibrium problem. *Investigação Operacional*, 13:119–134, 1993.

[5] L. Fernandes, J. Júdice, and J. Patricio. An investigation of interior-point and block pivoting algorithms for large-scale symmetric monotone linear complementarity problems. Technical report, Department of Mathematics, University of Coimbra, Coimbra, Portugal, 1994. To appear in *Computational Optimization and Applications.*

[6] A. George and M. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra and Its Applications*, 34:69–83, 1980.

[7] C. Glassey. A quadratic network optimization model for equilibrium single commodity trade flows. *Mathematical Programming*, 14:98–107, 1978.

[8] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, USA, 1983.

[9] R. Graves. A principal pivoting simplex algorithm for linear and quadratic programming. *Operations Research*, 15:482–494, 1967.

[10] F. Güder, J. Morris, and S. Yoon. Parallel and serial sucessive overrelaxation for multicommodity spatial price equilibrium problems. *Transportation Science*, 26:48–58, 1992.

[11] M. Hestenes and E. Steifel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards B*, 49:409–436, 1952.

[12] J. Júdice. Algorithms for linear complementarity problems. In E. Spedicato, editor, *Algorithms for Continuous Optimization. The State of the Art*, pages 435–474. Kluwer Academic Publishers, 1993.

[13] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[14] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise. A unified approach to interior point algorithms for linear complementarity problems. In *Lecture Notes in Computer Science 538*. Springer-Verlag, 1991.

[15] J. Kruskal. On the shortest spanning tree of graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[16] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2:575–601, 1992.

[17] A. Nagurney. *Network Economics: A Variational Inequality Approach*. Kluwer Academic Publishers, London, England, 1993.

[18] J. Pang. A hybrid method for the solution of some multicommodity spatial equilibrium problems. *Management Science*, 27:1142–1157, 1981.

[19] J. Pang and P. Lee. A parametric linear complementarity technique for the computation of equilibrium prices in a single commodity spatial model. *Mathematical Programming*, 20:81–102, 1981.

[20] P. Pardalos, Y. Ye, and C. Han. An interior point algorithm for large-scale quadratic problems with box constraints. *Lecture Notes in Control and Information*, 144:413–422, 1990.

[21] L. Portugal, F. Bastos, J. Júdice, J. Paixão, and T. Terlaky. An investigation of interior point algorithms for the linear transportation problem. Technical report, Department of Mathematics, University of Coimbra, Coimbra, Portugal, 1993. To appear in *SIAM Journal on Scientific Computing*.

[22] L. Portugal, J. Júdice, and L. Vicente. A comparison of block pivoting and interior-point algorithms for linear least-squares problems with nonnegative variables. Technical report, Department of Mathematics, University of Coimbra, Coimbra, Portugal, 1992. To appear in *Mathematics of Computation*.

[23] L. Portugal, M. Resende, G. Veiga, and J. Júdice. A truncated primal-infeasible dual-feasible network interior point method. Technical report, Department of Mathematics, University of Coimbra, Coimbra, Portugal, 1994.

[24] R. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[25] M. Resende and P. Pardalos. Interior point algorithms for network flow problems. Technical report, AT&T Bell Laboratories, Murray Hill, New Jersey, USA, 1994. To appear in *Advances in Linear and Integer Programming*, J. Beasley, ed., Oxford University Press, 1995.

[26] D. Shanno. Computational methods for linear programming. In E. Spedicato, editor, *Algorithms for Continuous Optimization. The State of the Art*, pages 383–413. Kluwer Academic Publishers, 1993.

[27] Q. Yeh. *A reduced dual affine scaling algorithm for solving assignment and transportation problems*. PhD thesis, Columbia University, New York, USA, 1989.