The augmented system variant of IPMs in two–stage stochastic linear programming computation *

Csaba Mészáros †
E-mail: meszaros@lutra.sztaki.hu

November, 1995
WP 95–11

Abstract

The application of interior point methods (IPM) to solve the deterministic equivalent of two–stage stochastic linear programming problems is a known and natural idea. Experiments show that among the interior point methods, the augmented system approach gives the best performance on these problems. However, most of their implementations encounter numerical difficulties in certain cases, which can result in losing the efficiency. We present our augmented system solver which "automatically" exploits the special behavior of the problems. We investigate special properties of the augmented system which make its use fast and numerically robust. We demonstrate our method by solving a number of large–scale two–stage stochastic linear programming problems, and we compare our solver with `fo1aug` [11, 8] which is considered as a state–of–the–art augmented system implementation of interior point methods.

1 Introduction

Stochastic programming handles mathematical programming problems where some of the given parameters are random variables. The importance of stochastic programming is that a more realistic mathematical model can be formulated when the uncertainty is presented. As a result of the wide variety of applications, a number of different stochastic optimization models appeared in the literature. An important one is as follows

*This work was supported in part by Hungarian Research Fund OTKA T-016413 and T–014102.
†Computer and Automation Research Institute, Hungarian Academy of Sciences

$$
\begin{array}{ll}
\min & c^T x \\
\text{subject to} & Ax = b \\
& \text{“}T(\omega)x = h(\omega)\text{”} \\
& x \geq 0,
\end{array} \tag{1}
$$

where ω is from the probability space Ω, $A \in R^{m_1 \times n_1}$ is a matrix, $T(\omega) \in R^{m_2 \times n_1}$ is a random matrix, $c, x \in R^{n_1}$ and $b \in R^{m_1}$ are vectors and $h(\omega) \in R^{m_2}$ is a random vector. Let us note that (1) is meanless from decision making point of view. The different decision principles, according to the various applications, make (1) right problem. Many real applications, from manufacturing through transportation and financial optimization can be formulated as (1) with the “two–stage” decision principle. In this decision principle the decisions are made in two stages and the observation of the random parameters takes place in between.

In practice, the two–stage stochastic linear programming model can be solved as a linear programming (LP) problem if the probability space Ω has finite number of elements. However, the size of the deterministic problem grows linearly with the number of realizations. Consequently, the effort to handling more uncertainty increases both the model size and complexity. Such problems have always been a challenge for the developers. The birth of the standard input format of stochastic problems [4] and the existence of a public accessible test problem library [14] gave also an impetus for researchers to implement and compare algorithms. As a consequence of the size explosion, numerous methods have been developed to solve the problem by exploiting their specialities. The reader interested can find a detailed and profound summary about the computational methods of stochastic programming in [3].

The specialization of the simplex–based methods are developed in two directions. The first one exploits the special structure of the bases which results in time savings in basis factorizations and in basis updates [27]. The second one is a decomposition approach where the problem is decomposed into smaller parts. This approach contains numerous possibilities for solving stochastic programs. The first result like this was the L-Shaped method [25] which is the straightforward application of the Benders' decomposition.

After four decades of hegemony in theory and practice of the famous simplex method, Karmarkar introduced a novel interior point algorithm in 1984 [17]. Karmarkar's communications led to an explosion of interest among researchers and practitioners of linear programming. One of the results of this is that very efficient implementations of interior point methods have been developed, using advanced sparse matrix techniques. Extensive numerical tests have shown conclusively that an efficient and robust implementation of a logarithmic barrier algorithm can solve very large LP problems substantially faster than a state–of–the–art simplex solver. Therefore, the use of interior point methods to stochastic problems is a natural idea.

However, the straightforward application of interior point methods to two–stage stochastic linear programming problems is very inefficient, due to the structure of the problems, namely that the columns corresponding to the first–stage decisions are dense, which results in heavy fill–in during computations. The problem was recognized by several authors and different solutions were developed.

Gondzio proposed the use of the preconditioned conjugate gradient algorithm. Two–stage stochastic linear problems allow a block–diagonal preconditioner, separable with respect to different realizations of the first–stage variables. This "understands" and exploits well the structure. However, the results reported in [12] show that problems can be solved by large effort only, and the preconditioned conjugate gradient method converge slowly on large problems, result in inaccurate solutions and high computational costs.

An interesting technique was proposed by Birge and Qi [6] who constructed the BQ decomposition to dual block–angular linear programs. The computational results reported in [5] suggests that it substantionally reduces the effort need to compute interior point iterations, but numerical difficulties occurs even by problems with 32 scenarios.

Another possibility is the "column splitting" technique which splits dense columns into shorter pieces. This method requires an additional variable and a constraint per columns split. The application of this approach to two–stage stochastic linear programming problems is discussed in [18]. The conclusion in [8] and the results in general linear programming problems [20] show that this technique is not the most efficient one.

As an alternative to these method, the Schur–complement mechanism [5, 18] and the direct factorization of the augmented system [8] were applied. The experience shows good behavior in fill–in but numerical instabilities in certain cases [18, 8]. The effect of numerical instabilities is either less accurate solutions or the need for recomputation of the pivot agenda. Both cause loss in efficiency of the algorithm.

In the paper we describe an approach which is related to the use of the augmented system as well. Our aim is to highlight ideas which improve the speed of the factorizations and make them numerically stable. For verifying the competence of our approach, we compared our algorithm with the barrier solver `folaug`, implemented by S. Mehrotra [11, 8]. This implementation is generally considered the presentative of the state–of–the–art level in the augmented system based solvers. For checking the performance of the implemented algorithms we selected a Sun Sparc–10 workstation with 128 Mbyte of memory as a common platform, and a number of two–stage stochastic linear programming problems from [14] .

The rest of papers is organized as follows: Section 2 describes the basic methodology of the two–stage stochastic linear programming. In Section 3 we present the augmented system variant of interior point methods. In Section 4 we highlight some of the important properties of the augmented system. Section 5 contains computational techniques which are crucial to the efficiency of the implementation. Then, in the computational experiences reported by Section 6, we demonstrate our approach by solving a number of large–scale two–stage stochastic linear programming problems and present the detailed comparison with `folaug`. We summarize our findings in Section 7.

2 Formulation of the problem

By the two–stage principle, we make some intermediate decision in the first stage of problem (1) without the knowledge of the random parameters. After observing uncertainty (presented by the realization of the random parameters), we correct the decision of stage 1. The aim in the first stage is to find x which can be corrected for all possible realizations, furthermore the sum of the first stage objective and the expected value of the correction cost is minimal :

$$\begin{array}{rl} \min & c^T x + E(\eta(\omega, x)) \\ \text{subject to} & Ax = b \\ & \quad x \geq 0. \end{array} \tag{2}$$

The correction cost is also called as "recourse function" and it can be obtained as

$$\begin{array}{rl} \eta(\omega, x) = & \min\ d(\omega)^T y \\ \text{subject to} & W(\omega)y + T(\omega)x = h(\omega) \\ & y \geq 0, \end{array} \tag{3}$$

where $y \in R^{n_2}$ denote the correction vector, $d(\omega)$ is a n_2 random vector and $W(\omega)$ is a $m_2 \times n_2$ random matrix. Usually, x and y are referred to as first and second stage decision vectors, and $W(\omega)$ is called the recourse matrix. We consider the case if the probability space is of finite number of elements. Let denote d_i, W_i, T_i and h_i for $i = 1, ..., k$ the possible realizations of $d(\omega)$, $W(\omega)$, $T(\omega)$ and $h(\omega)$ with $p_1, p_2, \ldots, p_k$ probabilities, then the solution of the large–scale linear programming problem

$$\begin{array}{llllll} \min & (c^T x + p_1 d_1^T y_1 + \ldots + p_k d_k^T y_k) & & & & \\ \text{Subject to} & Ax & & & = b & \\ & T_1 x & +W_1 y_1 & & = h_1 & \\ & \vdots & \ddots & & \vdots & \\ & T_k x & & +W_k y_k & = h_k & \\ & & x \geq 0,\ y_1 \geq 0, \ldots, y_k \geq 0\,. & & & \end{array} \tag{4}$$

gives the optimal solution of the first stage and all possible second stage problems in one step.

