

Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods

S. Bocanegra · F.F. Campos · A.R.L. Oliveira

Published online: 23 February 2007
© Springer Science+Business Media, LLC 2007

Abstract We devise a hybrid approach for solving linear systems arising from interior point methods applied to linear programming problems. These systems are solved by preconditioned conjugate gradient method that works in two phases. During phase I it uses a kind of incomplete Cholesky preconditioner such that fill-in can be controlled in terms of available memory. As the optimal solution of the problem is approached, the linear systems becomes highly ill-conditioned and the method changes to phase II. In this phase a preconditioner based on the LU factorization is found to work better near a solution of the LP problem. The numerical experiments reveal that the iterative hybrid approach works better than Cholesky factorization on some classes of large-scale problems.

Keywords Interior point methods · Preconditioning · Ill-conditioned systems

1 Introduction

The most expensive part of an interior point method is computing the search direction by solving one or more linear systems. Such systems are indefinite and can be written in a symmetric form, which is known as augmented system. Normally,

S. Bocanegra · F.F. Campos (✉)
Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Av. Antônio Carlos, 6627, 31.270-010 Belo Horizonte, MG, Brazil
e-mail: ffcampos@dcc.ufmg.br

S. Bocanegra
e-mail: silvana@dcc.ufmg.br

A.R.L. Oliveira
Departamento de Matemática Aplicada, Universidade Estadual de Campinas, C.P. 6065,
13.083-970 Campinas, SP, Brazil
e-mail: aurelio@ime.unicamp.br

they are solved by direct methods. A common approach in interior point solvers for linear programming reduces the augmented system to a smaller positive definite one, called normal equations system, and uses sparse Cholesky factorization to solve them [2, 11, 18, 27]. Sometimes, the use of direct methods becomes prohibitive due to storage and time limitations. In such situations iterative approaches are more interesting.

The success of implementations using iterative methods depends on how to choose an appropriate preconditioner since the matrix system becomes ill-conditioned as the optimal solution of the problem is approached. Several preconditioners have been used to solve the normal equations systems from interior point methods, see for example [1, 29, 36, 39]. Some of them are based on incomplete Cholesky factorization of the positive definite matrix. Typically, this class of preconditioners is efficient in initial iterations, but it deteriorates as the interior point method converges to a solution. In general, the normal equations system is more ill-conditioned and dense than augmented system and this has motivated the study of methods to solve the indefinite system [7, 14, 17, 21, 26, 34]. An excellent survey about the solution of large-scale systems in the augmented systems form can be found in [6]. The preconditioner for the augmented system proposed in [34] is based on LU factorization and it has an opposite behavior when compared to incomplete Cholesky based ones. It performs better near a solution of the linear programming problem when the matrices are highly ill-conditioned.

We are proposing an iterative hybrid approach to solve the normal equations system that arises in an interior point method for linear programming. The conjugate gradient method is preconditioned during the initial interior points iterations (*phase I*) using a kind of incomplete factorization called controlled Cholesky factorization (CCF) proposed by Campos [9] and in the remaining iterations (*phase II*) using the splitting preconditioner developed by Oliveira [33]. The amount of memory used by CCF preconditioner is easily controlled during interior point iterations. As the system becomes ill-conditioned more fill-in is allowed. When the CCF preconditioner loses efficiency, the system is already highly ill-conditioned and it is a good indicator that the splitting preconditioner will work better. An efficient heuristic is being developed to identify the change of phases. The hybrid approach is applied within the PCx code [11], an interior point method implementation.

This article is organized as follows. In Sect. 2 we recall the basic ideas of primal-dual interior point methods for linear programming. Section 3 discusses some of approaches that have been used to solve the linear systems. A hybrid preconditioner is presented in Sect. 4. Numerical experiments are shown in Sect. 5. In Sect. 6 the conclusions are drawn and further developments are suggested.

2 Primal–dual interior point methods

Interior point algorithms are widely used for solving large-scale linear programming problems. Consider the linear programming problem in the primal form,

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \tag{1}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ and $m \leq n$. The dual problem associated with Problem (1) is

$$\begin{aligned} & \max \quad b^T y \\ & \text{subject to} \quad A^T y + z = c, \\ & \quad \quad \quad z \geq 0 \end{aligned} \tag{2}$$

where $y \in \mathbb{R}^m$ is a vector of free variables and $z \in \mathbb{R}^n$ is the vector of dual slack variables. The Karush–Kuhn–Tucker optimality conditions for (1) and (2) are

$$\begin{aligned} Ax - b &= 0, \\ A^T y + z - c &= 0, \\ XZe &= 0, \\ (x, z) &\geq 0 \end{aligned}$$

where $X = \text{diag}(x)$, $Z = \text{diag}(z)$ and $e \in \mathbb{R}^n$ is the vector of all ones. One of the most successful among the interior point methods is Mehrotra’s predictor-corrector method [27, 29]. This method is based on Newton’s method applied to the modified KKT conditions to retain the nonnegativity of the (x, z) components and incorporating a centering parameter. The search direction is obtained by solving two linear systems, which have the same coefficient matrix but different right-hand sides. Firstly, the affine-scaling directions are computed by solving

