A note on hybrid preconditioners for large-scale normal equations arising from interior-point methods

M. I. Velazco * A. R. L. Oliveira * F. F. Campos †

Abstract

A previously developed approach for solving linear systems arising from interior-point methods applied to linear programming problems is considered and improved upon. The preconditioned conjugate gradient method is used to solve these systems in two different phases of the interior-point method: during the initial interior-point iterations, an incomplete Cholesky factorization preconditioner with controlled fill-in is used; in the second phase, near the optimal solution, a specialized preconditioner based upon the LU factorization is used to combat the high ill-conditioning of the linear systems in this phase. This approach works better than direct methods on some classes of large-scale problems. New heuristics are presented to identify the change of phases, thus achieving better computational results and solving additional problems. Moreover, new orderings of the constraint matrix columns are presented allowing savings in the preconditioned conjugate gradient method iteration number. Experiments are performed with a set of large-scale problems and both approaches are compared with respect to the number of iterations and running time.

Keywords: Interior-point methods, Preconditioners, Iterative methods.

1 Introduction

The most expensive step of an interior-point method is the computation of the search direction through the solution of one or more linear systems. Such systems are indefinite and can be written in a symmetric form known as an augmented system. A common approach reduces the augmented system to a smaller positive definite system called the normal equations. Usually they are solved by direct methods. However, for some classes of large-scale problems, the use of direct methods becomes prohibitive because of storage and running-time limitations. In such situations, iterative approaches are more interesting.

The performance of implementations using iterative methods depends on the choice of an appropriate preconditioner, in particular, the linear system becomes highly ill-conditioned as an 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, 18, 22, 26]. Some of them are based on incomplete Cholesky factorization of the normal equations system. Typically, this class of preconditioners is efficient in the 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 denser than the augmented system and this has motivated the study of methods to solve the indefinite system [5, 9, 12, 13, 2, 11, 17, 20, 23]. The preconditioner for the augmented system proposed in [20] is based on an LU factorization and has been shown to perform well near the solution of the linear programming problem but it gives poor results during the initial interior-point iterations.

In [6] an iterative hybrid approach was proposed for solving 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 an incomplete factorization known

*Applied Mathematics Department - State University of Campinas, C.P. 6065, 13.083-970 Campinas - SP, Brazil; (velazco,aurelio@ime.unicamp.br).

†Computer Science Department - Federal University of Minas Gerais, Av. Antônio Carlos, 6627, 31.270-010 Belo Horizonte - MG, Brazil; (ffcampos@dcc.ufmg.br).

as the controlled Cholesky factorization (CCF) [8] and in the remaining iterations (*phase II*) by the splitting preconditioner developed in [20]. The amount of memory used by the 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. A heuristic has been developed to determine the change of phases. The hybrid approach is applied within the PCx code [10].

In this work, the change of phases heuristics developed in [6] is improved upon and the column splitting rule in [20] is modified. As a result, the computational running times obtained in [6] are reduced and more of the large-scale problems can now be solved by the hybrid approach.

2 Primal-Dual interior-point methods

Primal-dual interior-point methods are widely used for solving large-scale linear programming problems [27]. Consider the linear programming problem in the standard 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 $m \leq n$. We remark that the following discussion can be easily extended when problems with bounded variables are considered.

The dual problem associated with problem (1) is

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

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

$$\begin{aligned} Ax - b &= 0 \\ A^T y + z - c &= 0 \\ XZe &= 0 \\ (x, z) &\geq 0, \end{aligned} \tag{3}$$

where $e \in \mathbb{R}^n$ is the vector of all ones and $X = diag(x)$, $Z = diag(z)$. The optimality conditions (3) are a system of nonlinear equations. The Primal-dual methods are, therefore, based on Newton's method applied to the optimality conditions without the nonnegativity of the (x, z) components. Mehrotra's predictor-corrector method [19] is one of the most successful variant amongst the class of interior-point methods. The search direction is computed by solving two linear systems which have the same coefficient matrix but different right-hand sides. The affine-scaling directions are computed by finding the solution of

