A Computational View of Interior–Point Methods for Linear Programming[1]

Jacek Gondzio[2] Tamas Terlaky[3]

Logilab, HEC Geneva, Section of Management Studies, University of Geneva,

102 Bd Carl Vogt, CH-1211 Geneva 4, Switzerland

e–mail: gondzio@divsun.unige.ch

Faculty of Technical Mathematics and Informatics, Delft University of Technology,

P.O. Box 5031, 2600 GA Delft, The Netherlands.

e–mail: t.terlaky@twi.tudelft.nl

Technical Report 1994.22
June 25, 1994

[1]The research of the first author has been partially supported by the Committe for Scientific Research of Poland, grant No PB 8 S505 015 05 and by the Fonds National de la Recherche Scientifique Suisse, grant No 12-34002.92. The research of the second author has been partially supported by OTKA No 2116.

[2]on leave from the Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland

[3]on leave from the Eötvös University, Budapest, Hungary.

Abstract

Many issues that are crucial for an efficient implementation of an interior point algorithm are addressed in this paper. To start with, a prototype primal–dual algorithm is presented. Next, many tricks that make it so efficient in practice are discussed in detail. Those include: the preprocessing techniques, the initialization approaches, the methods of computing search directions (and lying behind them linear algebra techniques), centering strategies and methods of stepsize selection.

Several reasons for the manifestations of numerical difficulties like e.g.: the primal degeneracy of optimal solutions or the lack of feasible solutions are explained in a comprehensive way.

A motivation for obtaining an optimal basis is given and a practicable algorithm to perform this task is presented. Advantages of different methods to perform postoptimal analysis (applicable to interior point optimal solutions) are discussed.

Important questions that still remain open in the implementations of interior point methods are also addressed, e.g.: performing correct postoptimal analysis, detecting infeasibility or resolving difficulties arising in a presence of unbounded optimal faces. Challenging practical problem of warm start is recalled and two potentially attractive approaches to it are suggested.

To facilitate the understanding of different implementation strategies, some illustrative numerical results on a subset of problems from the Netlib collection are presented.

Key Words: Linear programming, interior point methods, primal-dual algorithm, implementation, numerical linear algebra.

Contents

1 Introduction

Karmarkar's publication of 1984 of the new polynomial-time algorithm for linear programming (LP) [41] drew an enormous attention of the mathematical programming community and led to its great activity during the past ten years resulting in a flood of papers (see, e.g., a bibliography [46]).

The idea of crossing the interior of the feasible region in search for an optimum of the linear program was present at least since the sixties. These were for example: an affine–scaling method of Dikin [17] and a logarithmic barrier method SUMT of Fiacco and McCormick [21]. For at least two reasons, however, these methods could not at the time be shown competitive to the Simplex. First, due to the storage limitations, the size of the problems solved in the late sixties did never exceed a couple of hundred rows and columns and for such sizes the simplex method is practically unbeatable. Secondly, there were no sparse symmetric solvers available at that time (they appeared at the beginning of seventies) so the orthogonal projections must have killed the efficiency of interior point methods (IPMs). IPMs need significantly more memory than the simplex method which was an unacceptable requirement that time.

Clearly, the situation was quite different in 1984, which encouraged Karmarkar to claim about the excellent efficiency of his new approach. In fact, these claims still had to wait a couple of years to be confirmed by the computational results [60] and [1, 2].

Soon after Karmarkar's publication, Gill, Murray, Saunders, Tomlin and Wright [26] built the bridge between this new interior point method and the logarithmic barrier approach. Barrier methods were developed for the primal and for the dual LP formulation (see, e.g., the surveys [31, 68]). Early implementations that were based on pure primal or dual methods gave already competitive results with simplex implementations. Nowadays all the state of the art IPM implementations are those of primal–dual methods, hence in this paper we concentrate only on primal–dual methods.

First Megiddo [54] proposed applying a logarithmic barrier method to the primal and the dual problems at the same time. Independently, Kojima, Mizuno and Yoshise [43] developed the theoretical background of this method and gave complexity results. Its early implementations [52, 15] showed very much promise and encouraged further research in this field. For extensions that represent current state-of-the-art primal–dual implementations see Lustig, Marsten and Shanno [48, 49, 50] and Mehrotra [55, 56].

A primal–dual algorithm is a feasible IPM if all the iterates are primal and dual feasible, respectively. If the iterates are positive but infeasible then the primal–dual algorithm is called an *infeasible IPM*. This algorithm attains feasibility at the same time as optimality is reached. It had been successfully implemented that way [48] and had shown very good practical convergence long before a theoretical justification for such a behavior was found by Kojima, Megiddo and Mizuno [44]. The method has proven polynomial complexity: $\mathcal{O}(n^2L)$ in [80] and $\mathcal{O}(nL)$ in [58, 63].