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} \Delta_a x^k \\ \Delta_a y^k \\ \Delta_a z^k \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ r_a \end{bmatrix} \tag{3}$$

where $r_p = b - Ax^k$, $r_d = c - A^T y^k - z^k$ and $r_a = -X^k Z^k e$. Then, to compute the centering corrector direction $(\Delta_c x^k, \Delta_c y^k, \Delta_c z^k)$ the right-hand side vector of (3) is set to $r_d = 0$, $r_p = 0$ and $r_a = \mu_k e - \Delta_a X^k \Delta_a Z^k e$. Here μ_k is the centering parameter, $\Delta_a X^k = \text{diag}(\Delta_a x^k)$ and $\Delta_a Z^k = \text{diag}(\Delta_a z^k)$. The search direction is obtained by adding the affine-scaling direction to the centering corrector direction. In order to avoid this addition, the search directions are computed directly solving system (3) with r_a set to $\mu_k e - \Delta_a X^k \Delta_a Z^k e - X^k Z^k e$.

3 Linear system solution

The bulk of the work in interior point methods is the determination of search directions. Since both predictor-corrector systems share the same coefficient matrix, we will restrict the discussion to that of one linear system like (3). In practice, the variables Δz are eliminated and the system reduces to the augmented indefinite linear system, which can be written in a symmetric form

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1} r_a \\ r_p \end{bmatrix} \tag{4}$$

where $\Theta = Z^{-1}X$. System (4) can be reduced to a smaller positive definite one called normal equations system by eliminating the variables Δx from the first equation,

$$(A\Theta A^T)\Delta y = A\Theta(r_d - X^{-1}r_a) + r_p. \quad (5)$$

Direct or iterative methods can be applied to solve either system (4) or (5). The most common approach used in interior point methods for linear programming solves the normal equations system by a direct method like Cholesky factorization. This approach has the advantage of working with symmetric positive definite matrices. However, the presence of few dense columns in A causes loss of sparsity in $A\Theta A^T$. One way around this problem is to use iterative methods. Since they require the matrix only for computing matrix-vector products there is no need to compute $A\Theta A^T$ explicitly, unless the preconditioner depends on it. The classical iterative method used to solve the normal equations system is the preconditioned conjugate gradient [5, 29]. A preconditioned conjugate gradient was initially used in interior points methods in 1989 [1, 2]. However, to build the preconditioner such implementations often need to compute $A\Theta A^T$, which is usually less sparse than A . Moreover, it is in general more difficult to find a good preconditioner to the normal equations matrix than to the augmented one, since the former is likely to be more ill-conditioned than the latter.

For this reason, the augmented system strategy has been considered in several papers [7, 14, 17, 21, 26, 34] even though it is indefinite. The Cholesky factorization cannot be applied since there is no numerically stable way to factor a general indefinite matrix onto LDL^T with D diagonal. Most interior point solvers use direct methods like a Bunch–Parlett factorization [8] to solve the indefinite system. Another alternative that has been used in interior point methods transforms the indefinite system to quasidefinite one by the use of primal and dual regularization method [3]. In this case, the Cholesky-like factorization LDL^T with a diagonal D exists for any symmetric row and column permutation of the matrix. This approach also has been used with iterative methods [7]. For indefinite systems, the conjugate gradient method is not guaranteed to converge. However, it was successfully applied with a convenient preconditioner [7, 14]. The indefinite systems also have been used with preconditioned conjugate gradient to solve quadratic programming problems [12, 20].

Most preconditioners for indefinite systems from interior point methods are developed for solving non-linear and quadratic programming problems and almost always they are not efficient in solving large-scale linear programming problems. However, a class of preconditioners called splitting preconditioners was designed specially for indefinite systems arising from linear programming problems [34]. An important feature of this class is the option to reduce the preconditioned indefinite system to a positive definite one like the normal equations system allowing the use of conjugate gradient method. Since this class was developed for the last interior point iterations a diagonal scaling preconditioner has been used in the initial iterations. We have improved the robustness and performance of this approach implementing an efficient preconditioner instead of the diagonal scaling one.

4 Hybrid preconditioner

In general, matrix Θ changes significantly between interior point iterations and it becomes highly ill-conditioned in the final iterations. For this reason it is difficult to find a preconditioning strategy that produces good performance of iterative methods over the entire course of the interior point iterations.

We are proposing to apply the conjugated gradient method for solving system (5) preconditioned by a hybrid preconditioner matrix M ,

$$M^{-1}(A\Theta A^T)M^{-T}\bar{y} = M^{-1}(A\Theta(r_d - X^{-1}r_a) + r_p) \tag{6}$$

where $\bar{y} = M^T \Delta y$. Our approach assumes the existence of two phases during interior point iterations. In the first one, the controlled Cholesky preconditioner is used to build matrix M , this method is described in Sect. 4.1. After the change of phases, matrix M will be built using the splitting preconditioner, which is presented in Sect. 4.2.

4.1 Controlled Cholesky factorization preconditioner

The Controlled Cholesky Factorization (CCF) preconditioner was designed for solving general positive definite systems [9] and it was successfully applied to solve linear systems from implicit time-dependent partial differential equations [10]. This kind of factorization has some desirable properties from an optimization viewpoint, such as possibility to control fill-in with predictable memory requirements. However, it has not been used in this context yet.

