MATRIX PARTITIONING METHODS FOR INTERIOR POINT ALGORITHMS

Romesh Saigal
Department of Industrial & Operations Engineering
University of Michigan
Ann Arbor, MI 48109-2117

Technical Report 92-39

June 1992

Matrix Partitioning Methods for interior point Algorithms

Romesh SAIGAL

Department of Industrial and Operations Engineering,

The University of Michigan,

Ann Arbor, Michigan 48109-2117, USA

June 1992

Abstract

We consider here a linear programming problem whose rows of the constraint matrix can be partitioned into two parts. Such natural partitions exist in several special linear programs, including the assignment problem, the transportation problem, the generalized upper bounded variable problem, the block diagonal linear program; and can also be used to induce sparsity patterns in Cholesky factorizations. In this paper we propose a matrix partitioning method for interior point algorithms. The proposed method independently generates Cholesky factorizations of each part, and reduces the complexity to that of solving generally, a dense linear system involving several rank one updates of the indentity matrix. Here, we propose solving this linear system by an inductive use of the Sherman - Morrison - Woodbury formula. The proposed method is easily implemented on a vector, parallel machine as well as on a distributed system. Such partitioning methods have been popular in the context of the simplex method, where the special structure of the basis matrix is exploited.

Key words: Linear Programming, interior point methods, Cholesky factorizations, Matrix Partitioning Methods, Vector-parallel machines, Distributed systems.

Abbreviated title: Matrix Partitioning Methods.

1 Introduction

We consider here the linear programming problem

$$\begin{aligned} \min \quad & \boldsymbol{c}^T \boldsymbol{x} \\ & \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \boldsymbol{0}. \end{aligned}$$

where $\boldsymbol{A}$ is a $m \times n$ matrix of rank m, $\boldsymbol{b}$ is a m vector and $\boldsymbol{c}$ is a n vector. We assume that the matrix $\boldsymbol{A}$ has the partition

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{G} \\ \boldsymbol{H} \end{bmatrix} \quad , \quad \boldsymbol{b} = \begin{bmatrix} \boldsymbol{g} \\ \boldsymbol{h} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{G}$ and $\boldsymbol{H}$, respectively, are $m_1 \times n$ and $m_2 \times n$ matrices, and; $\boldsymbol{g}$ and $\boldsymbol{h}$ are, respectively, m_1 and m_2 vectors, with $m_1 + m_2 = m$. We also assume that such a partition is determined either by the structure of the problem or has been induced by sparsity considerations. Specially structured linear programs that have this *natural* partition include the assignment problem, the transportation problem, the generalized upper bounded variable problem and the block diagonal linear program. Such partitions may also be considered to arise in situations where rows are added, iteratively, to a linear program, and one of the set of rows $\boldsymbol{G}$ or $\boldsymbol{H}$ may be considered to be the ones added.

Many variants of the simplex method exist which exploit the induced special structure of the basis matrix. We refer the reader to Lasdon [6] for a background on these variants. Our goal, in this paper, is to consider the application of the recent interior point methods to this partitioned linear program. The study of these methods was started by the seminal work of Karmarakar [4]. Some notable papers that are related to this work are Barnes [1], Kojima, Mizuno and Yoshise [5] and Vanderbei, Meketon and Freedman [8]. The reader is encouraged to browse through these, the recent book Fang and Puthenpura [2], and many references therein, for an introduction to these methods. Our starting point in this paper is the implementation of these methods, which requires the solution of a linear system

$$\boldsymbol{C}\boldsymbol{z} = \boldsymbol{d} \tag{2}$$

where $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$ for some diagonal matrix $\boldsymbol{D}$; and $\boldsymbol{d}$, as a function of $\boldsymbol{A}$, $\boldsymbol{b}$, $\boldsymbol{c}$ and $\boldsymbol{D}$, is determined by the particular interior point method employed. In this paper, we reduce (2) to solving a $m_2 \times m_2$ system of the form

$$(\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T)\boldsymbol{u} \;=\; \boldsymbol{q} \tag{3}$$

where $\boldsymbol{E}$ is a $m_2 \times m_1$ matrix. If $\boldsymbol{E}_j$ is the jth column of the matrix $\boldsymbol{E}$, it can be readily seen that

$$\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T = \boldsymbol{I} - \sum_{i=1}^{m_1} \boldsymbol{E}_j \boldsymbol{E}_j^T, \tag{4}$$

and we view (4) as m_1 rank one updates to the identity matrix $\boldsymbol{I}$. We solve the system (4) by an inductive version of the Sherman - Morrison - Woodbury formula. This inductive method requires $O(m_2 m_1^2)$ multiplications, and is advantageous over inverting $\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T$ when $m_2 > m_1$. Another advantage of this method is its ready implementation on a vector and a parallel/distributed computing environment.

In section 2 we present partial Cholesky factorization, the basic idea behind the partitioning technique; and, in section 3 we present a technique to handle several rank one updates. In section 4 we present the variant to handle the transportation and assignment problems, in section 5 the GUB and in section 6 the block diagonal linear program and the multicommodity flow problem. Finally in section 7, we give some preliminary computational results comparing the transportation variant with LOQO, Vanderbei [9].

2 Partial Cholesky factorization of ADA^T

In this section we generate the general theory we will use to exploit the partition (1) of $\boldsymbol{A}$. The main result we need for this purpose is the following theorem:

Theorem 1 *Let $\boldsymbol{C}$ be a $m \times m$ symmetric positive definite matrix. Then there exists a unique $m \times m$ lower triangular matrix $\boldsymbol{L}$ with positive diagonal entries, such that $\boldsymbol{C} = \boldsymbol{L}\boldsymbol{L}^T$. In case $\boldsymbol{C}$ is dense, $\frac{1}{6}m^3 + \frac{1}{2}m^2 - \frac{3}{2}m$ multiplications are required to compute $\boldsymbol{L}$.*

Proof : See George and Liu [3].