We note that the matrix in (4) has $m_1 + km_2$ rows and $n_1 + kn_2$ columns. This results in very large but sparse linear programming problems where the attention to the structure can be very important. Some different important classes of the general problem (4) are identified in the literature. If W is not random, i.e., $W_1 = W_2 = \cdots = W_k$, then the problem has "fixed recourse". If the second stage of a fixed recourse problem is feasible for arbitrary right–hand side vectors, then we say that the problem has a "complete recourse".

It is a special case when $W = [I, -I]$ is called a "simple recourse" problem. By this, the columns of the second stage decisions indicate upper and lower bounds on the dual problem of (4). Let $\hat{x}$ and $\hat{y}_1, \ldots, \hat{y}_k$ denote the dual variables of (4), then the reduced dual problem can be written as

$$\begin{array}{lll} \max & b^T \hat{x} + \sum_{i=1}^{k} h_i \hat{y}_i & \\ \text{Subject to} & A^T \hat{x} + \sum_{i=1}^{k} T_i^T \hat{y}_i \leq c & \\ & p_i d \geq \hat{y}_i \geq -p_i d & \text{for } i = 1, \ldots k. \end{array} \tag{5}$$

Thus, the remaining problem has only n_1 rows and $m_1 + km_2$ columns, and can be solved more efficiently than (4). Let us observe that in the particular case when $T_1 = T_2 = \cdots = T_k$, problem (5) can be regarded the equivalent of a piecewise linear programming problem with the variable replacement method. Due to the fact that an equivalent piecewise linear programming problem has only n_1 rows and $m_1 + m_2$ columns, a specialized simplex method to piecewise linear programs seems to be the most powerful tool to solve the simple recourse problem in this particular case.

However, if W is a general matrix, no such successful reduction is possible. In the following we are going to discuss the practical application of interior point methods to problem (4). We believe that the importance of IPMs, in addition to their robustness, is that they can serve for solving a wide class of stochastic programs.

3 The augmented system variant of interior point methods

A common experience of extensive numerical tests shows that the logarithmic barrier methods from the family of interior point algorithms are the most powerful in the practice. Here, we do not describe any of their several variants in details. Readers interested in the full development of these algorithms are referred to the survey by Lustig, Marsten and Shanno [19].

We consider the following primal linear programming problem:

$$\begin{array}{ll} \min & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0, \end{array} \tag{6}$$

where A is an $m \times n$ matrix, c and x are n vectors, and b is an m vector. We assume that A is of full row rank.

Let y denote the n vector of dual variables. The dual problem associated with (6) can be written as follows:

$$\begin{array}{ll} \max & b^T y \\ \text{subject to} & A^T y + z = c \\ & z \geq 0, \end{array} \tag{7}$$

where z is the n vector of dual slack variables. Let X denote the diagonal matrix whose diagonal elements are the components of x, i.e., $X = \mathrm{diag}(x)$, and, in a similar manner, let $Z = \mathrm{diag}(z)$. The barrier algorithms generate a sequence of primal and/or dual solutions, which converge to the optimum. The fundamental task of the logarithmic barrier methods is to compute the search direction. In the general case the search direction can be obtained from the following set of equations:

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \pi \end{bmatrix} = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}. \tag{8}$$

In the context of interior point methods this set of equations is usually referred to as the "augmented system". Practically, any interior point method can be discussed within this framework. In every iteration the diagonal matrix D and the right–hand side vector of (8) are changed, corresponding to the rule of the applied algorithm. The solutions Δx and $\Delta \pi$ are used for the determination of the next iterate. Instead of the detailed discussion of the logarithmic barrier methods, we show the choice of D^2, α and β results in the most popular methods:

D^2	α	β	algorithm
X^2	c	0	primal affine scaling
Z^{-2}	μZe	b	dual affine scaling
$Z^{-1}X$	$Z - \mu X^{-1}e$	0	primal-dual path following

(9)

In the following we are going to denote the matrix on the left side of (8) with M. Several methods can be used to solve (8), both direct and iterative ones. In practice, the symmetric decomposition of M is the most important and commonly used approach. By this, the

$$M = L\Lambda L^T \tag{10}$$

factorization is computed where L is lower triangular and Λ is diagonal matrix.

Depending on the pivot agenda, this approach opens many possibilities. One of them is to eliminate first Δx from (8) which is equivalent to pivoting down the diagonal elements of $-D^{-2}$. Then the remaining system will be just the $AD^2A^T\Delta y = AD^2 r + h$ system of normal equations. Since AD^2A^T is positive definite matrix, the Cholesky factorization and in the preparation phase one of the well–known ordering methods can be used. As pointed out in many papers, the positive definite property makes this approach very attractive, but difficulties arise when A contains dense columns.

As an alternative to the normal equation approach, the direct factorization of (10) can be used without prescribing the pivoting in $-D^{-2}$ at the beginning. This scheme was implemented in two different ways: Mehrotra used the Bunch-Parlett factorization [7] which allows 2×2 blocks in Λ as well. As an alternative to this, the simple factorization with 1×1 pivots was suggested [29, 20]. Our investigation in the next section will show that such decomposition always exists on the indefinite matrix M.

The augmented factorization allows more freedom in the pivot sequence, which is important by several structures of A. Even its numerical behavior is better than that one by the normal equations as pointed out in [2]. However, the loss of the positive definiteness results in the possibility of numerical problems during the interior point iterations. Additionally, the initial ordering must be based on numerical issues, which rules out the use of the powerful symbolic orderings.

In [20] the authors concluded that the augmented factorization improves on normal equations only if the stucture of A allows the following partitioning:

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}, \tag{11}$$

where A_1 is sparse with no dense columns or rows, A_2 and A_3 are much smaller in dimensions than A_1 and the density in A_2 is considerably higher than of one in A_1. In [20] the authors describe a heuristic that recognizes this structure and is able to determine which of the two major factorization approaches to use. In the context of two–stage stochastic linear programming, this partitioning naturally exists, namely the columns of the second–stage decision variables play A_1, the columns of the first–stage decision variables play A_2, while A_3 is usually empty. But, in some of the cases it is imaginable that as result of the structure, the difference between the sparsity of A_1 and A_2 is not significant. This means that in certain cases the two–stage stochastic linear programming problem can be solved efficiently with the normal equation system. We return to this question by the explanation of our computational results.

4 On the theoretical background of the factorization

For presenting our ideas we need the general framework of the numerical factorization process on the symmetric system $M = \begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix}$. On the one hand a similar subject was extensively studied both theoretically and computationally in numerical analysis in context of least squares problems (see e.g. [2, 10] and the references therein). On the other hand the use of the augmented factorization in the framework of interior point methods raises some special questions. In this section we present important properties which make the augmented system approach efficient in interior point methods of linear programming.