Although the complexity of the infeasible primal–dual algorithm is worse than the best known complexity $\mathcal{O}(\sqrt{n}L)$ of most feasible IPMs (see, e.g., the surveys [31, 66]), it is now widely accepted that primal–dual infeasible IPMs are more efficient in implementations. Since infeasible IPMs are the methods of choice to date for "state of the art" implementations, throughout the whole paper we mean a *primal–dual infeasible IPM* as we speak about a primal–dual algorithm. To facilitate this, in Section 2 we shall then introduce a prototype primal–dual infeasible IPM algorithm.

A common feature of almost all IPMs is that they can be interpreted in terms of following the path of centers [69] that leads to the optimal solution (see, e.g., [31] and [66] for the up to date references). With some abuse of mathematics, a basic iteration of a path–following algorithm consists of moving from one point in a neighborhood of the central path to another one called *target* that preserves the property of lying in a neighborhood of central path and reduces the distance to optimality measured with some estimation of the duality gap. Such a movement can in principle involve more than one step towards the target. Depending on how significant is the update of the target (and, consequently, whether just one or more Newton steps are needed to reach the vicinity of the new target) one distinguishes between short and long step methods. Due to the considerable cost of every Newton step, usually, (at least in implementations) one Newton step is allowed before a new target is defined.

Every Newton step requires computing at least one orthogonal projection onto the null space of a scaled linear operator AD, where A is the LP constraint matrix and D is a positive diagonal scaling matrix that changes in subsequent iterations. Primal, dual and primal–dual variants of IPMs differ on the way matrix D is defined but the effort to compute Karmarkar's projection is always the same. Every orthogonal projection involves inversion of the matrix AD^2A^T – the most time–consuming linear algebra operation that takes about 60–90% of the computation time of a single interior point iteration. Unless the linear program is specially structured and this structure can be exploited to determine an easily invertible preconditioner for an iterative method, as e.g., conjugate gradients algorithm (implemented successfully for network problems [65, 62]), direct methods [18] that compute sparse symmetric factorization (Cholesky decomposition of the positive definite system AD^2A^T or Bunch–Parlett [13, 7] decomposition of the indefinite augmented system $\begin{bmatrix} D^{-2} & A^T \\ A & 0 \end{bmatrix}$) are the methods of choice. Computing projections onto affine spaces seems crucial for the efficiency of any interior point algorithm. We shall thus discuss it in detail in Section 3 that addresses also other issues of implementation of the IPM as e.g.: the role of presolve analysis, the choice of the starting point, the choice of the stepsizes in the primal and in the dual spaces, the role of centering, higher order methods, the termination conditions and, finally, the comparison of theoretical and practical complexity.

In Section 4 we shall add some remarks on the manifestations of the degeneracy and the ill–conditioning in the computations of projections.

After about forty years of the application of the simplex method (starting from its discovery in 1947 [16] till Karmarkar's breakthrough [41]), when it was beyond any competition, the operations research practitioners got used to seeing linear programming from the simplex perspective. This, in particular, applies to the use of postoptimality analysis available from the optimal basis solution. In fact, such a postoptimality analysis is almost always incomplete (frequently incorrect), see e.g., [32, 37, 40]. Nevertheless, there exist many applications in which optimal basis is necessary, e.g., reoptimization in integer programming. In such a case a need arises of identifying optimal basis from the interior point optimal solution. Fortunately, this can be done in a strongly polynomial time [53]. We shall address the problem of optimal basis identification in Section 5.

Section 6 will be devoted to some crucial questions that still remain open. Sensitivity analysis based on interior point optimal solution is generally more expensive but produces correct information. We discuss how to handle problems with unbounded level sets, how to detect infeasibility and how to implement efficient warm start in interior point algorithms.

Most relevant issues of interior point method implementations will be illustrated by solving a subset of the *Netlib* LP problem test collection using version 2.0 of the HOPDM (Higher Order

Primal Dual Method) code [4, 30]. All of our computations are made on a SUN SPARC–10 workstation.

Later on in the paper we will frequently speak about stability, robustness and efficiency of different methods. On stability, the usual numerical stability is meant. Talking about robustness, one thinks about that the algorithm gives reliable answer on a wide range (optimally all) of problem instances. Finally, efficiency relates to the speed of the algorithm, the speed of the implementation.