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I_n \\ Z^k & 0 & X^k \end{bmatrix} \begin{bmatrix} \Delta_a x^k \\ \Delta_a y^k \\ \Delta_a z^k \end{bmatrix} = \begin{bmatrix} r_p^k \\ r_d^k \\ r_a^k \end{bmatrix} \tag{4}$$

where $r_p^k = b - Ax^k$, $r_d^k = c - A^T y^k - z^k$ and $r_a^k = -X^k Z^k e$. 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 (4) is replaced by $r_d^k = 0$, $r_p^k = 0$, $r_a^k = \mu_k e - \Delta_a X^k \Delta_a Z^k e$ and the resulting systems solved. Here μ^k is a 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. The overall search directions may be

computed solving system (4) replacing r_a^k by $\mu^k e - \Delta_a X^k \Delta_a Z^k e - X^k Z^k e$ and solving a second linear system.

The predictor-corrector version can be summarized as follows:

Method 1 (Predictor-corrector version).
Given y^0 *and* $(x^0, z^0) > 0$.
For $k = 0, 1, 2, \ldots,$ *do*

(1) *Solve linear system (4).*

(2) *Compute* $\sigma^k = \left(\frac{\tilde{\gamma}^k}{\gamma^k}\right)^2$ *and set* $\mu^k = \sigma^k \left(\frac{\gamma^k}{n}\right)$ *where,* $\gamma^k = (x^k)^T z^k$ *and* $\tilde{\gamma}^k = (x^k + \tilde{\alpha}^k \Delta x^k)^T (z^k + \tilde{\alpha}^k \Delta z^k)$.

(3) *Solve linear system (4) replacing* r_a^k *by* $\mu^k e - \Delta_a X^k \Delta_a Z^k e - X^k Z^k e$.

(4) *Form the new iterate* $(x^{k+1}, y^{k+1}, z^{k+1}) = (x^k, y^k, z^k) + \alpha^k (\Delta x^k, \Delta y^k, \Delta z^k)$.

3 Linear system solution

The most expensive step in interior-point methods is the computation of search directions. Since both predictor-corrector systems share the same coefficient matrix, we will only discuss the linear system given by (4). In practice, variables Δz (from now on the superscript k will be dropped for clarity reason) are eliminated and the system reduces to the augmented indefinite linear system, which can be written in a symmetric form,

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r_d - X^{-1} r_a \\ r_p \end{bmatrix}, \tag{5}$$

where $\Theta = Z^{-1} X$. Notice that X^{-1} and Z^{-1} are well defined by definition of interior-point. Given that the matrix A is full rank, the system (5) can be reduced to a smaller positive definite system called the normal equations by eliminating Δx from the first equation,

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

Direct or iterative methods can be applied to solve either system (5) or (6). The most common approach used in interior-point methods for solving the normal equations system is the Cholesky factorization. This approach has the advantage of working with symmetric positive definite matrices. However, it can be very expensive if the Cholesky factor is dense. For instance, the presence of few dense columns in A causes loss of sparsity in $A \Theta A^T$.

One way around this problem is the use of iterative methods. The classical approach adopted to solve the normal equations system is the preconditioned conjugate gradient method. 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. For this reason, the augmented system strategy has been considered in several papers [4, 9, 13, 17, 20] even though it is indefinite.

However, the Cholesky factorization cannot be applied to such system since there is no numerically stable way to factor a general indefinite matrix onto LDL^T with D diagonal. Another alternative that has been used in interior-point methods changes the indefinite system to a quasidefinite one applying 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 has also been used with iterative methods [5]. For indefinite systems, the conjugate gradient method is not guaranteed to achieve convergence. However, it was successfully applied with a convenient preconditioner [5, 13].

Most preconditioners for indefinite systems from interior-point methods are developed for solving nonlinear and quadratic programming problems and almost always 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 [20].

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 application of the conjugate gradient method. Moreover, there are variants of the conjugate gradient method which take into account the normal equations structure of the linear system [25]. Such variants are not exploited in this work. Finally, since this class was developed for the last interior-point iterations, an alternative preconditioner is necessary for the initial iterations.