The matrix $\boldsymbol{L}$ in the above theorem is called the Cholesky factor of $\boldsymbol{C}$. Given that $\boldsymbol{C}$ is sparse, we refer the reader to George and Liu [3], an excellent reference on the methodology that preserves sparsity of $\boldsymbol{L}$. We now use this theorem to exploit the partition (1) of the rows of $\boldsymbol{A}$.

Assuming this partition, it can be easily seen that

$$\boldsymbol{ADA}^T = \begin{bmatrix} \boldsymbol{GDG}^T & \boldsymbol{GDH}^T \\ \boldsymbol{HDG}^T & \boldsymbol{HDH}^T \end{bmatrix}. \tag{5}$$

Then we can prove:

Lemma 2 *There exist lower triangular $m_1 \times m_1$ matrix $\boldsymbol{L}_1$ and $m_2 \times m_2$ matrix $\boldsymbol{L}_3$ such that $\boldsymbol{GDG}^T = \boldsymbol{L}_1\boldsymbol{L}_1^T$ and $\boldsymbol{HDH}^T = \boldsymbol{L}_3\boldsymbol{L}_3^T$.*

Proof : Since $\boldsymbol{A}$ is full row rank so are $\boldsymbol{G}$ and $\boldsymbol{H}$. Since $\boldsymbol{D}$ has positive diagonal entries, $\boldsymbol{GDG}^T$ and $\boldsymbol{HDH}^T$ are symmetric positive definite matrices, and we have our result from theorem (1).

We now use the lower triangular matrices $\boldsymbol{L}_1$ and $\boldsymbol{L}_3$, guaranteed by lemma (2), for solving (2) when $\boldsymbol{A}$ has the partition (1). This is done in the next theorem, which generates a partial Cholesky factor of $\boldsymbol{ADA}^T$.

Theorem 3 *Let the $m_1 \times m_1$ matrix $\boldsymbol{L}_1$ and $m_2 \times m_2$ matrix $\boldsymbol{L}_3$ be defined as in the lemma (2). Then, for the $m_2 \times m_1$ matrix $\boldsymbol{L}_2$ and $m_2 \times m_2$ matrix $\bar{\boldsymbol{D}}$ with*

$$\bar{\boldsymbol{D}} = \boldsymbol{L}_3(\boldsymbol{I} - \boldsymbol{L}_3^{-1}\boldsymbol{L}_2\boldsymbol{L}_2^T\boldsymbol{L}_3^{-T})\boldsymbol{L}_3^T, \tag{6}$$

$$\boldsymbol{L}_1\boldsymbol{L}_2^T = \boldsymbol{GDH}^T, \tag{7}$$

$$\boldsymbol{L} = \begin{bmatrix} \boldsymbol{L}_1 & \boldsymbol{0} \\ \boldsymbol{L}_2 & \boldsymbol{I} \end{bmatrix}, \hat{\boldsymbol{D}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \bar{\boldsymbol{D}} \end{bmatrix};$$

$$\boldsymbol{ADA}^T = \boldsymbol{L}\hat{\boldsymbol{D}}\boldsymbol{L}^T.$$

Proof : Can be readily verified by a direct multiplication of $\boldsymbol{L}$, $\hat{\boldsymbol{D}}$ and $\boldsymbol{L}^T$.

As a result of this theorem, the solution of the equation (2) can be expressed in terms of the matrices $\boldsymbol{L}_1$, $\boldsymbol{L}_2$, and $\boldsymbol{L}_3$. This is done in the next result.

Assume that the partitions

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{z}_1 \\ \boldsymbol{z}_2 \end{bmatrix}, \boldsymbol{d} = \begin{bmatrix} \boldsymbol{d}_1 \\ \boldsymbol{d}_2 \end{bmatrix}$$

conform to the partition (5) of $\boldsymbol{C}$ ($= \boldsymbol{A}\boldsymbol{D}\boldsymbol{A}^T$). Here, $\boldsymbol{d}_1$, $\boldsymbol{z}_1$ are m_1 vectors and $\boldsymbol{d}_2$, $\boldsymbol{z}_2$ are m_2 vectors.

Theorem 4 *Let the $m_2 \times m_1$ matrix $\boldsymbol{E}$ be defined so that $\boldsymbol{L}_3\boldsymbol{E} = \boldsymbol{L}_2$. The equation (2) can be solved by the following sequence of steps:*

Step 1 *Find $\boldsymbol{z}_1'$ by* forward solve

$$\boldsymbol{L}_1\boldsymbol{z}_1' = \boldsymbol{d}_1$$

Step 2 *Define*

$$\boldsymbol{z}_2' = \boldsymbol{d}_2 - \boldsymbol{L}_2\boldsymbol{z}_1'$$

Step 3 *Find $\boldsymbol{z}_2''$ by* forward solve

$$\boldsymbol{L}_3\boldsymbol{z}_2'' = \boldsymbol{z}_2'$$

Step 4 *Solve*

$$(\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T)\boldsymbol{z}_2''' = \boldsymbol{z}_2'' \tag{8}$$

Step 5 *Find $\boldsymbol{z}_2$ by* backward solve

$$\boldsymbol{L}_3^T\boldsymbol{z}_2 = \boldsymbol{z}_2'''$$

Step 6 *Find $\boldsymbol{z}_1$ by* backward solve

$$\boldsymbol{L}_1^T\boldsymbol{z}_1 = \boldsymbol{z}_1' - \boldsymbol{L}_2^T\boldsymbol{z}_2$$

Proof : Using the structure of $\boldsymbol{L}$ and $\hat{\boldsymbol{D}}$ (of theorem (3)) the following equations can be readily derived:

$$\boldsymbol{L}_1\boldsymbol{z}_1' = \boldsymbol{d}_1$$

$$
\begin{aligned}
z_2' &= d_2 - L_2^T z_1' \\
\bar{D} z_2 &= z_2' \\
L_1^T z_1 &= z_1' - L_2^T z_2.
\end{aligned}
$$

The theorem now follows from the structure (6) of $\bar{D}$.