2 A prototype primal–dual algorithm

Let us consider a primal linear programming problem

$$\begin{array}{rr} \text{minimize} & c^T x, \\ \text{subject to} & Ax = b, \\ & x + s = u, \\ & x, s \geq 0, \end{array} \tag{1}$$

where $c, x, s, u \in \mathcal{R}^n, b \in \mathcal{R}^m, A \in \mathcal{R}^{m*n}$ and its dual

$$\begin{array}{rr} \text{maximize} & b^T y - u^T t, \\ \text{subject to} & A^T y - t + z = c, \\ & z, t \geq 0, \end{array} \tag{2}$$

where $y \in \mathcal{R}^m$ and $z, t \in \mathcal{R}^n$.

To derive the primal-dual algorithm let us replace nonnegativity of constraints in the primal formulation with the logarithmic barrier penalty terms in the objective function, which gives the following logarithmic barrier function

$$L(x, s, \mu) = c^T x - \mu \sum_{j=1}^{n} \ln x_j - \mu \sum_{j=1}^{n} \ln s_j. \tag{3}$$

The first order optimality conditions for (3) are

$$\begin{array}{rcl} Ax & = & b, \\ x + s & = & u, \\ A^T y + \mu X^{-1} e - t & = & c, \\ \mu S^{-1} e - t & = & 0. \end{array} \tag{4}$$

Substituting

$$z = \mu X^{-1} e,$$

the first order optimality conditions (4) give

$$\begin{array}{rcl} Ax & = & b, \\ x + s & = & u, \\ A^T y + z - t & = & c, \\ XZe & = & \mu e, \\ STe & = & \mu e, \end{array} \tag{5}$$

where X, S, Z and T are diagonal matrices with the elements x_j, s_j, z_j and t_j, respectively, e is the n-vector of all ones and μ is a barrier parameter.

The set of solutions of (5) $(x(\mu), s(\mu))$ and $(y(\mu), z(\mu), t(\mu))$ defines the central path of the primal and dual problem, respectively. Having any primal and dual feasible solutions (x, s) and (y, z, t) the quality of centrality is measured by

$$\delta^2(x, s, y, z, t, \mu) = \sum_{j=1}^{n} \left(\sqrt{\frac{x_j z_j}{\mu}} - \sqrt{\frac{\mu}{x_j z_j}} \right)^2 + \sum_{j=1}^{n} \left(\sqrt{\frac{s_j t_j}{\mu}} - \sqrt{\frac{\mu}{s_j t_j}} \right)^2 \tag{6}$$

Note that (x, s) and (y, z, t) are on the *central path* iff $\delta(x, s, y, z, t, \mu) = 0$. The smaller δ is the better the points are centered.

Let us observe that the first three of the above equations are linear and force primal and dual feasibility of the solution. The last two equations are nonlinear and become the complementarity conditions for $\mu = 0$, which together with the feasibility constraints provides optimality of the solutions. If some solution with a certain μ is available, then for the complementarity gap one has

$$x^T z + s^T t = 2\mu e^T e = 2n\mu.$$

In any IPM, this quantity measures the error in complementarity. Observe that in feasible IPMs the complementarity gap reduces to the usual duality gap. It, clearly, vanishes at an optimal solution.

A single iteration of the basic primal–dual algorithm makes one step of Newton's method applied to the first order optimality conditions (5) with a given μ and then μ is updated (usually decreased). The algorithm terminates when infeasibility and the complementarity gap is reduced below a predetermined tolerance.

Having an $x, s, z, t \in \mathcal{R}^n_+$, $y \in \mathcal{R}^m$, Newton's direction is obtained by solving the following system of linear equations

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & T & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \\ \Delta z \\ \Delta t \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_u \\ \xi_c \\ \mu e - XZe \\ \mu e - STe \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} \xi_b &= b - Ax, \\ \xi_u &= u - x - s, \\ \text{and} \qquad \xi_c &= c - A^T y - z + t, \end{aligned}$$

denote the violations of the primal and the dual constraints, respectively. Primal–dual infeasible IPMs do not require the feasibility of the solutions (ξ_b, ξ_u and ξ_c might be nonzero) during the optimization process. Feasibility is attained during the process as optimality is reached. It is easy to verify that if a step of length one is made in the Newton's direction (7), then feasibility is reached at once. This is seldom the case as a smaller stepsize has usually to be chosen (a damped

Newton iteration is taken) to preserve nonnegativity of x, s, z and t. If this is the case and a stepsize $\alpha < 1$ is applied, then infeasibilities ξ_b, ξ_u and ξ_c are reduced $1-\alpha$ times.