First we examine the case when A is of full row rank. Thus, because M is of order $m+n$, the elimination process includes $m+n-1$ steps. Let I^s denote the set of row/column indices that were not pivots until the stage s and

$$M^s = \begin{bmatrix} D^s & (A^s)^T \\ A^s & P^s \end{bmatrix}$$

is the active submatrix, i.e., the part of the matrix that has not yet been factorized. With this convention, let denote $M^0 = M$. Thus $A^0 = A$, $D^0 = -D^{-2}$ and P^0 is an $m \times m$ null matrix. It is to be noted that $D^0_{ii} < 0$ for $i = 1, ..., n$ always holds as can be seen from (9). At each step we choose the pivot position from the diagonal of the current active submatrix, which preserves the symmetric structure. At the s-th step of the elimination we choose a pivot index $p \in I^s$ for which $m^s_{pp} \neq 0$. Then, $I^{s+1} = I^s - \{p\}$ and the elements of the active submatrix at stage $s+1$ can be computed as

$$m^{s+1}_{ij} = \begin{cases} m^s_{ij} & \text{if } m^s_{pj} = 0 \\ m^s_{ij} - \dfrac{m^s_{pi} m^s_{pj}}{m^s_{pp}} & \text{if } m^s_{jp} \neq 0 \end{cases} \quad \text{for } (i,j) \in I^{s+1} \times I^{s+1}, \tag{12}$$

The sequence of the pivot indices $\{p\}_1^m$ define the ordering permutation and the factorization (10) is obtained by

$$\begin{aligned} \Lambda_{ii} &= m^{p_i}_{p_i p_i} \quad \text{for } i = 1, ..., m+n, \\ L_{ij} &= \frac{m^{p_i}_{p_i p_j}}{m^{p_i}_{p_i p_i}} \quad \text{for } i,j = 1, ..., m+n,\ p_i \leq p_j. \end{aligned}$$

Due to the fact that M is an indefinite matrix, it does not stand to reason that such elimination is always successful. For example, the symmetric indefinite matrix

$$\begin{bmatrix} 1 & 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 4 \end{bmatrix}$$

can be factorized by 1×1 pivots following the pivot sequence $(2, 3, 1)$. The decomposition, however fails when we choose $(1, 1)$ for the first pivot position. The following statement guarantees that on our system M, the 1×1 pivot scheme is always valid. Having been fixed for a certain D, it is valid for arbitrary diagonal matrix D with positive diagonal values.

Statement: At each elimination step $0 \leq s < m+n$, system D^s is negative definite and system P^s is positive semidefinite.

Proof: (By induction on s.) The proposition is evidently true for $s = 0$. For $s > 0$, let D^{s+1} and P^{s+1} be the matrices obtained from D^s and P^s by elimination (12). If the pivot was chosen from D^s, i.e., if $p \leq n$ then, the negative definiteness of D^{s+1} follows directly from the negative definiteness of D^s. Furthermore, P^{s+1} is computed as

$$P^{s+1} = P^s - \frac{a_p^s (a_p^s)^T}{m_{pp}^s}, \tag{13}$$

where a_p^s denote the p-th column of A^s. Since D^s is negative definite, therefore, $m_{pp}^s < 0$ and the transformation (13) does not change the positive semidefiniteness of P^s. The other case, when $p > n$ may be proven in the similar manner.

It follows from the statement that if A is of full row rank, M can always be factorized by 1×1 pivots. Among them, exactly n pivots are negative and m are positive and for all permitted values of index i, $\Lambda_{ii} > 0 \Longleftrightarrow p_i > n$. Let us observe the next consequnce of the above statement.

Corollary: $P_{ii}^s = 0$ if and only if the submatrix of A consisting of the columns and rows defined by the pivot positions until the sth elimination step and the ith row of A is not of full row rank.

Actually, in our interior point implementation, we perform a numerical factorization by $D = I$ at the very beginning. As result of this factorization, we obtain the pivot agenda and the corresponding sparsity pattern. The numerical result of the decomposition is used for computation of the starting point. The last corollary is true for any diagonal matrix D with positive diagonal values and so, the pivot order is theoretically valid during all interior point iterations. Practically, indefiniteness may cause problems if some of the variables go very close to zero at early stages of the iterations, still far away from optimality. It happens very rarely if we apply a variant of the primal dual method that efficiently follows the central path. If we reach a very small pivot value during the factorization process and there are still some unprocessed columns of the original constraint matrix in the pivot agenda that could modify this diagonal value, then there are two alternatives:

(i) The very small values may simply be changed (see [26, 1]) and be corrected later during the back-solve operation. Unfortunately, this method carries some undesirable computational overhead per back-solve.

(ii) The pivot agenda is modified by moving the small pivot to the end of the pivot sequence. In this way, this pivot will be positive unless A is rank deficient. This process is very simple and fast but it may create some more fill-in in the factorization. According to our experiences, it has hardly any influence on the total solution efficiency.

5 Implementation

The actual performance of an algorithm depends heavily on its implementation. We do not describe any details of the well–known sparse matrix techniques (see e.g. [10]), we believe that they are of limited interest for experienced readers only. We highlight some of the important ideas of our implementation. As noted previously, as the first step, our solver scans the nonzero structure of the constraint matrix to determine the use either the normal equations or the augmented system. If the heuristic selects the normal equations approach, the pivot order is computed by the "inexact" minimum local fill–in ordering [23] which is a pure symbolical method. By the other case, however, we perform the initial numerical factorization with the setting $D = I$ on the augmented system. We note again that this process is used at the beginning of the interior point algorithm only. Later on, during iterations the specialized numerical factorization is used with supernodes and loop unrollings.

The initial factorization process is very similar to the unsymmetric case. This part of our implementation is based on the ideas described in [28] with some modifications to exploit the specialities of the symmetric factorization. Furthermore, since the active submatrix M at each elimination step is symmetric, only the storage of the lower triangular part is necessary. The off–diagonal nonzeros of the lower triangular part of the active submatrix are stored column–wise as a sequence of sparse vectors, and the diagonal of the active submatrix as a full vector. To enable efficient access, we maintain the symbolical row–wise representation as well, as suggested in [28]. The elimination loop begins with the pivot searching. This step involves both numerical and structural issues. Here we use the wonted relative pivot tolerance and the minimum degree criterion [10]. Then, we perform the update (12) on the active submatrix. Our update process consists of three parts. In the first part the pivot column is extracted from the column–wise representation by the help of the information presented by the row–wise structure. Then, we perform the update on the column–wise representation. The last step updates the symbolical row–wise structure. Since our implementation is based on the process described by U. Suhl and L. Suhl, the readers interested in the full development are referred to [28].

We would like to emphasize one of our observations. Namely, in the cases when some of the columns in the constraint matrix contain a number of nonzero elements, the procedure extracting the pivot column from the column–wise representation may slow down dramatically. It happens due to the simple reason that the lower–triangular storage scheme requires scanning some non–pivotal columns in each elimination step to determine all nonzero values of the pivot column. It does not influence the execution speed since the columns are sparse. Especially, in two–stage stochastic linear programs, the sparsity, however, may be not the case.

For increasing the efficiency of the factorization, we perform it in two passes. During the first pass we exclude the columns with large number of nonzeros from the process. This pass terminates if the remaining active submatrix is small, or no acceptable pivot is available. Then, we transform the dense columns corresponding to the stage of the elimination and we include them in the active submatrix. The following second pass accomplishes the factorization. We note that this is a similar technique to the Schur–complement method, but not equivalent with them, due to the numerical investigation during pivot search.

We note that the memory requirement of our implementation is significantly larger than that of the symbolical orderings. But, in context of the interior point methods, it is not a decisive issue. The initial factorization is only a preparation phase of the interior point methods, therefore, all working arrays used by the main algorithm are free at this stage. Since the higher order techniques and the efficient numerical factorization methods require considerably large amounts of memory, usually no other memory array is needed to be reserved for this step.

During iterations, our implementation executes the infeasible primal–dual path following method with Mehrotra's predictor–corrector technique and with the multiple centrality corrections of Gondzio. We do not describe here the ideas and importances of these methods, the reader is referred to the original papers of the authors [21, 13]. However, we made a minor change in the process of Gondzio, namely, we revised the rule for computing the number of corrections suggested by the author. We allowed to use of at least one correction in all cases, independently of the ratio of the computational efforts needed for factorization and backsolving. As the computational results show, in 75% of our test problems, it resulted in change with respect to the original rule of the author. The reason of our revision is that the method of Gondzio stabilizes the interior point method by improving centrality, which can decisively influence the efficiency of the augmented system factorizations.

Our solver is equipped with the supernodal factorization technique with loop unrollings. As reported in [22], this technique results in significant increase in efficiency when the factorization is "dense enough" to create large supernodes. In context of the two–stage stochastic linear programming, however, with the augmented system factorization we obtain extreme sparse factorizations in the overwhelming majority of cases. In such cases the use of the supernodal technique is of minor importance.