Consider the Cholesky factorization $A\Theta A^T = LL^T = \tilde{L}\tilde{L}^T + R$, where L is the factor obtained when factorization is complete, \tilde{L} when it is incomplete and R is a remainder matrix. Defining $E = L - \tilde{L}$ then the preconditioned coefficient matrix is

$$\tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} = (\tilde{L}^{-1}L)(\tilde{L}^{-1}L)^T = (I + \tilde{L}^{-1}E)(I + \tilde{L}^{-1}E)^T.$$

It is easy to see that when $\tilde{L} \approx L \Rightarrow E \approx 0 \Rightarrow \tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} \approx I$. It is assumed that the matrix $A\Theta A^T$ has been diagonally scaled to give a unit diagonal [15] in order to improve robustness. The CCF is based on the minimization of the Frobenius norm of E . Thus we consider the problem

$$\text{minimize } \|E\|_F^2 = \sum_{j=1}^m c_j \quad \text{with } c_j = \sum_{i=1}^m |l_{ij} - \tilde{l}_{ij}|^2.$$

Now c_j can be split in two summations:

$$c_j = \sum_{k=1}^{t_j+\eta} |l_{i_k j} - \tilde{l}_{i_k j}|^2 + \sum_{k=t_j+\eta+1}^m |l_{i_k j}|^2$$

where t_j is the number of nonzero entries below the diagonal in the j th column of matrix $A\Theta A^T$ and η is the number of extra entries allowed per column. The first summation contains all $t_j + \eta$ nonzero entries of the j th column of \tilde{L} . The second one has

only those remaining entries of the complete factor L which do not have corresponding entries in \tilde{L} . Considering that $\tilde{l}_{ij} \approx l_{ij}$ as $\eta \rightarrow m$ and l_{ij} is not computed, $\|E\|_F$ is minimized based on a heuristic, which consists of modifying the first summation. By increasing η , that is, allowing more fill-in, c_j will decrease simply because the first summation contains more terms and the second fewer. This is the same nonselective minimization that occurs when levels of fill-in are used. Moreover, $\|E\|_F$ is further minimized by choosing the $t_j + \eta$ largest entries of \tilde{L} in absolute value, thus annihilating the corresponding largest entries in L leaving only the smallest l_{ij} in the second summation. This is a selective minimization similar to a drop tolerance scheme [31]. The preconditioner \tilde{L} is built by columns. Consequently it needs only the j th column of $A\Theta A^T$ at each time. The main features of the CCF preconditioner are described as follows:

4.1.1 Choice of entries by value

The Controlled Cholesky preconditioner contains no more than a fixed number η of nonzero entries in each column in addition to the number t_j in that column of the original matrix. In this approach, all nonzero entries of the j th column resulting from those held for previous columns are computed, and then only the largest $t_j + \eta$ nonzero entries in absolute value are selected. Thus, each time that a column of the preconditioner is computed and stored, the larger entries are kept and the smaller ones discarded. Therefore, a better approximation to the full decomposition is obtained in the allowed storage space. CCF never considers the sparsity pattern of the original matrix. The nonzero entries position kept for the preconditioner may or may not cover all those of the original matrix since selection is made by value instead of position. CCF gives good preconditioners with small storage needs in comparison with preconditioners which conserve the sparsity pattern of the original matrix.

4.1.2 Generalization of improved ICF

Jones and Plassmann [24] developed an approach which allows a fixed number of nonzero entries in each row or column of the preconditioner. It is taken to be the number of nonzero entries in each row or column of the original coefficient matrix. Nevertheless, only the largest entries in magnitude are kept, meaning that the original sparsity pattern is ignored. Thus CCF can be seen as a generalization of the Jones and Plassmann method since fill-in is allowed in CCF.

4.1.3 Avoiding loss of positive definiteness by exponential shift

Manteuffel [28] proved that if the coefficient matrix V is symmetric positive definite then there is a constant $\sigma > 0$ such that an incomplete factorization $V + \sigma I$ exists. During computation of the j th column of \tilde{L} the diagonal element can become very small or even negative. In order to avoid loss of positive definiteness CCF discards the entire factor \tilde{L} , it increases diagonal of $A\Theta A^T$ by σ_i and it computes \tilde{L} again. Instead of defining a linear shift $\sigma_i = 10^{-2}i$ at the i th attempt for constructing the preconditioner \tilde{L} , as used by Jones and Plassmann [24], CCF uses an exponential shift $\sigma_i = 5 \cdot 10^{-4}2^{i-1}$ in order to have smaller diagonal perturbations.

Table 1 Fill-in and drop-out with CCF(η)

η	M	Storage
$-m$	$\text{diag}(A\Theta A^T)^{-1/2}$	less than $A\Theta A^T$
0	\tilde{L}	equal to $A\Theta A^T$
m	L	more than $A\Theta A^T$

Table 2 Comparison between storage demands of ICD(0) and CCF(η). m : order of $A\Theta A^T$, t : number of nonzero entries of $A\Theta A^T$ (excluding those above diagonal), η : number of extra entries per column, i, r : number of bytes for integer and real variables

Matrices	Maximum bytes required	For $i = 4$ and $r = 8$
$A\Theta A^T$	$(i + r)t + im + i$	$12t + 4m + 4$
$A\Theta A^T + \tilde{L}_{ICD(0)}$	$(i + 2r)t + im + i$	$20t + 4m + 4$
$A\Theta A^T + \tilde{L}_{CCF(\eta)}$	$(3i + 2r)t + ((2i + r)\eta + 2i)m + 2i$	$28t + (16\eta + 8)m + 8$

4.1.4 Versatile preconditioner

In Table 1 features of a versatile preconditioning matrix M for system (6) are given. This means that just a diagonal scaling is performed when $\eta = -m$. On the other hand a complete Cholesky factorization is obtained if $\eta = m$. So, η can be chosen in the interval $[-m, m]$ in such a way that M will require less or more storage than $A\Theta A^T$. In other words, fill-in ($\eta > 0$) and drop-out ($\eta < 0$) are both allowed with CCF.