The result of theorem (4) requires the Cholesky factors of GDG^T and HDH^T, which are expected to be less dense than the factor of ADA^T. The price for this enhanced sparsity is paid at Step 4 of the theorem. Here, generally, a dense system of equations has to be solved. In the next section, we will present a procedure that solves this system in $O(m_1^2 m_2)$ multiplications. If this system were solved by Cholesky factorization, calculation of EE^T would require $m_1 m_2^2$ multiplications, and the calculation of the Cholesky factor another $\frac{1}{6}m_2^2 + \frac{1}{2}m_2^2 - \frac{3}{2}m_2$. In the case $m_2 \geq m_1$, solving directly is more expensive.

For a dense matrix C, system (2) can be solved, using Cholesky factors, in $\frac{1}{6}m^3 + \frac{3}{2}m^2 - \frac{5}{2}m$ multiplications. Using the technique of this section with a partitioned matrix A, it requires $\frac{1}{6}(m^3 + 9m^2 + 6m_1^2 m_2 - 3m)$ multiplications, and is not competitive.

We now establish a basic property of the matrix EE^T, encountered in Step 4.

Theorem 5 *Let E be defined as in theorem (4), Step 4. The spectral radius of EE^T is less than 1.*

Proof : Since ADA^T is positive definite, so is $L\hat{D}L^T$ (see theorem (3) for definitions), and thus $\hat{D}$ is positive definite. Using the structure of $\hat{D}$, $\bar{D}$ is positive definite. Thus, from equation (6), $z^T(I - EE^T)z > 0$ for all z. Thus $z^T EE^T z < z^T z$ for all z, and we have our result.

3 A method for system with several rank one updates

In this section we develop an inductive version of the Sherman - Morrison - Woodbury formula for solving the system (8). For each $j = 1, \cdots m_1$, let E_j be the jth column of the matrix E, and

$$
E_{m_1+1} = z_2''.
$$

Also, let $\boldsymbol{B}_1 = \boldsymbol{I}$, and for each $k = 1, \cdots, m_1$

$$\boldsymbol{B}_{k+1} = \boldsymbol{B}_k - \boldsymbol{E}_k \boldsymbol{E}_k^T$$

and, note that $\boldsymbol{B}_{m_1+1} = \boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T$.

Now, for each $k = 1, \cdots, m_1 + 1$ and $j = 1, \cdots, m_1 + 1$ define

$$\boldsymbol{B}_k \boldsymbol{E}_j^{(k)} = \boldsymbol{E}_j.$$

Thus, for each $k = 1, \cdots, m_1$ and $j = k + 1, \cdots, m_1 + 1$,

$$\boldsymbol{B}_{k+1} \boldsymbol{E}_j^{(k+1)} = \boldsymbol{E}_j$$

or,

$$(\boldsymbol{B}_k - \boldsymbol{E}_k \boldsymbol{E}_k^T) \boldsymbol{E}_j^{(k+1)} = \boldsymbol{E}_j$$

or,

$$(\boldsymbol{I} - \boldsymbol{E}_k^{(k)} \boldsymbol{E}_k^T) \boldsymbol{E}_j^{(k+1)} = \boldsymbol{E}_j^{(k)}. \tag{9}$$

Using the Sherman - Morrison - Woodbury formula, we can write the solution to equation (9) as

$$\begin{aligned} \boldsymbol{E}_j^{(k+1)} &= \boldsymbol{E}_j^{(k)} + \frac{< \boldsymbol{E}_j, \boldsymbol{E}_k^{(k)} >}{1 - < \boldsymbol{E}_k, \boldsymbol{E}_k^{(k)} >} \boldsymbol{E}_k^{(k)} \\ &= \boldsymbol{E}_j^{(k)} + \frac{< \boldsymbol{E}_k, \boldsymbol{E}_j^{(k)} >}{1 - < \boldsymbol{E}_k, \boldsymbol{E}_k^{(k)} >} \boldsymbol{E}_k^{(k)}. \end{aligned}$$

where $< ., . >$ is the usual inner product of two vectors.

We can prove the following result about the above procedure:

Theorem 6 *The solution z_2''' of the equation (8) is $\boldsymbol{E}_{m_1+1}^{(m_1+1)}$.*

Proof : This is readily seen since by definition,

$$\boldsymbol{B}_{m_1+1} \boldsymbol{E}_{m_1+1}^{(m_1+1)} = \boldsymbol{E}_{m_1+1}$$

substituting the identities $\boldsymbol{B}_{m_1+1} = \boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T$ and $\boldsymbol{E}_{m_1+1} = \boldsymbol{z}_2''$, we obtain the theorem.

The above inductive procedure (on k) suggests the following algorithm in a vector and a distributed/parallel environment:

Step 1 1. Communicate $\boldsymbol{E}_j$ to processor $j = 1, \cdots, m_1 + 1$.