4 A Hybrid preconditioner

The matrix Θ changes significantly from one interior-point iteration to the next 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.

In [6] it was proposed to apply the conjugated gradient method for solving system (6) preconditioned by a hybrid preconditioner matrix M,

$$M^{-1}(A\Theta A^T)M^{-T}\bar{y} = M^{-1}(A\Theta(r_d - X^{-1}r_a) + r_p), \tag{7}$$

where $\bar{y} = M^T\Delta y$. This 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 and this method is briefly described in Section 4.1. After the change of phases, matrix M will be built using the splitting preconditioner, which is presented in Section 4.2.

4.1 Controlled Cholesky Factorization preconditioner

The Controlled Cholesky Factorization (CCF) preconditioner was designed for solving general positive definite systems [8] and it was successfully applied to solve linear systems from implicit time-dependent partial differential equations [8]. This factorization makes it possible to control fill-in with predictable memory requirements.

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

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

It is easy to see that when $\tilde{L} \approx L \Longrightarrow E \approx 0 \Longrightarrow \tilde{L}^{-1}(A\Theta A^T)\tilde{L}^{-T} \approx I_m$. It is assumed that the matrix $A\Theta A^T$ has been diagonally scaled to give a unit diagonal [14] in order to improve robustness. The CCF is based on the minimization of the Frobenius norm of E. Thus, we consider the problem

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

where l_{ij} represents the ij-th entry of L.

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 jth 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 jth 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 $\eta \to m$

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. Moreover, $\|E\|_F$ is further minimized by choosing the $t_j + \eta$ largest entries of $\tilde{L}$ in absolute value almost annihilating the corresponding largest entries in L leaving only the smallest l_{ij} in the second summation. The preconditioner $\tilde{L}$ is built by columns. Consequently it needs only the jth column of $A\Theta A^T$ at each time allowing great savings of memory or their computation in parallel when more than one processor is available.

The main features of the CCF preconditioner are described in detail in [6] and can be summarized as follows:

- Choice of entries by value, allowing us to work with the largest ones;
- Generalization of improved incomplete factorization;
- Avoiding loss of positive definiteness by exponential shift;
- Versatility preconditioner: the number of nonzero entries per column can vary from 1 to m;
- Predictable storage.

4.2 The Splitting preconditioner

The splitting preconditioner was proposed for indefinite systems arising from interior-point methods [20] for linear programming problems. This preconditioner is a generalization of those proposed by Resende and Veiga [24] 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 welcome feature since the linear system is known to be very ill-conditioned close to a solution and these systems are difficult to be solved 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 [20]:

It is possible do define $P \in \mathbb{R}^{n\times n}$ a permutation matrix, such that $A = [B\ N]P$ where $B \in \mathbb{R}^{m\times m}$ is nonsingular and $N \in \mathbb{R}^{n\times(n-m)}$, since A is full row rank. Thus,

$$A\Theta A^T = [B\ N]P^T\Theta P \begin{bmatrix} B^T \\ N^T \end{bmatrix} = [B\ N] \begin{bmatrix} \Theta_B & 0 \\ 0 & \Theta_N \end{bmatrix} \begin{bmatrix} B^T \\ N^T \end{bmatrix} = B\Theta_B B^T + N\Theta_N N^T.$$

Now, the preconditioner is given by $\Theta_B^{-\frac{1}{2}} B^{-1}$ and the preconditioned matrix T is the following:

$$T = \Theta_B^{-\frac{1}{2}} B^{-1}(A\Theta A^T)B^{-T}\Theta_B^{-\frac{1}{2}} = I_m + WW^T, \tag{8}$$

where $W = \Theta_B^{-\frac{1}{2}} B^{-1} N \Theta_N^{\frac{1}{2}} \in \mathbb{R}^{m\times(n-m)}$.

The product $B^{-1}N$ 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 smallest eigenvalue $\lambda_{min}(T)$ and the largest eigenvalue $\lambda_{max}(T)$ approach the value 1 and thus its condition number, the ratio $\kappa_2(T) = \frac{\lambda_{max}(T)}{\lambda_{min}(T)}$, too.