4.1.5 Predictable storage

The coefficient matrix $A\Theta A^T$ and the preconditioner \tilde{L} are stored using the Harwell-Boeing format [13]. Each matrix is stored into three vectors: one integer vector of dimension $m + 1$ for column pointers, one integer vector of dimension t (number of nonzero entries excluding those above diagonal) for row indices and one real vector of dimension t for elements. CCF needs other work vectors for building \tilde{L} . In incomplete Cholesky decomposition (ICD) fill-in is determined by adding levels resulting in uncontrolled storage demands. However, the storage required is predictable in CCF, as shown in Table 2, because only the largest $t_j + \eta$ nonzero entries are kept in each column of the preconditioner. The third column shows the amount of memory required when integer vectors using four bytes and real vectors of eight bytes are used in the code.

CCF(η) demands more storage than ICD(0) when $\eta > (1 - 2t - m)/(4m) \approx -t/(2m)$, for $i = 4$ and $r = 8$ bytes. Nevertheless CCF(η) has a more versatile data structure which allows the number of nonzero entries of \tilde{L} to vary.

4.2 Splitting preconditioner

The splitting preconditioner was proposed for indefinite systems arising from interior point methods [33] for linear programming problems. This preconditioner is a generalization of those proposed by Resende and Veiga [37] in the context of the minimum

cost network flow problem. The main appeal of this class of preconditioners is that it works better near a solution of the linear programming problem. That is a very welcome feature since the linear system is known to be very ill-conditioned close to a solution and these systems are difficult to solve by iterative methods. Additionally, the splitting preconditioner avoids the normal equations computation. However, since the preconditioner is specially tailored for the final iterations of the interior point methods, it fails to obtain convergence in the initial iterations for many linear programming problems. We are using a version of the splitting preconditioner derived as follows [34]:

Let $A = [BN]P$ where P is a permutation matrix such that B is nonsingular, then,

$$A\Theta A^T = B\Theta_B B^T + N\Theta_N N^T.$$

Now, multiplying it by $\Theta_B^{-\frac{1}{2}} B^{-1}$ and post-multiplying by its transpose leads to

$$T = \Theta_B^{-\frac{1}{2}} B^{-1} (A\Theta A^T) B^{-T} \Theta_B^{-\frac{1}{2}} = I + WW^T \tag{7}$$

where $W = \Theta_B^{-\frac{1}{2}} B^{-1} N \Theta_N^{\frac{1}{2}}$.

Notice that the preconditioned matrix is positive definite and its eigenvalues are greater or equal to one, that is, it has no eigenvalues in the neighborhood of zero.

Let $\tilde{N} = B^{-1}N$, which can be seen as a scaling of the linear programming problem. Close to a solution, at least $n - m$ entries of Θ are small. Thus, with a suitable choice of the columns of B , the diagonal entries of Θ_B^{-1} and Θ_N are very small close to a solution. In this situation, W approaches the zero matrix, T approaches the identity matrix and both the largest eigenvalue of T and $\kappa_2(T)$ approach one.

The price paid for avoiding the normal equations system is to find B and solve linear systems using it. However, the factorization $QB = LU$ is typically easier to compute than the Cholesky factorization. In fact, it is known [16] that the sparse pattern of L^T and U is contained in the sparse pattern of R , where $AA^T = R^T R$, for any valid permutation Q . In practice, the number of nonzero entries of R is much larger than the number of nonzero entries of L and U together.

4.2.1 Finding B

A strategy to form B is to minimize $\|W\|$ since close to a solution the preconditioned matrix approaches the identity since for a suitable choice of B 's columns the diagonal entries of Θ_B^{-1} and Θ_N are very small. This problem is hard to solve but an approximate solution can be obtained by selecting the first m linearly independent columns of $A\Theta$ with smallest norm-1 to form B .

A nice property of the splitting preconditioner is that it can work with the selected set of columns for some iterations. As a consequence, the preconditioner is very cheap to compute for such iterations. It is important to notice that keeping the matrix B from previous iterations does not mean to keep the same preconditioner since Θ will change from an iteration to the next and the preconditioner depends on it too.

For this application, the most economical way to compute the LU factorization is to work with the delayed update form. It fits very well to our problem because when

a linearly dependent column appears, it is eliminated from the factorization and the method proceeds with the next column in the ordering given by the heuristic.

One of the main drawbacks of a straightforward implementation of the splitting preconditioner is the excessive fill-in in the LU factorization. The reason is that the criterion for reordering the columns does not take the sparsity pattern of A into account. A good technique consists of interrupting the factorization when excessive fill-in occurs and reordering the independent columns found thus far by the number of nonzero entries. The factorization is then started from scratch and the process is repeated until m independent columns are found. The implementation described in [34] considers a factorization to have an excessive fill-in when it produces more nonzero entries than the number of nonzero entries from the normal equations system.