Let us now look closer to the system of linear equations (7). After elimination

$$\begin{aligned} \Delta z &= X^{-1}(\mu e - XZe - Z\Delta x), \\ \Delta s &= \xi_u - \Delta x, \\ \Delta t &= S^{-1}(\mu e - STe - T\Delta s) = S^{-1}(\mu e - STe - T\xi_u + T\Delta x), \end{aligned} \tag{8}$$

it reduces to

$$\begin{bmatrix} -D^{-2} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} r \\ h \end{bmatrix}, \tag{9}$$

where

$$\begin{aligned} D^2 &= (X^{-1}Z + S^{-1}T)^{-1}, \\ r &= \xi_c - X^{-1}(\mu e - XZe) + S^{-1}(\mu e - STe) - S^{-1}T\xi_u, \\ h &= \xi_b. \end{aligned} \tag{10}$$

The matrix in the *augmented system* (9) is sparse, symmetric but indefinite. This system of linear equations can be solved directly by the Bunch–Parlett factorization [13, 7] or, after multiplying the first equation by AD^2 and substituting from the second equation,the system can be reduced to sparse, symmetric and positive definite *normal equations system*

$$(AD^2A^T)\Delta y = AD^2 r + h. \tag{11}$$

Advantages of both approaches will be discussed in the next section. Here, we only conclude that once Δy is found from (11), it uniquely determines

$$\Delta x = D^2 A^T \Delta y - D^2 r,$$

and $\Delta s, \Delta z$ and Δt by (8).

Computing $(\Delta x, \Delta y)$ from (9) or Δy from (11) is usually (when a direct approach [18] is applied) divided into two phases: *factorization* of the matrix to some easily invertible form and the following *solve* that exploits this factorization. Usually, the second step is at least an order of magnitude cheaper than the first one. This observation led [42, 56] to introducing higher order terms when computing direction. Computing different corrector terms resolves to multiple solution of the same linear system for several right hand sides (it thus reuses the same factorization and is relatively inexpensive). The most successful technique that incorporates higher order information into the primal–dual algorithm comes from Mehrotra [55, 56]. We shall address this technique in more detail in the next section.

For the sake of brief presentation of the method, we assume that some direction $(\Delta x, \Delta y, \Delta s, \Delta z, \Delta t)$ is computed, the maximum stepsizes in it for primal, α_P and dual, α_D spaces that maintain non-negativity of variables are found and, after being slightly reduced with a factor α_0, a new iterate is computed

$$
\begin{aligned}
x^{k+1} &= x^k + \alpha_0 \alpha_P \Delta x, \\
s^{k+1} &= s^k + \alpha_0 \alpha_P \Delta s, \\
y^{k+1} &= y^k + \alpha_0 \alpha_D \Delta y, \\
z^{k+1} &= z^k + \alpha_0 \alpha_D \Delta z, \\
t^{k+1} &= t^k + \alpha_0 \alpha_D \Delta t.
\end{aligned} \tag{12}
$$

After making the step, the barrier parameter μ is reduced and the process is repeated. Formally, the algorithm can be summarized as follows.

Prototype Algorithm

Input
    (x^0, s^0) and (y^0, z^0, t^0): the initial pair of primal and dual solutions, respectively;

Parameters
    ε is the accuracy parameter;
    α_0 is the step size;

begin
    $x := x^{(0)}$; $s := s^{(0)}$; $y := y^{(0)}$; $z := z^{(0)}$; $t := t^{(0)}$;
    while stopping criteria is not satisfied **do**
        calculate the search directions by (8,9,10);
        calculate the new iterates by (12);
    end
end.

---

3 Some tricks that make it work

In this section we shall concentrate on several issues that seem to be crucial for an efficient implementation of interior point algorithms.

3.1 Presolve

Linear programs solved nowadays have often very large sizes and are usually generated automatically by some modelling support tools. They are thus often formulated in a way that is not necessarily the most suitable for a direct application of an LP solver. Hence it is advantageous to analyse and, if possible, simplify their formulation before passing them to a solver. The important role of presolving in linear programming was recognized [12] long time ago. It is worth to mention that presolve is strongly recommended for any solver independently on the method used.

The importance of presolve is even more pronounced when an interior point based solver is applied [2, 30] due to more involved linear algebra operations and the need of full rank of the matrix A. Preprocessing aims at three main goals, namely: problem size reduction, problem density reduction and ensuring full rank of the constraint matrix A.

Problem size reduction

A general purpose presolve of [30] repeats the logical analysis of the LP problem formulation until no further reduction is obtained (every single reduction creates possibility for further model simplifications). The following simple operations are applied.