6 Computational results

In this study we employ two linear programming solvers. Both are based on the infeasible primal–dual path–following method and on the predictor–corrector logic with higher order extensions. The solver `folaug` is implemented by Mehrotra [11], and uses the augmented system with the Bunch–Parlett factorization. Our solver, called `bpmpd`, is based on the heuristic described in [20] to choose either the augmented factorization with 1×1 pivots or the normal equation system with the "inexact" minimum local fill–in ordering [23]. To compare the two implementations, we selected the two–stage stochastic linear problems from the collection of D. Holmes [14] and the stochastic generalized network problems provided by Mulvey and Vladimirou [24]. Most of the problems were converted to their deterministic equivalent by the SLP–IOR system [15, 16], and the rest by the program of J. Czyzyk.

As a common, platform we used a Sun Sparc–10 workstation in our experiments. Both of the solvers were compiled with the Sun f77 fortran compiler using the -fast -O3 compiler options. The fortran source codes were configured to use at most 128 Mbyte of memory.

Table 1 contains our test problems and their important characteristics. Since the solution time and also the time per iterations are influenced by the current size of the problem that remains after the initial reduction (presolve), we included the characteristics of the initial problems and those remaining after the presolve of bpmpd. We also reported on the length of the longest column in the final problem, which provides with information about the structural properties. We followed the naming conventions described in [14], i.e., where given, the last number in the problem name refers to the number of realizations in the deterministic equivalent problem, and the letter "r", "c" or "b" before it to the problem types: "stochastic right–hand side", "stochastic objective" and "both stochastic right–hand side and objective".

Table 1: Problem statistics

Problem Name	Original Problem No. of rows	No. of columns	No. of nonzeros	Problem after pre-processing No. of rows	No. of columns	No. of nonzeros	Length of long. col.
aircraft	3754	7517	20267	3754	7517	20267	751
cep1	1521	3248	6712	1405	2708	5732	217
pgp2	4034	9220	18440	3640	8100	16200	540
sc205-2r-4	101	102	270	76	66	182	6
sc205-2r-8	189	190	510	140	118	334	10
sc205-2r-16	365	366	990	268	222	638	18
sc205-2r-27	607	608	1650	462	383	1128	29
sc205-2r-32	717	718	1950	524	430	1246	34
sc205-2r-50	1113	1114	3030	832	684	2010	52
sc205-2r-64	1421	1422	3870	1036	846	2462	66
sc205-2r-100	2213	2214	6030	1652	1354	3990	102
sc205-2r-200	4413	4414	12030	3272	2674	7870	202
sc205-2r-400	8813	8814	24030	6492	5294	15550	402
sc205-2r-800	17613	17614	48030	12972	10574	31070	802
scagr7-2r-4	167	180	546	112	167	415	13
scagr7-2r-8	319	340	1050	212	315	795	25
scagr7-2r-16	623	660	2058	412	611	1555	49
scagr7-2r-27	1041	1100	3444	687	1018	2600	82
scagr7-2r-32	1231	1300	4074	812	1203	3075	97
scagr7-2r-54	2067	2180	6846	1362	2017	5165	163
scagr7-2r-64	2447	2580	8106	1612	2387	6115	193
scagr7-2r-108	4119	4340	13542	2604	3907	9863	325
							ctd. →

Problem Statistics, ctd.							
	Original Problem			Problem after pre-processing			
Problem Name	No. of rows	No. of columns	No. of nonzeros	No. of rows	No. of columns	No. of nonzeros	Length of long. col.
scagr7-2r-216	8223	8660	27042	5196	7795	19691	649
scagr7-2r-432	16431	17300	54042	10380	15571	39347	1297
scagr7-2r-864	32847	34580	108042	20748	31123	78659	2593
scagr7-2c-4	167	180	546	112	167	415	13
scagr7-2c-16	623	660	2058	412	611	1555	49
scagr7-2c-64	2447	2580	8106	1612	2387	6115	193
scagr7-2b-4	167	180	546	112	167	415	13
scagr7-2b-16	623	660	2058	412	611	1555	49
scagr7-2b-64	9743	10260	32298	6412	9491	24355	769
scfxm1-2r-4	684	1014	3999	533	909	3130	20
scfxm1-2r-8	1276	1914	7319	993	1713	5674	24
scfxm1-2r-16	2460	3714	13959	1913	3321	10762	36
scfxm1-2r-27	4088	6189	23089	3178	5532	17758	58
scfxm1-2r-32	4828	7314	27239	3753	6537	20938	68
scfxm1-2r-64	9564	14514	53799	7433	12969	41290	132
scfxm1-2r-96	14300	21714	80359	11113	19401	61642	196
scfxm1-2r-128	19036	28914	106919	14793	25833	81994	260
scfxm1-2r-256	37980	57714	213159	29513	51561	163402	516
scfxm1-2c-4	684	1014	3999	533	909	3130	20
scfxm1-2b-4	684	1014	3999	533	909	3130	20
scfxm1-2b-16	2460	3714	13959	1913	3321	10762	36
scfxm1-2b-64	19036	28914	106919	14793	25833	81994	260
scrs8-2r-4	140	189	457	28	33	64	4
scrs8-2r-8	252	341	849	74	90	199	9
scrs8-2r-16	476	645	1633	112	129	256	16
scrs8-2r-27	784	1063	2711	221	269	577	28
scrs8-2r-32	924	1253	3201	224	257	512	32
scrs8-2r-64	1820	2469	6337	448	513	1024	64
scrs8-2r-64b	1820	2469	6337	530	630	1405	65
scrs8-2r-128	3612	4901	12609	1058	1254	2805	129
scrs8-2r-256	7196	9765	25153	2178	2566	5813	257
scrs8-2r-512	14364	19493	50241	4498	5206	11949	513
scrs8-2c-4	140	189	457	28	33	64	4
scrs8-2c-8	252	341	849	56	65	128	8
scrs8-2c-16	476	645	1633	112	129	256	16
							ctd. →

Problem Statistics, ctd.							
	Original Problem			Problem after pre-processing			
Problem Name	No. of rows	No. of columns	No. of nonzeros	No. of rows	No. of columns	No. of nonzeros	Length of long. col.
scrs8-2c-32	924	1253	3201	224	257	512	32
scrs8-2c-64	1820	2469	6337	448	513	1024	64
scrs8-2b-4	140	189	457	28	33	64	4
scrs8-2b-16	476	645	1633	112	129	256	16
scrs8-2b-64	1820	2469	6337	578	678	1557	65
scsd8-2r-4	90	630	1890	90	614	1858	10
scsd8-2r-8	170	1190	3650	170	1094	3458	18
scsd8-2r-8b	170	1190	3650	170	1094	3458	18
scsd8-2r-16	330	2310	7170	330	2246	7042	34
scsd8-2r-27	550	3850	12010	550	3734	11778	56
scsd8-2r-32	650	4550	14210	650	4422	13954	66
scsd8-2r-54	1090	7630	23890	1090	7270	23170	110
scsd8-2r-64	1290	9030	28290	1290	8774	27778	130
scsd8-2r-108	2170	15190	47650	2170	14342	45954	218
scsd8-2r-216	4330	30310	95170	4330	28614	91778	434
scsd8-2r-432	8650	60550	190210	8650	57030	183170	866
scsd8-2c-4	90	630	1890	90	616	1862	10
scsd8-2c-16	330	2310	7170	330	2250	7050	34
scsd8-2c-64	5130	35910	112770	5130	32966	106882	514
scsd8-2b-4	90	630	1890	90	614	1858	10
scsd8-2b-16	330	2310	7170	330	2126	6802	34
scsd8-2b-64	5130	35910	112770	5130	32902	106754	514
sctap1-2r-4	270	432	1516	212	392	1192	14
sctap1-2r-8	510	816	2924	416	760	2348	26
sctap1-2r-8b	510	816	2924	416	760	2348	26
sctap1-2r-16	990	1584	5740	824	1496	4660	50
sctap1-2r-27	1650	2640	9612	1385	2508	7839	83
sctap1-2r-32	1950	3120	11372	1640	2968	9284	98
sctap1-2r-54	3270	5232	19116	2762	4992	15642	164
sctap1-2r-64	3870	6192	22636	3272	5912	18532	194
sctap1-2r-108	6510	10416	38124	5516	9960	31248	326
sctap1-2r-216	12990	20784	76140	11024	19896	62460	650
sctap1-2r-480	28830	46128	169068	24488	44184	138756	1442
sctap1-2c-4	270	432	1516	212	392	1192	14
sctap1-2c-16	990	1584	5740	824	1496	4660	50
							ctd. →