Another factorization is already worth to be applied on the chosen set of independent columns using standard techniques for computing an efficient sparse LU factorization. This approach improves the results significantly for some problems. As a welcome side effect, it is not necessary to store U in the factorizations that determine B .

One difficulty in determining the subset of independent columns is the number of dependent columns visited in the process. The techniques developed [34] for determining the subset of columns and computing the splitting preconditioner are rather sophisticated since the subset of columns from A that form B is not known a priori. However, a careful implementation applied to the conjugate gradient method compares favorably with the Cholesky factorization approach on large scale problems whose Cholesky factorization contains too many nonzero entries.

4.3 Change of phases

The change of phases is a crucial point to improve the performance of this approach. We are studying a criterion for it based upon the matrix condition number, which can be estimated by Ritz values. Unfortunately these values vary extremely with the problem that is being solved, and it becomes difficult to find a criterion that has good performance for all problems.

We are using a heuristic to switch phases that seems to work fine for some test problems. The CCF preconditioner changes to the split preconditioner when the initial gap $(x_0^T z_0)$ for the linear programming is reduced by a factor of 10^6 or the number of inner iterations for solving the linear system reaches $m/2$, where m is the dimension of $A\Theta A^T$.

In the initial iteration, the number of nonzero entries allowed in CCF preconditioner is set by the initial parameter η

$$\eta_0 = \begin{cases} -|A\Theta A^T|/m & \text{if } |A\Theta A^T|/m > 10, \\ |A\Theta A^T|/m & \text{otherwise} \end{cases}$$

where $|\cdot|$ denotes the number of nonzero entries matrix. If the matrix $A\Theta A^T$ is dense, the η_0 parameter forces the preconditioner matrix to be the diagonal scaling. On the other hand, if the $A\Theta A^T$ matrix is sparse then a more dense preconditioner matrix is allowed.

As the conjugate gradient loses efficiency the parameter η is increased. If the number of iterations to achieve convergence is greater than $m/4$, η is increased by 10. This

process continues until η reaches η_{\max} or the change of phases is identified. The η_{\max} value is set based on the amount of available memory.

5 Numerical experiments

The hybrid approach was added to PCx code. Procedures for solving linear systems were coded in C, except the controlled Cholesky factorization, which was implemented in FORTRAN. Multiple centrality corrections [19] are not allowed in iterative approaches and this option was disabled. The remaining PCx's default parameters are adopted.

In all experiments we have used the preconditioned conjugate gradient method with termination criteria set by the Euclidean residual norm $\|r_k\|$. For solving both systems (*affine direction*) and (*final direction*) in phase I, the termination criteria is set as $\|r_k\| < 10^{-4}$. When the optimality gap is less than 10^{-5} or change of phases is detected then $\|r_k\| < 10^{-8}$. The limit of iterations is the system dimension.

5.1 Test problems

The problems used to test the hybrid approach are shown in Table 3. These experiments were selected as a way to identify classes of problems where the hybrid preconditioner has good performance. Some of them are public domain linear problems available in NETLIB [32], STOCHLP [23], MISC [22, 30]. The KBAPAH2 problem [4] belongs to a highly ill-conditioned problems collection. The QAP models tested here are from the QAPLIB collection [25] with the modification as described by Padberg and Rijal [35]. Table 3 summarizes the test data. The number of rows, columns and nonzero entries refer to preprocessed problems.

5.2 Computational results

The behavior of the hybrid approach is compared to the direct method and the incomplete Cholesky factorization (ICF) [38]. The results are presented in Table 4. For each test problem IT is the number of interior point iterations and the time indicated is the total CPU time in seconds, on an Intel Xeon 2.80 GHz processor with 1 Gbyte of RAM.

For problems where an optimal solution is reached, the number of outer iterations for interior point methods using both approaches is about the same. In problems NUG12, NUG15, QAP12 and QAP15 it is interesting to notice that an optimal solution is not obtained using the direct approach. This fact occurs due to the approach in Cholesky factorization code, which replaces tiny diagonals with a large numerical value. The CCF preconditioner uses a small diagonal perturbation (exponential shift) to avoid this breakdown.

The SCSD8-2B-64, SCSD8-2C-64 and SCSD8-2R-432 instances have dense columns in the matrix A . To avoid the density in the normal equations matrix, the direct approach treats these columns separately. Nevertheless, the direct approach fails due to a large residual error. The hybrid preconditioner reaches the optimal solution.