2. At processor j, set $\boldsymbol{E}_j^{(1)} = \boldsymbol{E}_j$.

3. set $k = 1$.

Step 2 From processor k, communicate $\boldsymbol{E}_k^{(k)}$, $\boldsymbol{E}_k$ and $< \boldsymbol{E}_k, \boldsymbol{E}_k^{(k)} >$ to each processor $j = k+1, \cdots, m_1 + 1$.

Step 3 At processor j, $j = k+1, \cdots, m_1 + 1$ compute

$$\boldsymbol{E}_j^{(k+1)} = \boldsymbol{E}_j^{(k)} + \frac{< \boldsymbol{E}_k, \boldsymbol{E}_j^{(k)} >}{1 - < \boldsymbol{E}_k, \boldsymbol{E}_k^{(k)} >} \boldsymbol{E}_k^{(k)}.$$

Step 4 Set $k = k+1$. If $k \leq m_1$, then go to Step 2, otherwise declare $\boldsymbol{E}_{k+1}^{(k+1)}$ as the solution z_2''' of equation (8).

A careful count of the number of multiplications needed are summarized in the following theorem:

Theorem 7 *The number of multiplications required by the algorithm to solve equation (8) is* $m_1^2 m_2 + 2m_1 m_2$. *In the vector and parallel environment, the number of vector operations required is* $3m_1$.

Proof : Can be readily verified by a careful counting.

There is a considerable advantage in keeping $m_1 < m_2$ while solving equation (8), a $m_2 \times m_2$ system, by the above strategy. Otherwise, it would require $m_1 m_2^2$ multiplications to obtain $\boldsymbol{E}\boldsymbol{E}^T$, $\frac{1}{6}(m_2^3 + 3m_2^2 - 9m_2)$ to obtain the Cholesky factor of $\boldsymbol{I} - \boldsymbol{E}\boldsymbol{E}^T$ and $m_2(m_2+1)$ to solve the system using the Cholesky factor. The method of this section is less advantageous if $m_2 < m_1$; and to use it effectively m_1 and m_2 should be re-defined.

4 Transportation and Assignment Problem

The transportation problem is the following: given m supply depots, with s_i (> 0) as the units of supply of some good at depot i for each $i = 1, \cdots, m$; n demand centers with d_j (> 0) as the units of demand of the good at the center j for each $j = 1, \cdots, n$; and $c_{i,j}$ (> 0) the cost

of shipping one unit of good between depot i and center j for each $i = 1, \cdots, m$, $j = 1, \cdots, n$; find the least cost quantity of good shipped between each depot and center. We assume here that the transportation problem is balanced, or $\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$; i.e., there is just enough quantity of good available at the supply depots to meet this demand. In this case it is well known that the transportation problem has an optimal shipping schedule, and it can be found by solving the following *Transportation Linear Program* :

$$\begin{aligned} \min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{i,j}x_{i,j} & \\ \sum_{j=1}^{n} x_{i,j} &= s_i,\ i = 1, \cdots, m \\ \sum_{i=1}^{m} x_{i,j} &= d_j,\ j = 1, \cdots, n \\ x_{i,j} &\geq 0,\ i = 1, \cdots, m;\ j = 1, \cdots, n. \end{aligned}$$

where $x_{i,j}$ is the number of units of good shipped from depot i to center j, for each i and j.

The Assignment problem is the following: given there are n individuals that can do any of the n tasks; and, assigning the individual i to the task j costs $c_{i,j}$ (> 0) for each $i = 1, \cdots, n$, $j = 1, \cdots, n$, find the least cost assignment of individuals to tasks. It is also required that each individual perform exactly one task, and that each task be performed by exactly one individual. This problem can be cast as a balanced transportation problem, with each individual associated with a supply depot with exactly one unit of supply, and each task with a demand center with exactly one unit of demand. The only difference is that the variables $x_{i,j}$ are required to take on values 0 or 1. As is well known, because of the total unimodularity property of the constraint matrix, setting $s_i = 1$ and $d_j = 1$ for each $i = 1, \cdots, n$ and $j = 1, \ldots, n$ and solving the transportation linear program suffices finds the optimal assignment.

The dual of this linear program is the following:

$$\begin{aligned} \max \sum_{i=1}^{m} s_i u_i + \sum_{j=1}^{n} d_j v_j & \\ u_i + v_j + s_{i,j} &= c_{i,j} \text{ for all } i = 1, \cdots m,\ j = 1, \cdots, n \\ s_{i,j} &\geq 0 \text{ for all } i = 1, \cdots m,\ j = 1, \cdots, n \end{aligned}$$

where u_i and v_j are the dual variables and $s_{i,j}$ are the dual slacks for each i and j.

The constraints of the (primal) linear program fall naturally into two sets, the supply constraints and the demand constraints. It is this partition that we will exploit in the algorithm. As is well known, the rank of the constraint matrix is $m + n - 1$, which is one less than the number of constraints. Thus one constraint must be discarded to ensure that the constraint matrix has full row rank. We will discard the supply constraint associated with the depot m, and use the following constraint matrix:

$$\boldsymbol{A} = \begin{bmatrix} e^T & 0 & \cdots & 0 & 0 \\ 0 & e^T & \cdots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & e^T & 0 \\ \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} & \boldsymbol{I} \end{bmatrix}$$

where $e^T = (1, 1, \cdots, 1)$, a n vector, and $\boldsymbol{I}$ is the $n \times n$ identity matrix.