1. Empty rows and columns are removed.

2. Singleton inequality constraints are replaced by bounds on the variables (variable with an entry in a singleton equality row is fixed and removed).

3. Lower and upper limits for every constraint i are determined

$$\underline{b_i} = \sum_{\{j: a_{ij} < 0\}} a_{ij} u_j, \qquad \text{and} \qquad \overline{b_i} = \sum_{\{j: a_{ij} > 0\}} a_{ij} u_j, \tag{13}$$

   that clearly satisfy

$$\underline{b_i} \leq \sum_j a_{ij} x_j \leq \overline{b_i}. \tag{14}$$

   Observe that due to the nonnegativity of x, the limits $\underline{b_i}$ and $\overline{b_i}$ are nonpositive and nonnegative, respectively. If the inequalities (14) are at least as tight as the original (inequality type) LP constraint, then the constraint i is *redundant*. If one of them contradicts the LP constraint, then the problem is infeasible. Finally, in some special cases (e.g.: "less than or equal to" row with $\underline{b_i} = b_i$, "greater than or equal to" row with $\overline{b_i} = b_i$, or equality type row for which b_i equals to one of the limits $\underline{b_i}$ or $\overline{b_i}$), the LP constraint becomes a *forcing* one. This means that the only way to satisfy the constraint is to fix all variables that appear in it on their appropriate bounds.

4. Constraint limits (13) are used to generate implied variable bounds. (Note, that LP variables were transformed to the standard form $0 \leq x \leq u$, before). This technique makes use of the original form of an LP constraint (i.e., its form before a slack variable has been added to it to transform it to the "standard" equality row of (1)). Assume, for example, that a nonredundant "less than or equal to" (LE) type constraint is given, i.e.,

$$\underline{b_i} < \sum_j a_{ij} x_j \leq b_i.$$

   Then

$$\begin{array}{llll} & \forall k: & a_{ik} > 0 & \underline{b_i} + a_{ik} x_k \leq \sum_j a_{ij} x_j \leq b_i, \\ \text{and} & \forall k: & a_{ik} < 0 & \underline{b_i} + a_{ik}(x_k - u_k) \leq \sum_j a_{ij} x_j \leq b_i, \end{array}$$

and a new implied bounds are given for all variables involved by row i

$$\begin{aligned} x_k \leq u_k' &= (b_i - \underline{b_i})/a_{ik} && \text{for all} \quad k: \quad a_{ik} > 0, \\ x_k \geq l_k' &= u_k + (b_i - \underline{b_i})/a_{ik} && \text{for all} \quad k: \quad a_{ik} < 0. \end{aligned}$$

If they are tighter than the original ones, then variable bounds are improved. Note, that a technique similar to that is decribed above is particularly useful when it imposes finite bounds on free variables. Free variables do not, in such a case, have to be split and represented as the difference of two nonnegative variables.

5. For every singleton column, a row with an entry in it is used to generate implied bounds on a variable referring to it. If these bounds are at least as tight as the original ones, then the variable becomes an implied free. Consequently, both the row (implied free constraint) and the singleton free column is eliminated.

6. Nonnegative unbounded variables ($0 \leq x \leq +\infty$) refering to singleton columns are used to generate bounds on dual variables y. Namely, if the variable j refers to a singleton column with an entry a_{ij} and $u_j = +\infty$ (i.e., $t_j = 0$), then dual constraint (2) becomes an inequality

$$a_{ij} y_i \leq c_j.$$

This inequality can be solved and, depending on the sign of a_{ij}, produces a lower or upper bound on y_i.

7. Once all dual variables have explicit (possibly infinite) bounds

$$p_i \leq y_i \leq q_i, \qquad i = 1, 2, \cdots, m, \tag{15}$$

lower and upper limits for every dual constraint j are generated

$$\underline{c_j} = \sum_{\{i: a_{ij} < 0\}} a_{ij} q_i + \sum_{\{i: a_{ij} > 0\}} a_{ij} p_i, \quad \text{and} \quad \overline{c_j} = \sum_{\{i: a_{ij} < 0\}} a_{ij} p_i + \sum_{\{i: a_{ij} > 0\}} a_{ij} q_i. \tag{16}$$

These limits are compared with the cost coefficient c_j and applied to identify variables for which the sign of the reduced cost $z_j - t_j$ can be restricted (*dominated variables*). Dual constraint j becomes redundant if the above holds, so the dominated variable is fixed on its appropriate bound and eliminated from the problem. If the reduced cost $z_j - t_j$ has a weak sign restriction (it is nonnegative or nonpositive), then the variable is a *weakly dominated* one. Surprisingly, if this is the case, then (under some additional conditions [30]), the variable can also be eliminated.