Table 3 Test problems data

Problem	Row	Column	Nonzeros A	Collection
KBAPAH2	12	28	299	KBAPAH
ELS-19	4350	13186	50882	QAP
CHR22B	5587	10417	36520	QAP
CHR25A	8149	15325	53725	QAP
SCR15	2234	6210	24060	QAP
SCR20	5079	15980	61780	QAP
ROU20	7359	37640	152980	QAP
STE36A	27683	131076	512640	QAP
STE36B	27683	131076	512640	QAP
STE36C	27683	131076	512640	QAP
QAP12	2794	8856	33528	NETLIB
QAP15	5698	22275	85470	NETLIB
SCSD8-2B-64	5130	35910	112770	STOCHLP
SCSD8-2C-64	5130	35910	112770	STOCHLP
SCSD8-2R-432	8650	60550	190210	STOCHLP
NUG08	742	1632	5936	MISC
NUG12	2794	8856	33528	MISC
NUG15	5698	22275	85470	MISC
PDS-20	32276	106180	226494	MISC
PDS-40	64265	214385	457538	MISC
PDS-60	96503	332862	709178	MISC
PDS-80	126109	430800	916852	MISC
PDS-100	156243	514577	1096002	MISC

Table 5 provides an insight into the performance of the hybrid approach. $|M|$ column indicates nonzero entries in the hybrid preconditioner matrix at last iteration. The $A\Theta A^T$ column shows only the nonzero entries of the matrix triangular part. Matrix L column contains the number of nonzero entries in the complete Cholesky factor.

The ICF fails to converge in almost all test problems mainly due to the criteria used to avoid loss of positive definiteness. In this ICF code when diagonal element is very small or negative it is substituted by 1.

The KABAPAH2 problem belongs to a degenerated problems collection in which robust implementations such as PCx and HOPDM [18] fail to find the optimal solution. The hybrid approach succeeds to converge in eight iterations.

The PDS test problems do generate a very sparse $A\Theta A^T$ matrix. On the other hand, the Cholesky factorization generates a large number of fill-in entries. As the dimension of the problem increases, the hybrid approach performs better compared to the direct and the ICF approaches. Figure 1 illustrates this fact. For these instances the criteria to change phases is not reached since the CCF preconditioner has good performance during all interior point iterations. Moreover, the nonzero entries in CCF preconditioner are not increased.

Table 4 Comparisons between iterative hybrid and ICF and direct solver. F denotes failure; M denotes memory exceeded

Problem	IT			Time performance		
	Hybrid	Cholesky	ICF	Hybrid	Cholesky	ICF
KBAPAH2	8	13	F	0.0	0.0 ^a	F
ELS-19	31	30	F	290.1	497.1	F
CHR22B	32	28	F	26.2	29.1	F
CHR25A	32	31	F	87.4	91.9	F
SCR15	24	22	F	44.1	78.6	F
SCR20	21	21	F	234.0	910.6	F
ROU20	24	21	F	2288.4	5807.8	F
STE36A	37	M	F	36545.7	M	F
STE36B	27	M	F	24327.1 ^a	M	F
STE36C	41	M	F	80142.3	M	F
QAP12	21	46	F	449.1	340.6 ^a	F
QAP15	23	54	F	3247.1	2977.4 ^a	F
SCSD8-2B-64	7	F	F	6.6	F	F
SCSD8-2C-64	7	F	F	6.4	F	F
SCSD8-2R-432	18	F	F	65.0	F	F
NUG08	10	12	F	3.8	3.15	F
NUG12	21	51	F	349.1	609.1 ^a	F
NUG15	24	44	F	2971.1	3885.7 ^a	F
PDS-20	61	61	61	1258.3	1197.8	2579.8
PDS-40	78	74	77	2811.8	13246.1	6642.1
PDS-60	80	75	79	7254.2	41460.3	19335.8
PDS-80	83	81	83	11337.1	94756.1	31397.9
PDS-100	87	M	87	16540.4	M	48736.3

^aDenotes that the problem was nearly solved

Table 6 shows the influence of the parameter η in the convergence of conjugate gradient (ITCG) during the k th interior point iteration. A diagonal scaling preconditioner is used in initial iterations (1–3). While conjugate gradient loses efficiency η is increased allowing more fill-in in the CCF preconditioner. This procedure continues until the criteria of change of phases is achieved. The $|M|$ column reports the number of nonzero entries of the CCF and splitting preconditioners.

The QAP test problems also lead to normal equations matrices that are not very sparse. The Cholesky factorization has a large number of nonzero entries and it makes the direct approach less effective. Using a negative parameter η it is possible to build an efficient preconditioner of phase I with a small cost.

The STE36 problems have hundreds of thousands nonzero entries in the complete Cholesky factor. No results for the problems PDS-100, STE36A, STE36B and STE36C are reported for Cholesky approach because it would take a large amount of time and memory to solve these problems.

Table 5 The influence of fill-in on the hybrid approach performance

Problem	η_0	$ M $	$A \Theta A^T$	Matrix L
KBAPAH2	8	120	120	120
ELS-19	-31	63705	137825	3849458
CHR22B	-27	19784	153680	1453225
CHR25A	-30	28821	249324	2653126
SCR15	-26	27060	59009	1330759
SCR20	-32	1010537	166709	6672137
ROU20	-48	4995689	356689	20818131
STE36A	-56	3219943	1564487	176625274
STE36B	-56	2422737	1564487	176625274
STE36C	-56	2486979	1564487	176625274
QAP12	-31	468892	60174	2138580
QAP15	-27	1493922	155986	8197968
SCSD8-2B-64	-138	5130	709145	27026045
SCSD8-2B-64	-138	5130	709145	27026045
SCSD8-2R-432	-226	8650	1956985	76779485
NUG08	-12	33849	9274	233032
NUG12	-20	658470	56405	2793152
NUG15	-26	1762211	150470	11053969
PDS-20	5	307339	169915	7089645
PDS-40	5	631026	341419	28195225
PDS-60	5	981789	523496	58118583
PDS-80	5	1271825	676093	94270275
PDS-100	5	1512937	1060567	120240013