Problem Statistics, ctd.							
	Original Problem			Problem after pre-processing			
Problem Name	No. of rows	No. of columns	No. of nonzeros	No. of rows	No. of columns	No. of nonzeros	Length of long. col.
sctap1-2c-64	3390	5424	19820	2864	5176	16220	170
sctap1-2b-4	270	432	1516	212	392	1192	14
sctap1-2b-16	990	1584	5740	824	1496	4660	50
sctap1-2b-64	15390	24624	90220	13064	23576	74020	770
fxm2-6	1520	2172	12139	1237	1995	10282	22
fxm2-16	3900	5602	31239	3177	5145	26442	36
pltexpA2-6	686	1820	3703	680	1562	3271	8
pltexpA2-16	1726	4540	9233	1710	3852	8081	18
stormG2-8	4409	10193	27424	3647	9583	25093	18
stormG2-27	14441	34114	90903	12197	32060	83670	37
deter0	1923	5468	11173	1923	5467	11154	19
deter1	5527	15737	32187	5527	15736	32134	53
deter2	6095	17313	35731	6095	17312	35650	81
deter3	7647	21777	44547	7647	21776	44474	73
deter4	3235	9133	19231	3235	9132	19160	71
deter5	5103	14529	29715	5103	14528	29666	49
deter6	4255	12113	24771	4255	12112	24730	41
deter7	6375	18153	37131	6375	18152	37070	61
deter8	3831	10905	22299	3831	10904	22262	37

In Table 2 we give some important characteristics of the initial factorizations by the two implementations. As performance indicators, the set–up time (given in seconds, provided by bpmpd only), the number of nonzeros in the factors, the number of operations of one factorization and the average time per factorizations were used. The set–up time refer to the ordering time including one numerical factorization. When giving the number of operations, we meant the number of the pairs of multiplication and addition needed to factorize the augmented system. In case of bpmpd, this was computed by the formulae

$$\sum_{i=1}^{n+m} \frac{c_i(c_i+1)}{2},$$

where c_i denotes the number of off–diagonal nonzeros in the ith factor of the augmented system. In case of folaug the number of "expected" operations were reported. We computed the average time per iterations by dividing the total solution time by the number of iterations. This provides with information about the "overall" speed of the factorizations, including the time loss by pivot reordering and overcoming numerical difficulties. Table 2 contains two columns corresponding to each comparison term, the columns referred to as bpmpd and folaug contain the results obtained by our and Mehrotra's implementation respectively. In some of the cases, our solver chose the normal equations approach. These are noted with an asterix (*) in the "Nonzeros" column of bpmpd.

Table 2: Comparison of BPMPD and FO1AUG factorizations

Problem Name	Ord. time bpmpd		Nonzeros bpmpd	Nonzeros fo1aug	Operations bpmpd	Operations fo1aug	Time/iteration bpmpd	Time/iteration fo1aug
aircraft	1.02		20317	20317	36904	36904	0.792	2.726
cep1	0.65		10133	13423	19803	25097	0.302	0.362
pgp2	1.15		29459	36304	61616	67424	0.879	1.412
sc205-2r-4	0.02	*	582	656	2222	1349	0.007	0.020
sc205-2r-8	0.04	*	1378	1236	9202	2576	0.020	0.031
sc205-2r-16	0.17		1469	2388	3364	4968	0.045	0.066
sc205-2r-27	0.10		2573	3972	5889	8257	0.074	0.112
sc205-2r-32	0.13		2861	4692	6548	9752	0.078	0.168
sc205-2r-50	0.22		4589	7284	10498	15134	0.128	0.213
sc205-2r-64	0.34		5645	9300	12916	19320	0.180	0.267
sc205-2r-100	0.50		9101	14484	20816	30084	0.286	0.383
sc205-2r-200	1.27		17961	28884	41076	59984	0.699	0.886
sc205-2r-400	1.54		43353	57684	129426	119784	1.428	1.902
sc205-2r-800	3.57		86633	115284	258826	239384	3.059	4.049
scagr7-2r-4	0.03		895	933	2486	2042	0.016	0.026
scagr7-2r-8	0.05		1605	1773	4000	3870	0.030	0.045
scagr7-2r-16	0.12		3123	3453	7793	7526	0.071	0.084
scagr7-2r-27	0.22		5209	5763	13004	12553	0.131	0.129
scagr7-2r-32	0.30		6161	6813	15386	14838	0.149	0.160
scagr7-2r-54	0.49		10327	11433	25787	24892	0.276	0.303
scagr7-2r-64	0.64		12233	13533	30558	29462	0.306	0.336
scagr7-2r-108	0.89		20155	22449	51814	48274	0.560	0.653
scagr7-2r-216	1.85		50401	44805	186013	96334	1.389	1.412
scagr7-2r-432	4.11		100729	89517	371989	192454	3.202	3.027
scagr7-2r-864	9.04		201385	178941	743941	384694	7.133	7.128
scagr7-2c-4	0.03		895	933	2486	2042	0.016	0.027
scagr7-2c-16	0.13		3123	3453	7793	7526	0.074	0.081
scagr7-2c-64	0.62		12233	13533	30558	29462	0.333	0.429
scagr7-2b-4	0.03		895	933	2486	2042	0.016	0.022
scagr7-2b-16	0.11		3123	3453	7793	7526	0.070	0.087
scagr7-2b-64	4.45		55327	53853	174086	117206	1.901	1.736
scfxm1-2r-4	0.26	*	9422	10984	78574	107629	0.122	0.192
scfxm1-2r-8	0.55	*	18403	21091	193378	235881	0.272	0.630
scfxm1-2r-16	1.16	*	37576	43632	573592	547611	0.568	0.884
scfxm1-2r-27	5.97		60572	74591	706703	1002629	1.327	2.541
scfxm1-2r-32	5.27		71242	87906	829118	1185214	1.444	3.272
scfxm1-2r-64	15.03		139530	186516	1612574	3054196	2.950	13.410
								ctd. →

Comparison of BPMPD and FO1AUG factorizations, ctd.								
Problem Name	Ord. time bpmpd		Nonzeros bpmpd	fo1aug	Operations bpmpd	fo1aug	Time/iteration bpmpd	fo1aug
scfxm1-2r-96	27.78		207818	278132	2396030	4553300	4.812	22.054
scfxm1-2r-128	45.95		276107	369748	3179495	6052404	6.851	49.163
scfxm1-2r-256	46.01		761187	736212	13503839	12048820	16.375	122.255
scfxm1-2c-4	0.27	*	9422	10984	78574	107629	0.130	0.195
scfxm1-2b-4	0.25	*	9422	10984	78574	107629	0.117	0.202
scfxm1-2b-16	1.21	*	37576	43632	573592	547611	0.586	0.851
scfxm1-2b-64	46.08		276107	369748	3179495	6052404	6.591	29.092
scrs8-2r-4	0.00	*	106	663	172	1283	0.003	0.020
scrs8-2r-8	0.02	*	568	1255	1923	2793	0.018	0.033
scrs8-2r-16	0.03		419	2451	663	6512	0.036	0.068
scrs8-2r-27	0.08		961	4002	1584	10417	0.054	0.120
scrs8-2r-32	0.06		835	4707	1319	12192	0.067	0.134
scrs8-2r-64	0.12		1667	9219	2631	23552	0.148	0.270
scrs8-2r-64b	0.15		2370	9219	4001	23552	0.150	0.250
scrs8-2r-128	0.36		4718	18243	7949	46272	0.365	0.532
scrs8-2r-256	0.70		11570	36291	24371	91712	0.760	1.132
scrs8-2r-512	1.58		24230	72387	51911	182592	1.727	2.425
scrs8-2c-4	0.00	*	106	663	172	1283	0.004	0.020
scrs8-2c-8	0.01	*	228	1255	424	2793	0.017	0.034
scrs8-2c-16	0.03		419	2451	663	6512	0.033	0.064
scrs8-2c-32	0.06		835	4707	1319	12192	0.063	0.129
scrs8-2c-64	0.11		1667	9219	2631	23552	0.126	0.249
scrs8-2b-4	0.01	*	106	663	172	1283	0.007	0.020
scrs8-2b-16	0.03		419	2451	663	6512	0.034	0.072
scrs8-2b-64	0.16		2774	9219	5029	23552	0.147	0.254
scsd8-2r-4	0.07	*	3291	3323	20575	20623	0.057	0.074
scsd8-2r-8	0.33		8341	8510	127103	126221	0.161	0.216
scsd8-2r-8b	0.42		8341	8510	127103	126221	0.174	0.230
scsd8-2r-16	0.79		16306	16434	270708	270900	0.410	0.432
scsd8-2r-27	1.45		25005	25237	373229	373577	0.543	0.581
scsd8-2r-32	1.78		29943	30199	481547	481931	0.743	0.904
scsd8-2r-54	3.29		43709	44435	443810	445151	1.118	1.166
scsd8-2r-64	4.20		51857	52375	521682	522711	1.550	1.580
scsd8-2r-108	9.64		85609	87311	861170	863975	2.395	2.133
scsd8-2r-216	9.90		226003	173063	3492374	1701623	5.269	4.294
scsd8-2r-432	23.23		514641	344567	9429218	3376919	11.717	9.906
								ctd. →