The price paid for avoiding the normal equations system is to find B and solve linear systems using it. However, the factorization $QB = LU$, where $Q \in \mathbb{R}^{m\times m}$ is a permutation matrix, is typically easier to compute than the Cholesky factorization. In fact, it is known [15] 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.

A strategy to form B is to minimize $\|W\|$ since close to a solution the preconditioned matrix approaches the identity for a suitable choice of B's columns due to the diagonal entries of Θ_B^{-1} and Θ_N values. This problem is hard to solve. An approximate solution was obtained in [20] by selecting the first m linearly independent columns of $A\Theta$ with smallest 1-norm to form B.

4.2.1 Computing the splitting preconditioner efficiently

A naive implementation of the splitting preconditioner can be very expensive, mainly because B is hard to find. A detailed description of how to implement it efficiently can be found in [20]. Next, the main topics are briefly commented.

A nice property of the splitting preconditioner is that we can work with the selected set of columns for some iterations. As a consequence, B can be kept and the preconditioner is very cheap to compute for such iterations since only the diagonal matrix Θ changes.

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

One of the main drawbacks of a straightforward implementation 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-ill occurs. The independent columns found thus far are reordered by the number of nonzero entries. The factorization is then started from scratch using the new ordering and the process repeated until m independent columns are found.

A second factorization is then applied to the chosen set of independent columns using standard techniques for computing an efficient sparse LU factorization. As a welcome side effect, it is not necessary to store U in the previous factorization that determines B.

Before starting the first factorization, the sparse pattern of the ordered matrix is studied in order to identify columns that are symbolically (in)dependent, that is, columns that are linearly (in)dependent in structure no matter the numerical values of their nonzero entries. Those columns that are dependent are discarded while the independent ones are moved to the front and reordered among them as mentioned early. This technique does not perform any floating point operation and leads to faster implementations.

4.3 A new column ordering

In [20] the first m linearly independent columns of $A\Theta$ with smallest 1-norm are selected to form B aiming to obtain W close to the zero matrix. However, the 1-norm has a tendency to diminish the effect of outliers, a feature not desirable in this context since the goal is to split the columns in two sets of size m and $n-m$, respectively. In fact, we would like to stress the so called "outliers". In this work we are using the 2-norm instead of the 1-norm, avoiding this tendency [2]. This choice have improved the performance of the splitting preconditioner for most problems. It also allows better computational results after the change of phases and reduces the number of iterations for the convergence of the conjugate gradient method. Experiments with the ∞-norm show that it does not obtain better performance in comparison with the 2-norm in a similar fashion as the results obtained by the 1-norm. It seems that it happens because the ∞-norm is given by the value of one single entry instead of a combination of the whole column.

Experiments ordering the columns of $A\Theta^{\frac{1}{2}}$ were also performed, although the results did not improve for any of the three norms tested in comparison with the ordering given by columns of $A\Theta$.

4.4 Change of phases

In a previous work [6], the change of phase was performed when the initial gap ($x_0^T z_0$) is reduced by a factor of 10^6 or the number of inner iterations of preconditioner iterative method reaches $\frac{m}{2}$. This change of phase actually happens in a very early stage of the optimization process and the hybrid approach works better when the change occurs almost at the end of the process.

In this work, a new change of phase is presented. If the number of iterations needed for the conjugate gradient method to achieve convergence is greater than $\frac{m}{6}$ the parameter η in the controlled Cholesky factorization is updated: $\eta \leftarrow \eta + 10$. The switch now happens when η falls above the maximum allowed. In the next section comparative numerical experiments are presented with both heuristics, showing that the new one obtains better computational results.

5 Numerical experiments

The hybrid approach was integrated in the PCx code [10]. The procedures for solving linear systems were coded in C, except the controlled Cholesky factorization, which was implemented in FORTRAN. The PCx's default parameters are adopted except the multiple centrality corrections [16] that was disabled. All tests were performed on an Intel Core 2 Duo 2.2GHz processor with 2Gb of memory under Linux using the gcc and gfortran compilers.