Table 6 The influence of parameter η in phase I of the NUG12 problem

k	η	ITCG	$ M $	k	ITCG	$ M $
1	-20	616	2794	10	759	612068
2	-20	628	2794	11	876	608006
3	-20	2218	2794	12	464	603650
4	-10	1673	26994	13	385	628561
5	0	504	56405	14	495	640283
6	0	509	56405	15	281	629110
7	0	941	56405	16	315	620448
8	10	1174	83946	17	290	596487
9	20	1884	111433	18	234	629142
				19	314	617925
Change of phases				20	527	632490
				21	355	658470

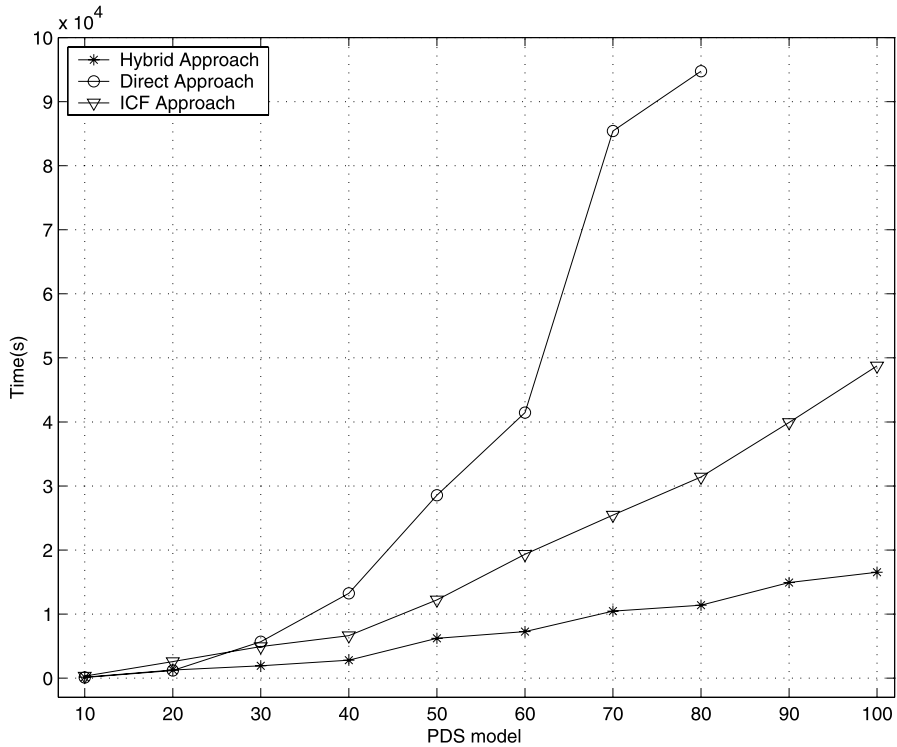


Fig. 1 PDS model—Hybrid versus Direct and ICF approaches

6 Conclusions

We have provided computational evidence that the hybrid approach works better than direct approach for some classes of large linear programming problems. This fact occurs when the Cholesky factor has a large number of fill-in entries. The preconditioner used in the phase I has fewer nonzero entries than complete Cholesky factor. Another feature that helps the hybrid approach to work better is that the number of LU factorizations for building the preconditioner used in phase II is very small compared to the number of iterations.

The performance of the hybrid approach will be improved if the optimal time of changing phases could be identified. We are developing an efficient procedure for this. One idea to be tested is merging the two preconditioners to identify the best iteration to change the phases. Other factor to be considered to improve performance is the adjustment of the optimal parameter η used in phase I. This parameter can be increased during iterations as the systems become ill-conditioned.

Acknowledgements We gratefully acknowledge the three anonymous referees for their helpful suggestions for improvement of both presentation and contents of this article. The authors would like to thank the Brazilian institutions CAPES, CNPq, FAPEMIG and FAPESP for financial support.