Thus we define,

$$\boldsymbol{G} = \begin{bmatrix} e^T & 0 & \cdots & 0 & 0 \\ 0 & e^T & \cdots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \cdots & e^T & 0 \end{bmatrix}, \boldsymbol{H} = (\boldsymbol{I}, \boldsymbol{I}, \cdots, \boldsymbol{I}),$$

where $\boldsymbol{G}$ is a $(m-1) \times mn$ and $\boldsymbol{H}$ is a $n \times mn$ matrix. The diagonal matrix $\boldsymbol{D}$ has mn entries along the diagonal, and has the partition

$$\boldsymbol{D} = \begin{bmatrix} \boldsymbol{D}^{(1)} & 0 & \cdots & 0 \\ 0 & \boldsymbol{D}^{(2)} & \cdots & 0 \\ & & \ddots & \\ 0 & 0 & \cdots & \boldsymbol{D}^{(m)} \end{bmatrix}$$

where $\boldsymbol{D}^{(i)}$ is a $n \times n$ diagonal matrix, and its jth diagonal entry $D^{(i)}_{jj}$ is the $ijth$ diagonal entry of $\boldsymbol{D}$, which corresponds to the entry of $\boldsymbol{D}$ associated with the variable $x_{i,j}$. The exact form of this entry depends on the particular interior point algorithm implemented.

It can now be readily confirmed that

$$\boldsymbol{GDG}^T = \begin{bmatrix} \boldsymbol{e}^T\boldsymbol{D}^{(1)}\boldsymbol{e} & & & \\ & \boldsymbol{e}^T\boldsymbol{D}^{(2)}\boldsymbol{e} & & \\ & & \ddots & \\ & & & \boldsymbol{e}^T\boldsymbol{D}^{(m-1)}\boldsymbol{e} \end{bmatrix}$$

a $(m-1) \times (m-1)$ diagonal matrix; and

$$\boldsymbol{HDH}^T = \sum_{i=1}^{n} \boldsymbol{D}^{(i)}$$

a $n \times n$ diagonal matrix. Thus $\boldsymbol{L}_1$ and $\boldsymbol{L}_3$ are diagonal matrices, with the jth diagonal entry of $\boldsymbol{L}_1$ and $\boldsymbol{L}_3$ being $(\sum_{k=1}^{n} D_{kk}^{(j)})^{\frac{1}{2}}$ and $(\sum_{i=1}^{m} D_{jj}^{(i)})^{\frac{1}{2}}$, respectively. Also

$$\boldsymbol{GDH}^T = \begin{bmatrix} \boldsymbol{e}^T\boldsymbol{D}^{(1)} \\ \boldsymbol{e}^T\boldsymbol{D}^{(2)} \\ \cdot \\ \boldsymbol{e}^T\boldsymbol{D}^{(m-1)} \end{bmatrix},$$

and

$$\boldsymbol{L}_2 = (\boldsymbol{D}^{(1)}\boldsymbol{e}, \boldsymbol{D}^{(2)}\boldsymbol{e}, \cdots, \boldsymbol{D}^{(m-1)}\boldsymbol{e})\boldsymbol{L}_1^{-1}.$$

Thus

$$\begin{aligned} \boldsymbol{E} &= \boldsymbol{L}_3^{-1}\boldsymbol{L}_2 \\ &= \boldsymbol{L}_3^{-1}(\boldsymbol{D}^{(1)}\boldsymbol{e}, \boldsymbol{D}^{(2)}\boldsymbol{e}, \cdots, \boldsymbol{D}^{(m-1)}\boldsymbol{e})\boldsymbol{L}_1^{-1}, \end{aligned}$$

which is a $n \times (m-1)$ positive matrix.

Theorem 8 *Each iteration of the interior point method requires* $nm^2 + 6nm + 2(m-n+1)$ *multiplications for the partitioned assignment problem.*

Proof : Using the structure of $\boldsymbol{L}_1$, $\boldsymbol{L}_2$ and $\boldsymbol{L}_3$; and solving (8) by the method of section 3, and carefully counting the multiplications at each step, we get the theorem.

5 Generalized Upper Bounding Problem

The generalized upper bounding linear program is the following:

$$
\begin{array}{lccccccccccc}
\text{minimize} & \boldsymbol{c}_0\boldsymbol{x}_0 & + & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & + & \cdots & + & \boldsymbol{c}_p^T\boldsymbol{x}_p & \\
& \boldsymbol{A}_0\boldsymbol{x}_0 & + & \boldsymbol{A}_1\boldsymbol{x}_1 & + & \boldsymbol{A}_2\boldsymbol{x}_2 & + & \cdots & + & \boldsymbol{A}_p\boldsymbol{x}_p & = \boldsymbol{b} \\
& & & \boldsymbol{e}_1^T\boldsymbol{x}_2 & & & & & & & = 1 \\
& & & & & \boldsymbol{e}_2^T\boldsymbol{x}_2 & & & & & = 1 \\
& & & & & & & \ddots & & & \vdots \\
& & & & & & & & & \boldsymbol{e}_p^T\boldsymbol{x}_p & = 1 \\
& \boldsymbol{x}_0 \geq \boldsymbol{0} & & \boldsymbol{x}_1 \geq \boldsymbol{0} & & \boldsymbol{x}_2 \geq \boldsymbol{0} & & \cdots & & \boldsymbol{x}_p \geq \boldsymbol{0} &
\end{array}
$$

where, for each $j = 0, \cdots, p$, $\boldsymbol{A}_j$ is a $m \times n_j$ matrix, $\boldsymbol{c}_j$ and $\boldsymbol{x}_j$ are n_j vectors; and for $j = 1, \cdots p$, $\boldsymbol{e}_j$ a is a n_j vector of all ones. Transportation and assignment problems have this structure as well, but in important applications, $m << p$ (i.e., m is much less than p). We make the usual assumption that the constraint matrix has the full row rank $m + p$.