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, the criteria change to $||r^k|| < 10^{-8}$. The conjugate gradient method maximum number of iterations is given by the system dimension.

5.1 Test problems

All test problems are public domain linear problems. Most QAP models tested here are from the QAPLIB collection [7] with the modification described in [21]. Some of the NUG problems, however, are not modified. To clarify when one of the NUG problems has been modified, the letter M is added to the end of the problem's name. Table 1 summarizes the test data. The number of rows and columns refer to preprocessed problems.

5.2 Computational results

The behavior of both hybrid approaches is presented in three tables. In Table 2 the results using the previous heuristic approach with 1-norm and 2-norm for reordering the columns of A are presented. For each test problem IPM represents the interior-point method outer iterations number, $Time$ indicates the total CPU time in seconds and PCG represents the average number of iterations of the iterative method after the change of phase occurs. For some problems the change of preconditioner does not happen and the symbol (-) is used to represent that situation. In some problems the number of iterations for iterative methods is reduced, although the total time of processing does not reduce. This is due to the fact that the preconditioner with the new ordering is less sparse leading to more expensive inner-iterations.

The previous heuristic does not achieve good performance for the NUG problems. The change of phases is triggered in an early stage of the optimization process. For all tested problems better computational results are achieved when the CCF is used until nearly the end of the optimization process. Table 3 shows the results when the new change of phase is used for both the 1-norm and 2-norm column reordering. Therefore, the change of preconditioner occurs in the same iteration. For each test problem IPM represents the number of outer iterations, $Time$ indicates the total CPU time in seconds and PCG represents the average number of iterations of the iterative method

after the change of preconditioner. For some problems the change of phase does not occur and the symbol (-) is used to represent that.

Table 4 shows a comparison between the two heuristic approaches. For each test problem IPM represents the number of outer iterations, $Time$ indicates the total CPU time in seconds and PCG represents the average number of iterations of the iterative method after the change of preconditioner. The results show that the processing time was reduced in most of test problems. In some cases the running time was not reduced, although the average of inner-iterations reduces. The set of NUG problems show that the new rules for change of phases and the new column ordering improve robustness of this approach allowing it to solve more problems. We remark that the direct approach for the largest NUG problems fails either due to lack of memory or to the very long time it takes to achieve convergence. Experiments have shown that better results are obtained with the ∞-norm in a few cases. However, the overall better results are achieved when the 2-norm is used in the column ordering.

6 Conclusions

This work presents new heuristics for the change of phases of a hybrid preconditioner for solving linear systems arising from interior-point methods. We have provided computational evidence that with the new change of phases more problems are solved and the running time to solve many other problems is enhanced.

Additionally, a new column ordering is proposed. In the previous approach, the 1-norm was used to reorder the columns of $A\Theta$. The numerical experiments with 2-norm and ∞-norm show that 2-norm improves the performance of the splitting preconditioner for most problems. The new column ordering reduces the number of iterations for the convergence of the conjugate gradient method and the total processing time in most tested instances. Experiments adopting the ordering of the columns according to the norm of $A\Theta^{\frac{1}{2}}$ instead of $A\Theta$ were also performed but this approach has failed to improve the results.

Acknowledgments The authors would like to thank the Foundation for the Support of Research of the State of São Paulo (FAPESP) and the Brazilian Council for the Development of Science and Technology (CNPq).

References

[1] I. Adler, M. G. C. Resende, G. Veiga, and N. Karmarkar, *An implementation of Karmarkar's algorithm for linear programming*, Mathematical Programming, 44 (1989), pp. 297–335.

[2] G. AL-Jeiroudi, J. Gondzio, and J. A. J. Hall, *Preconditioning indefinite systems in interior point methods for large scale linear optimization*, Optim. Methods Softw., 23 (2008), pp. 345–363.

[3] A. Altman and J. Gondzio, *Regularized symmetric indefinite systems in interior point methods for linear and quadratic optimization*, Optimization Methods and Software, 11 (1999), pp. 275–302.