8. Dual constraint limits (16) are used to generate implied bounds on dual variables. A technique similar to that of point 4 is applied. Implied bounds tighter than the original ones replace those of (15).

All techniques mentioned by now are worth to be run before and LP solver is applied. They usually reduce the problem size considerably; sometimes they identify primal or dual infeasibility or unboundedness.

Improving sparsity of the problem

The way in which the LP constraint matrix is involved in interior point iterations justifies further presolve effort that aims at decreasing the cost of calculating the solution of the equations (7) and improving the accuracy of solutions. Both (never mind which form of equation is used to compute the search directions (9) or (11)), are strongly influenced by the sparsity structure of A. The later depends very much on the conditioning of matrix A and requires at least that it has full row rank.

Sparsity of A can usually be improved. In general, one can look for a nonsingular matrix $M \in \mathcal{R}^{m*m}$ such that the matrix MA is the sparsest possible. Primal feasibility constraints can in such case be replaced with an equivalent formulation

$$MAx = Mb, \tag{17}$$

much more suitable for direct application of interior point solver. Exact solution of this *Sparsity Problem* [14] is an NP–complete problem but efficient heuristics [2, 14, 30] usually produce satisfactory nonzero reductions in A. The algorithm of [30], for example, looks for such a row of A that has a sparsity pattern being the subset of the sparsity pattern of other rows and uses it to pivot out nonzero elements from other rows.

Even very sparse A can sometimes produce relatively dense factors in (9) or (11). The later, additionally, fills dramatically if dense columns are present in A. If the number of dense columns is not excessive, then the technique of splitting them into shorter pieces [29, 73] might be a remedy. Note that the augmented system approach (9) suffers less from the presence of dense columns (see e.g., Section 3.3).

Full rank of matrix A

Theoretically, detecting rank deficiency of A is not a problem. One can continue the Gaussian Elimination on A until zero submatrix is obtained. To make this process reliable, complete pivoting should be used [28], which is prohibitively expensive due to the destruction of the sparsity structure. A practicable, generalized Markowitz pivoting [64] can be used to eliminate most linear dependencies with a reasonable cost. However, not all the codes offer such an option. On the other hand, a lot of linear dependencies can be identified with a search for duplicate rows and with the heuristics to make A sparser.

Let us observe that, due to primal degeneracy, that is practically always present, the normal equations matrix (11) becomes rank deficient when optimum is approached even in a case when original A has full row rank (for more details see section 4.1). All codes are thus equipped with safeguards against such problems and usually recover from small rank deficiency of A.

Some numerical experiments

Tables 1 and 2 illustrate advantages of presolve effort. They report problem sizes, normal equations statistics (the number of off-diagonal nonzeros in the adjacency structure AA^T, the number of off-diagonal nonzeros in Cholesky factor L and the number of flops required to compute L), and the iterations and CPU time to solve some Netlib tests to 8–digits accuracy in two cases: without presolve analysis and after it, respectively. Additionally, Table 2 reports CPU time of presolve analysis.

Those and all the following computational results reported in this paper have been obtained with the HOPDM code [4, 30]. The code is written in FORTRAN. It has been compiled with the

Table 1. Original problem sizes and solution statistics.

Problem	Original problem size			Normal equations			Solution	
	m	n	nonz	nz(AA^T)	nz(L)	flops	iters	time
25fv47	820	1571	10400	11074	33217	2.42e6	26	21.41
80bau3b	2235	9301	20413	9972	40038	2.25e6	48	78.23
bnl2	2280	3489	13999	13457	84184	1.32e7	36	103.33
cycle	1886	2857	20720	27714	87322	8.59e6	27	66.88
d2q06c	2171	5167	32417	26991	163407	3.46e7	31	237.77
ganges	1309	1681	6912	7656	30563	1.67e6	19	15.19
pilot	1440	3449	41092	59540	204512	5.25e7	44	513.15
pilot87	2029	4663	70682	115951	489267	2.33e8	44	1935.24
pilotnov	951	1968	12186	10857	48992	5.18e6	19	27.81
sctap3	1480	2480	8874	7386	17973	6.61e5	11	5.47
ship12s	1042	2763	8178	5345	6457	8.08e4	13	3.48
sierra	1222	2016	7252	4896	11635	2.73e5	18	6.53
stocfor2	2157	2031	8343	12738	26282	4.57e5	18	9.00
woodw	1098	8405	37474	20421	48988	3.44e6	28	44.50

F77 compiler with default optimization option (-O) and has been run on a (one processor) SUN SPARC 10 workstation.