As is evident, the constraints of this problem have the *natural* partition (1) with

$$
\boldsymbol{G} = (\boldsymbol{A}_0, \boldsymbol{A}_1, \cdots, \boldsymbol{A}_p)
$$

$$
\boldsymbol{H} = \begin{bmatrix}
\boldsymbol{0} & \boldsymbol{e}_1^T & \boldsymbol{0} & \cdots & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{e}_2^T & \cdots & \boldsymbol{0} \\
 & & & \ddots & \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{e}_p^T
\end{bmatrix},
$$

and the diagonal matrix $\boldsymbol{D}$ has the partition (corresponding to the partiton of $\boldsymbol{A}$)

$$
\boldsymbol{D} = \begin{bmatrix}
\boldsymbol{D}^{(0)} & & & \\
 & \boldsymbol{D}^{(1)} & & \\
 & & \ddots & \\
 & & & \boldsymbol{D}^{(p)}
\end{bmatrix} \tag{10}
$$

where $\boldsymbol{D}^{(j)}$ is a $n_j \times n_j$ diagonal matrix corresponding to columns of $\boldsymbol{x}_j$ in $\boldsymbol{A}$, for each $j = 1, \cdots, p$.

Here

$$\boldsymbol{GDG}^T = \sum_{j=0}^{p} \boldsymbol{A}_j \boldsymbol{D}^{(j)} \boldsymbol{A}_j^T$$

$$\boldsymbol{HDH}^T = \begin{bmatrix} \boldsymbol{e}_1^T \boldsymbol{D}^{(1)} \boldsymbol{e}_1 & & & \\ & \boldsymbol{e}_2^T \boldsymbol{D}^{(2)} \boldsymbol{e}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{e}_p^T \boldsymbol{D}^{(p)} \boldsymbol{e}_p \end{bmatrix}$$

with $\boldsymbol{HDH}^T$ a diagonal matrix. Thus $\boldsymbol{L}_3$ is a diagonal matrix, and $\boldsymbol{L}_1$, the Cholesky factor of the $m \times m$ matrix

$$\sum_{j=0}^{p} \boldsymbol{A}_j \boldsymbol{D}^{(j)} \boldsymbol{A}_j^T = \boldsymbol{L}_1 \boldsymbol{L}_1^T.$$

In applications, even when $\boldsymbol{A}_j$ are sparse, we would expect their sum to be considerably more dense, and thus we expect $\boldsymbol{L}_1$ to be relatively dense. Also

$$\boldsymbol{GDH}^T = (\boldsymbol{A}_1 \boldsymbol{D}^{(1)} \boldsymbol{e}_1, \boldsymbol{A}_2 \boldsymbol{D}^{(2)} \boldsymbol{e}_2, \cdots, \boldsymbol{A}_p \boldsymbol{D}^{(p)} \boldsymbol{e}_p).$$

Thus if $\boldsymbol{L}_2 = (\boldsymbol{l}_1, \boldsymbol{l}_2, \cdots, \boldsymbol{l}_p)$,

$$\boldsymbol{L}_1 \boldsymbol{l}_j = \boldsymbol{A}_j \boldsymbol{D}^{(j)} \boldsymbol{e}_j$$

and we can expect the $p \times m$ matrix $\boldsymbol{L}_2$ to be dense. In addition, $\boldsymbol{E} = \boldsymbol{L}_3^{-1} \boldsymbol{L}_2$.

Theorem 9 *Each step of an interior point method requires* $\frac{3}{2}pm^2 + \frac{1}{6}m^3 + \frac{11}{2}pm + \frac{3}{2}m^2 - \frac{1}{2}m + 2p$ *when applied to the partitioned GUB problem.*

Proof : Can be obtained by a careful calculation of the work at each step of the algorithm. The multiplications required to obtain the Cholesky factor $\boldsymbol{L}_1$ are included in the above formula.

6 Block diagonal linear program

The block diagonal linear program is the following:

$$
\begin{array}{lccccccccccl}
\text{minimize} & \boldsymbol{c}_0^T\boldsymbol{x}_0 & + & \boldsymbol{c}_1^T\boldsymbol{x}_1 & + & \boldsymbol{c}_2^T\boldsymbol{x}_2 & + & \cdots & + & \boldsymbol{c}_p^T\boldsymbol{x}_p & \\
& \boldsymbol{A}_0\boldsymbol{x}_0 & + & \boldsymbol{A}_1\boldsymbol{x}_1 & + & \boldsymbol{A}_2\boldsymbol{x}_2 & + & \cdots & + & \boldsymbol{A}_p\boldsymbol{x}_p & = \boldsymbol{b}_0 \\
& & & \bar{\boldsymbol{A}}_1\boldsymbol{x}_2 & & & & & & & = \boldsymbol{b}_1 \\
& & & & & \bar{\boldsymbol{A}}_2\boldsymbol{x}_2 & & & & & = \boldsymbol{b}_2 \\
& & & & & & & \ddots & & & \vdots \\
& & & & & & & & & \bar{\boldsymbol{A}}_p\boldsymbol{x}_p & = \boldsymbol{b}_p \\
& \boldsymbol{x}_0 \geq \boldsymbol{0} & & \boldsymbol{x}_1 \geq \boldsymbol{0} & & \boldsymbol{x}_2 \geq \boldsymbol{0} & & \cdots & & \boldsymbol{x}_p \geq \boldsymbol{0} &
\end{array}
$$