[4] L. Bergamaschi, J. Gondzio, M. Venturin, and G. Zilli, *Inexact constraint preconditioners for linear systems arising in interior point methods*, Computational Optimization and Applications, 36 (2007), pp. 137–147.

[5] L. Bergamaschi, J. Gondzio, and G. Zilli, *Preconditioning indefinite systems in interior point methods for optimization*, Computational Optimization and Applications, 28 (2004), pp. 149–171.

[6] S. Bocanegra, F. F. Campos, and A. R. L. Oliveira, *Using a hybrid preconditioner for solving large-scale linear systems arising from interior point methods*, Computational Optimization and Applications, 36 (2007), pp. 149–164. Special issue on "Linear Albra Issues arising in Interior Point Methods".

[7] R. S. Burkard, S. Karisch, and F. Rendl, *QAPLIB - A quadratic assignment problem library*, European Journal of Operations Research, 55 (1991), pp. 115–119.

[8] F. F. Campos and N. R. C. Birkett, *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 (1998.), pp. 126–138.

[9] J. S. Chai and K. C. Toh, *Preconditioning and iterative solution of symmetric indefinite linear systems arising from interior point methods for linear programming*, Computational Optimization and Applications, 36 (2007), pp. 221–247.

[10] J. Czyzyk, S. Mehrotra, M. Wagner, and S. J. Wright, *PCx an interior point code for linear programming*, Optimization Methods & Software, 11-2 (1999), pp. 397–430.

[11] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen, *Implicit-factorization preconditioning and iterative solvers for regularized saddle-point systems*, SIAM Journal on Matrix Analysis and Applications, 28 (2006), pp. 170–189.

[12] H. S. Dollar and A. J. Wathen, *Approximate factorization constraint preconditioners for saddle-point matrics*, SIAM Journal of Scientific Computing, 27 (2006), pp. 1555–1572.

[13] C. Durazzi and V. Ruggiero, *Indefinitely preconditioned conjugate gradient method for large sparse equality and inequality constrained quadratic problems*, Numerical Linear Algebra with Applications, 10 (2003), pp. 673–688.

[14] G. E. Forsythe and E. G. Straus, *On best conditioned matrices*, in Proceedings of the Amer. Math. Soc., vol. 6, 1955, pp. 340–345.

[15] A. George and E. Ng, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 390–409.

[16] J. Gondzio, *Multiple centrality corrections in a primal-dual method for linear programming*, Computational Optimization and Applications, 6 (1996), pp. 137–156.

[17] C. Keller, N. I. M. Gould, and A. J. Wathen, *Constraint preconditioning for indefinite linear systems*, SIAM Journal on Matrix Analysis and Applications, 21 (2000), pp. 1300–1317.

[18] S. Mehrotra, *Implementations of affine scaling methods: Approximate solutions of systems of linear equations using preconditioned conjugate gradient methods*, ORSA Journal on Computing, 4 (1992), pp. 103–118.

[19] ——, *On the implementation of a primal-dual interior point method*, SIAM Journal on Optimization, 2 (1992), pp. 575–601.

[20] A. R. L. Oliveira and D. C. Sorensen, *A new class of preconditioners for large-scale linear systems from interior point methods for linear programming*, Linear Algebra and Its Applications, 394 (2005), pp. 1–24.

[21] M. Padberg and M. P. Rijal, *Location, Scheduling, Design and Integer Programming*, Kluwer Academic, Boston, 1996.

[22] L. F. Portugal, M. C. Resende, G. Veiga, and J. J. Júdice, *A truncated primal-infeasible dual-feasible network interior point method*, Networks, 35 (2000), pp. 91–108.

[23] T. Rees and C. Greif, *A preconditioner for linear systems arising from interior point optimization methods*, SIAM J. Sci. Comput., 29 (2007), pp. 1992–2007.

[24] M. G. C. Resende and G. Veiga, *An efficient implementation of a network interior point method*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 12 (1993), pp. 299–348.

[25] Y. Saad, *Iterative Methos for Sparse Linear Systems*, SIAM Publications, SIAM, Philadelphia, PA, USA, 2003.