The analysis of results collected in Tables 1 and 2 shows that presolve analysis is a valuable technique that often leads to considerable savings of the solution time. These savings, however, are very much problem dependent (they are significant for GANGES but only marginal for PILOT).

3.2 Starting point

The choice of a good starting point for an interior point algorithm is still not solved with full satisfaction. Surprisingly, points that are relatively close to the optimal solution (but are not well centered) lead often to bad performance and/or numerical difficulties.

IPMs (also infeasible IPMs) are quite sensitive to the choice of an initial point. Fortunate guess of $(x^0, s^0, y^0, z^0, w^0)$ (possible for some well understood linear programs) can reduce the computational effort considerably. On the other hand, bad choice may be disastrous for the efficiency as many iterations will have to be done before the iterates get reasonably centered and the algorithm allows large steps.

The starting points in most implementations of primal–dual infeasible IPMs [4, 48, 55] are some variation of the approximate solution of the following auxiliary QP problem

$$\text{minimize} \quad c^T x + \tfrac{\varrho}{2}(x^T x + s^T s),$$

Table 2. Advantages of presolve analysis.

Problem	Size after presolve			Normal equations			Solution		Presolve
	m	n	nonz	nz(AA^T)	nz(L)	flops	iters	time	time
25fv47	769	1535	9959	10456	33377	2.55e6	28	24.05	1.32
80bau3b	1965	8736	19048	9176	37320	2.13e6	43	65.55	2.99
bnl2	1848	3007	12458	12242	79538	1.22e7	30	78.49	2.44
cycle	1400	2403	14111	18376	53182	3.54e6	41	57.81	2.57
d2q06c	2012	4964	30263	24860	139274	2.54e7	41	254.37	5.06
ganges	840	1172	5487	6627	11765	2.43e5	20	6.90	1.17
pilot	1350	3329	40506	58265	199640	5.18e7	44	501.63	7.98
pilot87	1968	4595	70361	115290	469878	2.18e8	43	1696.19	9.54
pilotnov	830	1861	11466	9175	38896	3.64e6	19	23.89	2.63
sctap3	1346	2356	8229	6694	14767	3.89e5	13	6.54	1.03
ship12s	340	1919	4273	2139	2191	1.71e4	14	2.96	0.49
sierra	1129	2008	6956	4418	10147	2.07e5	18	7.05	1.27
stocfor2	1968	1854	7064	9806	21180	3.30e5	19	8.70	0.71
woodw	703	5347	19727	12611	31237	1.90e6	33	38.28	3.02

$$\text{subject to} \qquad Ax = b, \qquad\qquad (18)$$
$$x + s = u,$$

where ϱ is a predetermined weight parameter. A solution of (18) can be given by an explicit formula and can be computed at the cost comparable to a single interior point iteration. It is supposed to minimize the norm of the primal solution (x, s) and promotes points that are better in the sense of the LP objective. As the solution of (18) may have negative components in x and s, those negative components are pushed towards positive values sufficiently bounded away from zero (all elements smaller than δ are replaced by δ, say, $\delta = 1$). Independently, an initial dual solution (y, z, t) is chosen similarly to satisfy $y = 0$ and the dual constraint (2). Again, all elements of z and t smaller than δ are replaced by δ.

It is worth to note that the code of [77] shows consistent good efficiency for a simple starting point $(x^0, s^0, y^0, z^0, w^0) = (e, e, 0, e, e)$, a phenomena not known for a primal–dual method (for more details see Sections 6.2 and 6.3).

3.3 The linear algebra

Every iteration of an interior point method for linear programming requires computing at least one Newton's direction for the first order optimality conditions. This, in turn, is equivalent to computing the projection of some vector of $\mathcal{R}^n$ onto the null space of the linear operator AD. The diagonal scaling matrix D depends on the variant of the algorithm but the computational effort remains practically the same for all interior point algorithms. This explains why a comparison of the efficiency of different algorithms is often limited to the comparison of iteration numbers to reach the desired accuracy.

As was shown in Section 2, the large, sparse system of Newton's equations reduces to the so–called augmented system (9). After scaling the primal component of the search direction $\Delta\tilde{x} = -D^{-1}\Delta x$ and substituting $g = Dr$, this system becomes

$$\begin{bmatrix} I & \tilde{A}^T \\ \tilde{A} & 0 \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta y \end{bmatrix} = \begin{bmatrix} g \\ -h \end{bmatrix}, \qquad (19)$$

where $\tilde{A} = AD$. Usually this system is referred to as the *augmented system*. Observe that equations (19) define the unique orthogonal decomposition of g into $\Delta\tilde{x} \in Ker(\tilde{A})$ and $\tilde{A}^T\Delta y \in Im(\tilde{A}^T)$ when $h = 0$.