Comparison of BPMPD and FO1AUG factorizations, ctd.									
Problem	Ord. time		Nonzeros		Operations		Time/iteration	
Name	bpmpd		bpmpd	fo1aug	bpmpd	fo1aug	bpmpd	fo1aug
scsd8-2c-4	0.08	*	3295	3323	20581	20623	0.059	0.076
scsd8-2c-16	1.22		16314	16434	270720	270900	0.400	0.442
scsd8-2c-64	13.45		303889	204823	5595730	2011863	7.518	6.473
scsd8-2b-4	0.09	*	3291	3323	20575	20623	0.067	0.080
scsd8-2b-16	0.79		16066	16434	270348	270900	0.381	0.440
scsd8-2b-64	13.48		303761	204823	5595538	2011863	7.624	6.422
sctap1-2r-4	0.07	*	2638	2820	13277	11484	0.045	0.057
sctap1-2r-8	0.15		4221	5849	15070	39062	0.095	0.122
sctap1-2r-8b	0.19		4221	5849	15070	39062	0.107	0.123
sctap1-2r-16	0.30		8222	10371	28359	52679	0.189	0.221
sctap1-2r-27	0.53		13744	17026	47092	84194	0.326	0.381
sctap1-2r-32	0.67		16254	20051	55607	98519	0.402	0.446
sctap1-2r-54	1.40		27298	33361	93073	161549	0.714	0.872
sctap1-2r-64	1.78		32318	39411	110103	190199	0.970	0.927
sctap1-2r-108	3.06		54406	66031	185035	316259	1.617	1.913
sctap1-2r-216	5.82		111911	131371	393025	625679	3.466	3.545
sctap1-2r-480	11.89		276397	291091	1246648	1382039	8.042	8.202
sctap1-2c-4	0.06	*	2638	2820	13277	11484	0.035	0.061
sctap1-2c-16	0.29		8222	10371	28359	52679	0.171	0.224
sctap1-2c-64	1.44		28302	34571	96479	167279	0.736	0.836
sctap1-2b-4	0.06	*	2638	2820	13277	11484	0.037	0.056
sctap1-2b-16	0.34		8222	10371	28359	52679	0.211	0.215
sctap1-2b-64	5.93		149543	155571	677939	740279	4.105	4.567
fxm2-6	0.72	*	27569	31158	257074	322130	0.362	0.819
fxm2-16	2.06	*	75238	82429	1001852	889804	1.141	2.233
pltexpA2-6	0.49	*	15262	9313	221335	82953	0.224	0.239
pltexpA2-16	0.99		20741	21663	180579	183643	0.433	0.482
stormG2-8	9.85		113366	113016	3879415	3334826	2.419	2.983
stormG2-27	44.64		329760	320923	8192959	5751170	7.988	7.777
deter0	0.93		21353	21407	65561	66322	0.641	0.683
deter1	3.58		61373	61461	188168	189439	2.091	2.321
deter2	4.77		66089	66194	190656	192146	2.557	2.749
deter3	5.40		84913	85021	260288	261859	3.016	3.284
deter4	1.90		33419	33507	90256	91475	1.138	1.233
deter5	3.23		56665	56749	173744	174955	2.002	2.122
deter6	2.51		47249	47325	144896	145987	1.582	1.683
								ctd. →

Comparison of BPMPD and FO1AUG factorizations, ctd.							
Problem	Ord. time	Nonzeros		Operations		Time/iteration	
Name	bpmpd	bpmpd	fo1aug	bpmpd	fo1aug	bpmpd	fo1aug
deter7	4.88	70789	70885	217016	218407	2.647	2.687
deter8	2.28	42541	42613	130472	131503	1.392	1.492

* : bpmpd used the normal equations system

In many cases we obtained similar results by the two solvers. It is not very surprising due to the fact that both of them are based on similar features. It is to be noted that the similar behavior was true by the cases when bpmpd chose the normal equation approach also. It means that the heuristic [20] made correct decisions on all our test problems. In spite of the fact that the factorizations had similar densities, we observed large deviation on the iteration speed in certain cases. For example, on problem *sfcxm1–2r–256* bpmpd computes nearly 8 times faster one iteration than fo1aug. Its reason is that fo1aug encounter serious numerical difficulties which results in considerable time loss during the recomputation of the pivot order. We believe that it is the principal ground of all differences in this comparison term.

Table 3 shows the comparison of the overall performance of the two implementations. Figures given include the number of iterations to optimality, the total solution time and the primal objective function values. All timing results are given in seconds, and reflect on the pure algorithmic time, without the input of the MPS file and the time spent on produce an output. Both of the solvers were configured to stop when the relative duality gap, computed by formulae

$$\frac{c^T x - b^T y}{|c^T x| + 1.0}$$

was less than 10^{-8} and both the primal ad dual infeasibilities were below 10^{-6}.

Table 3: Comparison of BPMPD and FO1AUG performance

Problem	Iterations		Solution time		Primal objective	
Name	bpmpd	fo1aug	bpmpd	fo1aug	bpmpd	fo1aug
aircraft	12	11	9.51	29.99	1567.04235	1567.04235
cep1	16	18	4.83	6.51	355160.086	355160.088
pgp2	31	23	27.24	32.47	447.324315	447.324315
sc205-2r-4	7	8	0.05	0.16	-60.4229604	-60.4229607
sc205-2r-8	8	9	0.16	0.28	-60.4229606	-60.4229605
sc205-2r-16	10	11	0.45	0.73	-55.3877140	-55.3877140
sc205-2r-27	11	10	0.81	1.12	-15.1057402	-15.1057402
sc205-2r-32	11	11	0.86	1.85	-55.3877140	-55.3877140
sc205-2r-50	13	13	1.66	2.77	-30.7641141	-30.7641141
						ctd. →