References

1. Adler, I., Karmarkar, N.K., Resende, M.G.C., Veiga, G.: Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA J. Comput.* **1**(2), 84–106 (1989)
2. Adler, I., Karmarkar, N.K., Resende, M.G.C., Veiga, G.: An implementation of Karmarkar's algorithms for linear programming. *Math. Program.* **44**, 297–335 (1989)
3. Altman, A., Gondzio, J.: Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization. *Optim. Methods Software* **11**, 275–302 (1999)
4. Ašić, M.D., Kovačević-Vujčić, V.V.: Ill-conditionedness and interior-point methods. *Univ. Beograd. Publ. Elektrotehn. Fak. Ser. Mat.* **11**, 53–58 (2000)
5. Baryamureeba, V.: Solution of large-scale weighted least squares problems. *Numer. Linear Algebra Appl.* **9**, 93–106 (2002)
6. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. *Acta Numer.* **14**, 1–137 (2005)
7. Bergamaschi, L., Gondzio, J., Zilli, G.: Preconditioning indefinite systems in interior point methods for optimization. *Comput. Optim. Appl.* **28**(2), 149–171 (2004)
8. Bunch, J.R., Parlett, B.N.: Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.* **8**, 639–655 (1971)
9. Campos, F.F.: Analysis of conjugate gradients—type methods for solving linear equations. Ph.D. thesis, Oxford University Computing Laboratory, Oxford (1995)
10. Campos, F.F., Birkett, N.R.C.: An efficient solver for multi-right hand side linear systems based on the CCCG(η) method with applications to implicit time-dependent partial differential equations. *SIAM J. Sci. Comput.* **19**(1), 126–138 (1998)
11. Czyzyk, J., Mehrotra, S., Wagner, M., Wright, S.J.: PCx an interior point code for linear programming. *Optim. Methods Software* **11**(2), 397–430 (1999)
12. Dollar, H.S., Gould, N.I.M., Wathen A.J.: On implicit-factorization constraint preconditioners. Technical Report RAL-TR-2004-036, Rutherford Appleton Laboratory. Also In: Di Pillo, G., Roma, M., (eds.) *Large Scale Nonlinear Optim.* Springer (2004, to appear)
13. Duff, I.S., Grimes, R.G., Lewis, J.G.: User's guide for the Harwell–Boeing sparse matrix collection (release I). Technical Report PA-92-86, CERFACS, Toulouse, France (1992)
14. Durazzi, C., Ruggiero, V.: Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems. *Numer. Linear Algebra Appl.* **10**, 673–688 (2003)
15. Forsythe, G.E., Straus, E.G.: On best conditioned matrices. *Proc. Am. Math. Soc.* **6**, 340–345 (1955)
16. George, A., Ng, E.: An implementation of Gaussian elimination with partial pivoting for sparse systems. *SIAM J. Sci. Stat. Comput.* **6**, 390–409 (1985)
17. Gill, P.E., Murray, W., Pongcelón, D.B., Saunders, M.A.: Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix Anal. Appl.* **13**, 292–311 (1992)
18. Gondzio, J.: HOPDM (Version 2.12)—A fast LP solver based on a primal–dual interior point method. *Eur. J. Oper. Res.* **85**, 221–225 (1995)
19. Gondzio, J.: Multiple centrality corrections in a primal–dual method for linear programming. *Comput. Optim. Appl.* **6**, 137–156 (1996)
20. Gould, N.I.M., Hribar, M.E., Nocedal, J.: On the solution of equality constrained quadratic programming problems arising in optimization. *SIAM J. Sci. Comput.* **23**, 1376–1395 (2001)
21. Haws, J.C., Meyer, C.D.: Preconditioning KKT systems. Online at http://meyer.math.ncsu.edu/Meyer/PS_Files/KKT.pdf
22. Hungarian Academy of Sciences OR Lab: Miscellaneous LP models. Online at http://www.sztaki.hu/~meszaros/public_ftplptestset/misc
23. Hungarian Academy of Sciences OR Lab: Stochastic LP test sets. Online at http://www.sztaki.hu/~meszaros/public_ftplptestset/stochlp
24. Jones, M.T., Plassmann, P.E.: An improved incomplete Cholesky factorization. *ACM Trans. Math. Software* **21**, 5–17 (1995)
25. Karisch, S., Burkard, R.S., Rendl, F.: QAPLIB—a quadratic assignment problem library. *Eur. J. Oper. Res.* **55**, 115–119 (1991)
26. Keller, C., Gould, N.I.M., Wathen, A.J.: Constraint preconditioning for indefinite linear systems. *SIAM J. Matrix Anal. Appl.* **21**(4), 1300–1317 (2000)
27. Lustig, I.J., Marsten, R.E., Shanno, D.F.: On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM J. Optim.* **2**, 435–449 (1992)

28. Manteuffel, T.A.: An incomplete factorization technique for positive definite linear systems. *Math. Comput.* **34**(150), 473–497 (1980)
29. Mehrotra, S.: On the implementation of a primal–dual interior point method. *SIAM J. Optim.* **2**(4), 575–601 (1992)
30. Mittelmann–LP models. Miscellaneous LP models collect by Hans D. Mittelmann. Online at <ftp://plato.asu.edu/pub/lptestset/pds>
31. Munksgaard, N.: Solving sparse symmetric sets of linear equations by preconditioned conjugate gradients. *ACM Trans. Math. Software* **6**(2), 206–219 (1980)
32. NETLIB LP repository: NETLIB collection LP test sets. Online at <http://www.netlib.org/lp/data>
33. Oliveira, A.R.L.: A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. Ph.D. thesis, Department of Computational and Applied Mathematics, Rice University, Houston (1997)
34. Oliveira, A.R.L., Sorensen, D.C.: A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra Appl.* **394**, 1–24 (2005)
35. Padberg, M., Rijal, M.P.: *Location, Scheduling, Design and Integer Programming*. Kluwer Academic, Boston (1996)
36. Portugal, L.F., Resende, M.G.C., Veiga, G., Júdice, J.J.: A truncated primal-infeasible dual-feasible network interior point method. *Networks* **35**, 91–108 (2000)
37. Resende, M.G.C., Veiga, G.: An implementation of the dual affine scaling algorithm for minimum cost flow on bipartite uncapacitated networks. *SIAM J. Optim.* **3**, 516–537 (1993)
38. SLATEC collection. The dsics.f subroutine in NETLIB repository. Online at <http://www.netlib.org/slatec/lin/dsics.f>
39. Wang, W., O’Leary, D.P.: Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming. *Numer. Algorithms* **25**(1–4), 387–406 (2000)