[26] W. Wang and D. P. O'Leary, *Adaptive use of iterative methods in predictor-corrector interior point methods for linear programming*, Numerical Algorithms, 25 (2000), pp. 387–406.

[27] S. J. Wright, *Primal–Dual Interior–Point Methods*, SIAM Publications, SIAM, Philadelphia, PA, USA, 1997.

Problem	Row	Column	Collection
KBAPAH2	12	28	KBAPAH
ELS-19	4350	13186	QAP
CHR22B	5587	10417	QAP
CHR25A	8149	15325	QAP
SCR15	2234	6210	QAP
SCR20	5079	15980	QAP
ROU20	7359	37640	QAP
STE36A	27683	131076	QAP
STE36B	27683	131076	QAP
STE36C	27683	131076	QAP
QAP12	2794	8856	NETLIB
QAP15	5698	22275	NETLIB
SCSD8-2B-64	5130	35910	STOCHLP
SCSD8-2C-64	5130	35910	STOCHLP
SCSD8-2R-432	8650	60550	STOCHLP
PDS-20	32276	106180	MISC
PDS-40	64265	214385	MISC
PDS-60	96503	332862	MISC
PDS-80	126109	430800	MISC
PDS-100	156243	514577	MISC
NUG08M	742	1632	MISC
NUG12M	2794	8856	MISC
NUG15M	5698	22275	MISC
NUG05	210	225	QAP
NUG05-3rd	1410	1425	QAP
NUG06	372	486	QAP
NUG06-3rd	3972	4686	QAP
NUG07	602	931	QAP
NUG07-3rd	9742	12691	QAP
NUG08	912	1632	QAP
NUG08-3rd	19728	29856	QAP
NUG12	3192	8856	QAP
NUG15	6330	22275	QAP

Table 1: Test problems data.

Problem	1–NORM			2–NORM		
	IPM	Time	PCG	IPM	Time	PCG
KBAPAH2	8	0.01	-	8	0.01	-
ELS-19	31	114.75	328	31	155.97	368
CHR22B	329	34.53	389	29	37.24	433
CHR25A	28	66.68	535	28	68.51	530
SCR15	24	14.81	216	24	18.53	246
SCR20	21	135.28	638	21	179.28	512
ROU20	24	1571.86	648	24	3014.92	555
STE36A	37	19543.78	3029	*	*	*
STE36B	37	47469.21	7343	37	39067.58	6162
STE36C	41	73749.53	6135	*	*	*
QAP12	20	350.47	779	20	345.67	740
QAP15	23	3196.30	1676	*	*	*
SCSD8-2B-64	7	3.52	-	7	3.52	-
SCSD8-2C-64	7	3.44	-	7	3.44	-
SCSD8-2R-432	18	33.95	-	18	33.95	-
PDS-20	60	661.77	-	60	661.77	-
PDS-40	79	1276.17	-	79	1276.17	-
PDS-60	85	3646.18	-	85	3646.18	-
PDS-80	83	4811.76	-	83	4811.7	-
PDS-100	86	8244.13	-	86	8244.13	-
NUG08M	10	2.05	88	10	2.05	89
NUG12M	20	369.31	782	20	345.82	635
NUG15M	*	*	*	*	*	*
NUG05	*	*	*	*	*	*
NUG05-3rd	*	*	*	*	*	*
NUG06	6	0.25	-	6	0.25	-
NUG06-3rd	*	*	*	*	*	*
NUG07	11	1.11	52	11	1.10	53
NUG07-3rd	*	*	*	*	*	*
NUG08	10	3.76	88	10	3.81	89
NUG08-3rd	*	*	*	*	*	*
NUG12	20	370.53	782	20	348.38	635
NUG15	*	*	*	*	*	*

Table 2: Previous change of phases and comparison of the 1-norm/2-norm orderings.
*: means that the method failed. -:means no change of preconditioner