Comparison of BPMPD and FO1AUG performance, ctd.						
Problem Name	Iterations bpmpd	Iterations fo1aug	Solution time bpmpd	Solution time fo1aug	Primal objective bpmpd	Primal objective fo1aug
sc205-2r-64	11	12	1.98	3.21	-55.3877140	-55.3877140
sc205-2r-100	14	11	4.00	4.21	-10.0704935	-10.0704935
sc205-2r-200	12	13	8.39	11.52	-10.0704935	-10.0704935
sc205-2r-400	13	14	18.57	26.63	-10.0704935	-10.0704935
sc205-2r-800	16	15	48.94	60.74	-10.0704935	-10.0704935
scagr7-2r-4	13	14	0.21	0.37	-832902.150	-832902.151
scagr7-2r-8	15	15	0.45	0.67	-832902.150	-832902.151
scagr7-2r-16	16	16	1.14	1.35	-832902.148	-832902.150
scagr7-2r-27	18	18	2.35	2.33	-833543.814	-833543.815
scagr7-2r-32	20	18	3.32	2.88	-832902.093	-832902.091
scagr7-2r-54	21	20	5.79	6.06	-833834.103	-833834.102
scagr7-2r-64	21	21	6.42	7.06	-832902.041	-832902.042
scagr7-2r-108	28	22	15.69	14.36	-834038.849	-834038.847
scagr7-2r-216	30	26	42.77	36.72	-834038.847	-834038.848
scagr7-2r-432	35	29	110.35	87.79	-834029.780	-834029.776
scagr7-2r-864	40	28	276.60	199.58	-834376.099	-834376.098
scagr7-2c-4	14	13	0.23	0.35	-832260.034	-832260.034
scagr7-2c-16	17	14	1.25	1.13	-832260.031	-832260.034
scagr7-2c-64	22	18	7.44	7.72	-818719.598	-818719.605
scagr7-2b-4	13	13	0.21	0.29	-832941.326	-832941.327
scagr7-2b-16	16	16	1.12	1.40	-832902.149	-832902.150
scagr7-2b-64	27	25	51.31	43.39	-832901.547	-832901.547
scfxm1-2r-4	20	20	2.44	3.83	2877.56386	2877.56386
scfxm1-2r-8	21	22	5.72	13.87	2877.56387	2877.56386
scfxm1-2r-16	24	26	13.62	22.98	2877.56386	2877.56386
scfxm1-2r-27	28	28	37.17	71.14	2886.96461	2886.96461
scfxm1-2r-32	28	26	40.42	85.06	2877.56387	2877.56386
scfxm1-2r-64	39	34	115.05	455.95	2877.56387	2877.56386
scfxm1-2r-96	40	34	192.48	749.85	2891.70611	2891.70611
scfxm1-2r-128	39	33	267.17	1622.39	2891.70612	2891.70611
scfxm1-2r-256	45	*15	736.86	*1833.83	2891.70612	*3976.95358
scfxm1-2c-4	20	19	2.60	3.70	2875.98337	2875.98336
scfxm1-2b-4	19	20	2.23	4.05	2875.98336	2875.98336
scfxm1-2b-16	25	26	14.65	22.13	2877.56387	2877.56387
scfxm1-2b-64	41	37	270.23	1076.39	2877.56386	2877.56386
scrs8-2r-4	9	8	0.03	0.16	123.176565	123.176565
						ctd. →

Comparison of BPMPD and FO1AUG performance, ctd.						
Problem	Iterations		Solution time		Primal objective	
Name	bpmpd	fo1aug	bpmpd	fo1aug	bpmpd	fo1aug
scrs8-2r-8	9	11	0.16	0.36	1122.97085	1122.97084
scrs8-2r-16	9	9	0.32	0.61	123.176565	123.176565
scrs8-2r-27	11	15	0.59	1.80	587.166361	587.166361
scrs8-2r-32	9	9	0.60	1.21	123.176565	123.176565
scrs8-2r-64	9	12	1.33	3.24	123.176603	123.176603
scrs8-2r-64b	12	14	1.80	3.50	1346.09633	1346.09633
scrs8-2r-128	12	17	4.38	9.05	1177.34504	1177.34504
scrs8-2r-256	14	16	10.64	18.11	1144.16062	1144.16062
scrs8-2r-512	14	16	24.18	38.80	1074.09934	1074.09934
scrs8-2c-4	10	8	0.04	0.16	112.104275	112.104275
scrs8-2c-8	10	9	0.17	0.31	112.415909	112.415909
scrs8-2c-16	10	10	0.33	0.64	112.431726	112.431726
scrs8-2c-32	11	11	0.69	1.42	112.156236	112.156236
scrs8-2c-64	12	15	1.51	3.73	111.881190	111.881190
scrs8-2b-4	9	8	0.06	0.16	112.104275	112.104275
scrs8-2b-16	9	9	0.31	0.65	112.104276	112.104276
scrs8-2b-64	11	14	1.62	3.55	1121.45123	1121.45123
scsd8-2r-4	7	5	0.40	0.37	15.5000000	15.5000000
scsd8-2r-8	7	5	1.13	1.08	16.0000000	16.0000000
scsd8-2r-8b	7	5	1.22	1.15	16.0000000	16.0000000
scsd8-2r-16	8	5	3.28	2.16	15.9995000	15.9995000
scsd8-2r-27	12	11	6.52	6.39	24.0000002	24.0000001
scsd8-2r-32	8	5	5.94	4.52	15.9990000	15.9990000
scsd8-2r-54	13	12	14.53	13.99	23.8499979	23.8499979
scsd8-2r-64	8	6	12.40	9.48	15.8427501	15.8427500
scsd8-2r-108	14	14	33.53	29.86	23.6999943	23.6999944
scsd8-2r-216	15	14	79.03	60.11	23.3999839	23.3999839
scsd8-2r-432	15	16	175.75	158.50	25.7999695	25.7999695
scsd8-2c-4	7	5	0.41	0.38	15.0000000	15.0000000
scsd8-2c-16	8	5	3.20	2.21	15.0000000	15.0000000
scsd8-2c-64	9	6	67.66	38.84	14.9992320	14.9992320
scsd8-2b-4	7	5	0.47	0.40	15.2500000	15.2500000
scsd8-2b-16	7	5	2.67	2.20	27.5000000	27.5000000
scsd8-2b-64	9	6	68.62	38.53	15.9986680	15.9986680
sctap1-2r-4	10	10	0.45	0.57	280.500000	280.500000
sctap1-2r-8	11	12	1.04	1.46	360.500000	360.500000
						ctd. →

Comparison of BPMPD and FO1AUG performance, ctd.						
Problem Name	Iterations bpmpd	Iterations fo1aug	Solution time bpmpd	Solution time fo1aug	Primal objective bpmpd	Primal objective fo1aug
sctap1-2r-8b	11	12	1.18	1.48	250.000000	250.000000
sctap1-2r-16	12	13	2.27	2.87	359.000000	359.000000
sctap1-2r-27	13	16	4.24	6.09	247.500000	247.500000
sctap1-2r-32	12	14	4.82	6.25	354.000000	354.000001
sctap1-2r-54	15	17	10.71	14.83	249.250000	249.250000
sctap1-2r-64	14	16	13.58	14.83	344.000000	344.000000
sctap1-2r-108	16	19	25.87	36.35	248.000000	248.000000
sctap1-2r-216	17	20	58.92	70.91	249.000000	249.000000
sctap1-2r-480	19	23	152.80	188.64	248.500000	248.500000
sctap1-2c-4	11	12	0.38	0.73	236.250000	236.250000
sctap1-2c-16	14	16	2.39	3.58	326.400000	326.400000
sctap1-2c-64	16	17	11.78	14.21	200.390625	200.390626
sctap1-2b-4	12	12	0.44	0.67	239.250000	239.250000
sctap1-2b-16	13	15	2.74	3.23	280.800001	280.800000
sctap1-2b-64	19	22	77.99	100.48	484.560000	484.560000
fxm2-6	20	21	7.24	17.19	18417.0656	18417.0656
fxm2-16	24	26	27.39	58.05	18416.7590	18416.7590
pltexpA2-6	15	14	3.36	3.35	-9.47935368	-9.47935367
pltexpA2-16	17	16	7.36	7.71	-9.66330824	-9.66330825
stormG2-8	39	50	94.33	149.13	15535236.6	15535236.6
stormG2-27	59	65	471.28	505.48	15508983.2	15508983.2
deter0	17	22	10.90	15.03	-2.04591961	-2.04591961
deter1	23	26	48.09	60.34	-2.55756435	-2.55756435
deter2	19	23	48.58	63.23	-1.82986994	-1.82986993
deter3	20	25	60.33	82.11	-2.10779072	-2.10779072
deter4	18	23	20.48	28.36	-1.42803505	-1.42803505
deter5	21	25	42.05	53.06	-2.26122357	-2.26122358
deter6	20	24	31.63	40.38	-2.30798670	-2.30798670
deter7	22	25	58.23	67.18	-2.22856797	-2.22856797
deter8	22	26	30.62	38.78	-2.79429546	-2.79429546
Total	1882	*1863	4213.73	*8917.46		

* On problem scfxm1-2r-256 FO1AUG ran out of memory after 15 iterations

In all cases, except from *scfxm1-2r-256* on which fo1aug ran out of memory after 15 iterations, the final objective values obtained in the two implementations match in at least 8 significant digits. We believe that it demonstrates the numerically robustness of the two solvers, and confirms the correct optimal values of the test problems. The difference in the number of iterations seems insignificant as well, taken on the whole it is about 1%. However, the solution times are much different. As noted previously, its reason is that bpmpd more successfully avoid the numerical difficulties than fo1aug. Our experiences confirm the competence of special techniques and the way of the use the augmented factorization described in Section 4.