where, for each $j = 0, \cdots, p$, $\boldsymbol{A}_j$ is a $m_0 \times n_j$ matrix, $\boldsymbol{c}_j$ and $\boldsymbol{x}_j$ are n_j vectors; and, for each $j = 1, \cdots, p$, $\bar{\boldsymbol{A}}_j$ is a $m_j \times n_j$ matrix. Let $m = m_0$ and $M = m_1 + \cdots + m_p$. The constraints of this problem have the *natural* partition (1) with

$$
\boldsymbol{G} = (\boldsymbol{A}_0, \boldsymbol{A}_1, \cdots, \boldsymbol{A}_p)
$$

$$
\boldsymbol{H} = \begin{bmatrix}
\boldsymbol{0} & \bar{\boldsymbol{A}}_1 & \boldsymbol{0} & \cdots & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \bar{\boldsymbol{A}}_2 & \cdots & \boldsymbol{0} \\
 & & & \ddots & \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \cdots & \bar{\boldsymbol{A}}_p
\end{bmatrix},
$$

and the diagonal matrix $\boldsymbol{D}$ has the partition (10). It can be verified that

$$
\boldsymbol{G}\boldsymbol{D}\boldsymbol{G}^T = \sum_{j=0}^{p} \boldsymbol{A}_j \boldsymbol{D}^{(j)} \boldsymbol{A}_j^T;
$$

and,

$$
\boldsymbol{H}\boldsymbol{D}\boldsymbol{H}^T = \begin{bmatrix}
\bar{\boldsymbol{A}}_1 \boldsymbol{D}^{(1)} \bar{\boldsymbol{A}}_1^T & & & \\
 & \bar{\boldsymbol{A}}_2 \boldsymbol{D}^{(2)} \bar{\boldsymbol{A}}_2^T & & \\
 & & \ddots & \\
 & & & \bar{\boldsymbol{A}}_p \boldsymbol{D}^{(p)} \bar{\boldsymbol{A}}_p^T
\end{bmatrix}
$$

is a block diagonal matrix. As in the GUB case, the factor $\boldsymbol{L}_1$ of $\boldsymbol{G}\boldsymbol{D}\boldsymbol{G}^T$ will be relatively dense. $\boldsymbol{L}_3$, the Cholesky factor of $\boldsymbol{H}\boldsymbol{D}\boldsymbol{H}^T$ is block diagonal, and each diagonal factor can

be computed independently. Thus, we can assume that

$$\boldsymbol{L}_3 = \begin{bmatrix} \boldsymbol{L}_{31} & & & \\ & \boldsymbol{L}_{32} & & \\ & & \ddots & \\ & & & \boldsymbol{L}_{3p} \end{bmatrix}.$$

When $\bar{\boldsymbol{A}}_j$ is sparse, we can preserve the sparsity of $\boldsymbol{L}_{3j}$ (the Cholesky factor of $\bar{A}_j \boldsymbol{D}^{(j)} \bar{\boldsymbol{A}}_j^T$) by the techniques of sparse Cholesky factorization, George and Liu [3]. Also

$$\boldsymbol{GDH}^T = (\boldsymbol{A}_1 \boldsymbol{D}^{(1)} \bar{\boldsymbol{A}}_1^T, \cdots, \boldsymbol{A}_p \boldsymbol{D}^{(p)} \bar{\boldsymbol{A}}_p^T)$$

is a $m \times M$ matrix. If $\boldsymbol{L}_2^T = (\boldsymbol{L}_{21}^T, \cdots, \boldsymbol{L}_{2p}^T)$,

$$\boldsymbol{L}_1 \boldsymbol{L}_{2j}^T = \boldsymbol{A}_j \boldsymbol{D}^{(j)} \bar{\boldsymbol{A}}_j^T.$$

We expect $\boldsymbol{L}_{2j}$ to be, generally, dense.

Let $\boldsymbol{E} = (\boldsymbol{E}^{(1)}, \boldsymbol{E}^{(2)}, \cdots, \boldsymbol{E}^{(p)})^T$ which conforms to the partition of $\boldsymbol{L}_3$, Then, for each $j = 1, 2, \cdots, p$

$$\boldsymbol{L}_{3j} \boldsymbol{E}^{(j)} = \boldsymbol{L}_{2j}$$

and we can readily establish the following theorem:

Theorem 10 *For a dense problem,it takes* $\frac{1}{6}\sum_{j=0}^{p}(m_j^3 + 3m_j^2 - 9m_j) + \frac{1}{2}Mm(m+1) + \frac{m}{2}\sum_{j=1}^{p} m_j(m_j+1)$ *multiplications to generate* $\boldsymbol{L}_1$, $\boldsymbol{L}_2$, $\boldsymbol{L}_3$ *and* $\boldsymbol{E}$*; and, takes* $Mm^2+4Mm+m^2+m+\sum_{j=1}^{p} m_j^2 + M$ *multiplications to solve the systems of theorem (4). If* $m_j = \bar{m}$ *for all* $j = 1, \cdots p$*, the above forms reduce to* $\frac{1}{6}(m^3 + 3m^2 - 9m + p\bar{m}^3 + 3p\bar{m}^2 - 9p\bar{m}) + \frac{1}{2}p\bar{m}m(m+1) + \frac{1}{2}pm\bar{m}(\bar{m}+1)$ *and* $pm^2\bar{m} + 4pm\bar{m} + m^2 + m + p\bar{m}^2 + p\bar{m}$ *respectively. Thus, in this case, the computations grow linearly in p.*

Proof : This can be readily proved by a careful counting of the required multiplications.