Problem	1–NORM			2–NORM		
	IPM	Time	PCG	IPM	Time	PCG
KBAPAH2	8	0.01	-	8	0.01	-
ELS-19	31	128.67	305	31	144.26	361
CHR22B	29	25.34	240	29	27.61	272
CHR25A	28	66.96	548	28	67.74	503
SCR15	24	16.16	215	24	18.82	189
SCR20	21	147.96	638	21	167.76	622
ROU20	24	1478.82	560	24	2302.25	377
STE36A	38	24492.67	3338	38	29402.3	2553
STE36B	*	*	*	37	32938.95	4795
STE36C	42	54635.78	3639	41	30186.23	4359
QAP12	20	274.06	506	20	248.95	406
QAP15	23	2728.56	1093	23	2815.45	1118
SCSD8-2B-64	7	3.52	-	7	3.52	-
SCSD8-2C-64	7	3.44	-	7	3.44	-
SCSD8-2R-432	18	33.95	-	18	33.95	-
PDS-20	60	661.77	-	60	661.77	-
PDS-40	79	1276.17	-	79	1276.17	-
PDS-60	85	3646.18	-	85	3646.18	-
PDS-80	83	4811.76	-	83	4811.7	-
PDS-100	86	8244.13	-	86	8244.13	-
NUG08M	9	1.57	63	9	1.63	62
NUG12M	20	243.59	500	20	208.92	441
NUG15M	*	*	*	23	2310.06	1819
NUG05	8	0.06	-	8	0.06	-
NUG05-3rd	6	2.18	32	6	2.21	38
NUG06	6	0.19	60	6	0.17	51
NUG06-3rd	7	21.22	-	7	21.22	-
NUG07	11	0.73	59	10	0.76	52
NUG07-3rd	8	65.45	-	8	65.45	-
NUG08	9	3	63	9	3.17	62
NUG08-3rd	9	1047.9	-	9	1047.9	-
NUG12	20	243.38	500	20	208.66	441
NUG15	*	*	*	23	2298.88	1819

Table 3: New change of phases and comparison of the 1-norm/2-norm orderings.
*: means that the method failed. -:means no change of preconditioner

Problem	Previous			New		
	IPM	Time	PCG	IPM	Time	PCG
KBAPAH2	8	0.01	-	8	0.01	-
ELS-19	31	114.75	328	31	144.26	361
CHR22B	29	34.53	389	29	27.61	272
CHR25A	28	66.68	535	28	67.74	503
SCR15	24	14.81	216	24	18.82	189
SCR20	21	135.28	638	21	167.76	622
ROU20	24	1571.86	648	24	2302.25	377
STE36A	37	19543.78	3029	38	29402.3	2553
STE36B	37	47469.21	7343	37	32938.95	4795
STE36C	41	73749.53	6135	41	30186.23	4359
QAP12	20	350.47	779	20	248.95	406
QAP15	23	3196.30	1676	23	2815.45	1118
SCSD8-2B-64	7	3.52	-	7	3.52	-
SCSD8-2C-64	7	3.44	-	7	3.44	-
SCSD8-2R-432	18	33.95	-	18	33.95	-
PDS-20	60	661.77	-	60	661.77	-
PDS-40	79	1276.17	-	79	1276.17	-
PDS-60	85	3646.18	-	85	3646.18	-
PDS-80	83	4811.76	-	83	4811.7	-
PDS-100	86	8244.13	-	86	8244.13	-
NUG08M	10	2.05	88	9	1.63	62
NUG12M	20	369.12	782	20	208.92	441
NUG15M	*	*	*	23	2310.06	1819
NUG05	*	*	*	8	0.06	-
NUG05-3rd	*	*	*	6	2.21	38
NUG06	6	0.25	-	6	0.17	51
NUG06-3rd	*	*	*	7	21.22	-
NUG07	11	1.11	52	10	0.76	52
NUG07-3rd	*	*	*	8	65.45	-
NUG08	10	3.76	88	9	3.17	62
NUG08-3rd	*	*	*	9	1047.9	-
NUG12	20	370.53	782	20	208.66	441
NUG15	*	*	*	23	2298.88	1819

Table 4: New and previous change of phases comparison with the 2-norm ordering.
*: means that the method failed. -:means no change of preconditioner