7 Conclusions

In the paper we examined the use of the augmented system factorization in interior point methods to solve two–stage stochastic linear programming problems. It has attractive properties, e.g., the possibility to reduce the computational cost of the factorizations without supposing full row rank for any part of the constraint matrix. On the other hand, its indefinite property can cause that the pivot agenda computed at the beginning is not valid during all the iterations of the interior point algorithm. We proved that the pivot sequence obtained by our approach is theoretically stable. We pointed out one important issue on the use of the augmented factorization, namely to avoid numerical difficulties. We suggested some special extensions of the implementation which make the factorization more robust in context of interior point methods. We concluded that the improvement of centrality can decisively influence the efficiency of the augmented factorization. The comparison of our solver with `folaug` unequivocally confirms the need of such techniques.

We concluded also that the analyzer which examines the structure of the matrix and decides about the use of either the augmented factorization or the standard Cholesky factorization plays an important role. It is the case especially in the multi–stage stochastic programming problems where the structure of the constraint matrix may more often vary depending on the number of stages and realizations. It means that the normal equations approach can more often be advantageous in these problems. Therefore, detecting such cases in a preparation phase is very important.

Acknowledgments

The author would like to express his sincere thanks to Joseph Czyzyk without whose help this paper could never have been written.

References

[1] Andersen, K.D. (1995) A modified Schur complement method for handling dense columns in interior point methods for linear programming, *Technical Report,* Odense University.

[2] Arioli, M., Duff, I.S. and Rijk, P.P.M. (1989) On the Augmented System Approach to Sparse Least-Squares Problems, *Numerische Mathematik* 55, p. 667–684.

[3] Birge, J.R. (1995) Current Trends in Stochastic Programming Computation and Application, *Technical Report,* Department of Industrial and Operations Engineering, University of Michigan

[4] Birge, J.R., Dempster, M.A.H., Gassman, H.I., Gunn, E.A., King, A.J. and Wallace, S.W. (1987) A standard input format for multiperiod stochastic linear programs, *COAL newsletter*, 17, p. 1-20.

[5] Birge, J. and Holmes, D. (1992) Efficient solution of two–stage stochastic linear programs using interior point methods, *Computational Optimization and Applications* 1, p. 245–276.

[6] Birge, J. and Qi, L. (1988) Computing block–angular Karmarkar projections with applications to stochastic programming, *Management Science*, 34:12, p. 1472–1479.

[7] Bunch, J.R. and Parlett, B.N. (1971) Direct methods for solving symmetric indefinite systems of linear equations, *SIAM Journal on Numerical Analysys*, 8 p. 639–655.

[8] Czyzyk, J., Fourer, R. and Mehrotra, S. (1994) A study of the augmented system and column–splitting approaches for solving two–stage stochastic linear programs by interior point methods, *Technical Report* 93–05, Department of Industrial Engineering and Management Sciences, Northwestern University, Illinois

[9] Dantzig, G.B. and Madansky, A. (1961) On the solution of the two–stage linear programs under uncertainty, in: Neyman, J. ed., *Proceedings of the Fourth Berkely Symposium on Mathematical Statistics and Probability,* University of California Press, Berkeley, p. 165–176.

[10] Duff, I.S., Erisman, A.M. and Reid J.K. (1989) *Direct Methods for Sparse Matrices*, Oxford University Press, New York.

[11] Fourer, R., Mehrotra, S. (1993) Solving symmetric indefinite systems in an interior point method for linear programming, *Mathematical Programming*, 62, p. 15–39.

[12] Gondzio, J. (1994) Preconditioned Conjugate Gradients in an Interior Point Method for Two–stage Stochastic Programming, *Working Paper* WP–94–130, IIASA International Institute for Applied Systems Analysis, Laxemburg, Austria

[13] Gondzio, J. (1994) Multiple centrality corrections in primal dual method for linear programming, *Technical Report 1994.20*, Department of Management Studies, University of Geneva, Switzerland

[14] Holmes, D. (1994) A Collection of Stochastic Programming Problems, *Technical Report 94-11,* Deptartment of Industrial and Operations Engineering, The University of Michigan

[15] Kall P. and Mayer J. (1992) SLP–IOR: A model management system for stochastic linear programming – system design, in: Beulens, A.J.M. and Sebastian, J–H. (eds.) *Optimization–Based Computer–Aided Modeling and Design*, Springer p. 139–147.

[16] Kall, P. and Mayer J. (1992) A modell management system for stochastic linear programming, in: Kall, P. (ed.) *System Modelling and Optimization*, Springer p. 580–587.

[17] Karmarkar, N. (1984) A New Polynomial Time Algorithm for Linear Programming, Combinatorica 4, p. 375–395.

[18] Lustig, I.J., Mulvey, J.M. Carpenter, T.J. (1991) Formulating two–stage stochastic programs for interior point methods, *Operations Research* Vol.39, No.5, p. 757–770.

[19] Lustig, I. J., Marsten, R. E. and Shanno, D. .F. (1994) Interior Point Methods for Linear Programming: Computational State of the Art, *ORSA Journal on Computing* 6, p. 1–14.

[20] Maros, I. and Mészáros, Cs. (1995) The role of the augmented system in interior point methods, *Technical Report TR/06/95*, Brunel University, Department of Mathematics and Statistics, London.

[21] Mehrotra, S. (1991) Higher Order Methods and their Performance, *Technical Report 90–16R1,* Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, USA.

[22] Mészáros, Cs. (1994) Fast Cholesky–factorization for interior point method of linear programming, *Working Paper* WP 95–7, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest.

[23] Mészáros, Cs. (1995) The "inexact" minimum local fill–in ordering algorithm, *Working Paper* WP 95–7, Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest.

[24] Mulvey, J.M. and Vladimirou H. (1989) Stochastic network optimization models for investment planing *Ann. Operations Researches* 20, p. 187–217.

[25] Van Slyke, R. and Wets, R.J.-B. (1969) The L-shaped linear programs with application to optimal control and stochastic programming, *SIAM Journal on Applied Mathematics* 17, p. 638–663.

[26] Stewart, G.W. (1974) Modifying pivot elements in Gaussian elimination, *Math. Comp.*, 28, p. 537–542.

[27] Strazicky, B. (1980) Some results concerning an algorithm for the discrete recourse problem, in: Dempster, M.A.H. ed., *Stochastic Programming,* Academic Press, New York.

[28] Suhl, U.H. and Suhl, L.M. (1990) Computing Sparse LU Factorizations for Large–Scale Linear Programming Bases, *ORSA Journal on Computing* Vol. 2, No. 4, p. 325–335.

[29] Vanderbei, R.J., Carpenter, T.J. (1993) Symmetric indefinite systems for interior point methods *Mathematical Programming*, 58, p. 1–32

[30] Wets, R.J-B. (1974) Stochastic programs with fixed recourse: the equvivalent deterministic problem, *SIAM Review* 16, p 309–339.