One specially structured, and important problem which shares the block-diagonal structure is the Multi-commodity flow problem. In its node-arc formulation, p is the number of commodities, $\boldsymbol{A}_i = \boldsymbol{I}$ for each $i = 0, \cdots, p$; and $\bar{\boldsymbol{A}}_i = \boldsymbol{R}$, where $\boldsymbol{R}$ is a node-arc incidence

matrix of the directed network through which these commodities flow. Thus,

$$GDG^T = \sum_{j=0}^{p} D^{(j)}$$

is diagonal, and the sparsity pattern of $\bar{A}_i D^{(i)} \bar{A}_i^T$, for each $i = 1, \cdots, p$, is the same. Also,

$$A_i D^{(i)} \bar{A}_i^T = D^{(i)} R^T.$$

Thus

$$L_{2i}^T = (\sum_{j=1}^{p} D^{(j)})^{-\frac{1}{2}} D^{(i)} R^T$$

$$L_{3i} E^{(i)} = R D^{(i)} (\sum_{j=1}^{p} D^{(j)})^{-\frac{1}{2}}. \tag{11}$$

L_{3i} is the Cholesky factor of $RD^{(i)}R^T$, and, for each i, has the same sparsity pattern as that of the factor of RR^T. It is readily confirmed that , in the notation of George and Liu [3], the graph associated with RR^T is the unordered network on which the commodities flow. This facilitates considerably the use of their techniques for generating sparse Cholesky factors L_{3i}. Since each column of R has only two nonzero entries, $E^{(i)}$ will be sparse; and will have the same sparsity pattern for each i.

For a network with m nodes, n arcs and p commodities, L_1 is a $n \times n$ diagonal matrix, for each $i = 1, \cdots, p$, L_{3i} is $m \times m$ matrix, L_{2i} is $m \times n$ and E is $pm \times n$. Also, assume that it takes δ multiplications to get a sparse Cholesky factor of RR^T; and note that $\delta \leq \frac{1}{6}(m^3 + 3m^2 - 9m)$; and that the number of non-zero elements in this factor are η. Then, the following can be shown:

Theorem 11 *Given L_1, L_2, L_3 and E, it takes $pmn^2 + 4pn + 2n + p\eta$ multiplications to solve all the systems of theorem (4).*

7 Computational Experience

In this section, we present some computational experience of solving assignment problems by the procedure suggested here and by the state-of -the-art code LOQO, Vanderbei [9].

LOQO is a state of the art interior point code based on the primal-dual homotopy method, and implements a predictor-corrector strategy for tracing the path of centers. The per iteration times of this code are compared with the per iteration times of a specialized transportation code, implementing the dual affine scaling strategy. The LOQO per iteration times may be a little larger because of the step size selection in the predictor-corrector strategy. The per-iteration times and their ratio are given in Table below. We point out that for these special problems, the specialized version was about 3 to 4 times faster than the general purpose code, LOQO.

		Assignment			LOQO			Ratio
Problem	Size	iter	time *	iter time	iter	time	iter time	Loqo/Asgn
1	200×200	14	20.78	1.48	15	75.72	5.048	3.41
2	200×200	17	25.09	1.476	17	85.30	4.911	3.33
3	200×200	21	31.02	1.477	20	96.61	4.830	3.27
4	300×300	14	65.72	4.69	18	301.07	16.72	3.565
5	300×300	19	88.80	4.67	19	314.16	16.53	3.540
6	300×300	21	98.06	4.67	21	418.5	16.86	3.61

* This time is the total time for all the iterations, without the input/output time. All times are in seconds.

The three problems of size 200×200; and, of size 300×300 are generated randomly, and present increasing difficulty to interior point methods. On the first problem, these methods converge to a solution in the interior of a face, with very few variables at value 1. On the second problem, these methods converge to a solution with more that 75 % of the variables at value 1, while on the third problem they converge to a vertex, with all variables at value 1. For the first problem, LOQO found a solution to the accuracy of eight significant figures, but for the second and third, it was unable to find this accurate a solution. For these problems it found a solution to 7 digits of accuracy.

References

[1] E. R. Barnes, " A variation of Karmarkar's Algorithm for solving Linear Programming Problems, " *Mathematical Programming* 36(1986) 174 - 182.

[2] S. C. Fang and S. Puthenpura, *Linear Optimization and Extensions* , Prentice Hall, in print.

[3] A. George and J. W. Liu, *Computer solution of large positive definite systems* , Prentice Hall, Englewood Cliffs, New Jersey (1981).

[4] N. Karmarkar, " A new polynomial time algorithm for linear programming", *Combinatorica* 4(1984) 373 - 395.

[5] M. Kojima, S. Mizuno and A. Yoshise, " A primal-dual interior point method for linear programming ", in *Progress in Mathematical Programming: Interior Point Methods* , edited by N. Megiddo, Springer Verlag, New York (1989) 29-48.

[6] L. S. Lasdon, *Optimization Theory for Large Systems* The Macmillian Company, London (1970).

[7] R. Saigal, "An Infinitely summable series implementation of interior point methods", Technical Report 92 - 37, Department of Industrial and Operations Engineering, University of Michigan, May 1992.

[8] R. J. Vanderbei, M. S. Meketon and B. A. Freedman, " A modification of Karmarkar linear programming algorithm ", *Algorithmatica* 1(1986) 395 - 407.

[9] R. J. Vanderbei, *LOQO User's Manual* , Program in Statistics and Operations Research, Princenton University, Princeton, N. J. 08544, June 1992.