The system (19) has much better stability properties [6, 7] than its reduced form obtained after eliminating $\Delta\tilde{x}$

$$(\tilde{A}\tilde{A}^T)\Delta y = \tilde{A}g + h, \qquad (20)$$

which is called the *normal equation*. Direct solution of (19) needs Bunch-Parlett [13] factorization of the large, sparse, symmetric but not positive definite matrix

$$\hat{L}\Lambda\hat{L}^T = \begin{bmatrix} I & \tilde{A}^T \\ \tilde{A} & 0 \end{bmatrix}, \qquad (21)$$

where $\hat{L}$ is a unit lower triangular matrix and Λ is a block diagonal matrix with diagonal blocks of size one or two. As the matrix is indefinite, there is no guarantee that nonzero diagonal elements can be found in all intermediate steps of (symmetric) Gaussian Elimination. Cholesky decomposition thus cannot, in general, be found and the possible way to overcome the difficulties is to allow indefinite 2x2 block pivots in Λ. The augmented system approach has several advantages:

1. Good accuracy properties.
2. Easily generalizable to exploit the sparsity of KKT systems arising in nonseparable quadratic programming and linear complementarity problems.
3. Naturally handles free variables. If $x_j \in (-\infty, +\infty)$, then D_j^{-2} in (9) is replaced by zero.
4. Dense columns of A do not degrade its efficiency, do not lead to significant fill in.

It has two important disadvantages. It is more complicated to implement and it remains, in the average, less efficient (about 40%) than its counterpart applying Cholesky decomposition to the normal equations matrix (20)

$$LL^T = \tilde{A}\tilde{A}^T. \tag{22}$$

The reason of the efficiency of the Cholesky decomposition comes from the fact that matrix $\tilde{A}\tilde{A}^T$ is always positive definite therefore the sparsity preserving sequence of pivots (ordering) in which the symmetric Gaussian Elimination is performed can be defined in advance, i.e., before the numerical operations start. Thus the *Analyse* phase is completely separated from the *Factorise* phase [18]. Even more important, the sparsity pattern of $\tilde{A}\tilde{A}^T$ is the same in all interior point iterations so the Analyse phase has to be done only once, before optimization.

If we choose to solve the augmented system (19) by Gaussian Elimination with 1x1 pivots from the upper left diagonal block, we end up with the normal equations. Due to stability and fill–in reasons, other pivot sequences, possibly including 2x2 pivots, are preferable. These sequences are influenced by the numerical values in (19). They cannot be determined in advance on the sole basis of the sparsity structure analysis. To choose stable pivots (to get an improvement over normal equations) one has to inspect actual numerical values at the same time. Hence the analyse and factorize phases cannot be separated: the factorization is more expensive than in the normal equations approach. Additionally, due to changes of the diagonal scaling matrix D in subsequent IPM's iterations, the pivot sequence should be redefined. In practice, it suffices to update it once every couple of iterations (only if numerical stability detoriates) and, as the experience of [22] shows, rarely more than one such update is needed. An advantage of the augmented system approach is that more freedom in the pivot choice opens the possibility of getting sparser factors than in the normal equations approach (e.g., degrading influence of dense columns can be avoided).

Before we pass to a brief description of the Cholesky decomposition let us comment again on the accuracy of the two competitive approaches. System (19) is definitely more stable than (20) [7]. To further improve stability both approaches can be equipped with easy to implement techniques that improve their accuracy, like e.g., iterative refinement or the use of a regularizing term added to $\tilde{A}\tilde{A}^T$ (that results in bounding all pivots away from zero). Computational experience proves that the normal equations approach also produces sufficiently accurate directions to reach the desired 8–digits correct solutions for practical linear programs [30, 23, 50].

Although this is the case, several researchers [19, 22, 72, 74] have decided to incorporate the augmented system approach [7] in their IPM implementations. We thus find it interesting to give a bit of insight in the computational effort of the two competitive approaches. Table 3

Table 3. Augmented system vs normal equations.

Problem	Augmented system		Normal equations				
	nz(L)	flops	nz(AA^T)	flops(AA^T)	nz(L)	flops(L)	total flops
25fv47	48183	3.41e6	10456	8.56e4	33377	2.55e6	2.64e6
80bau3b	60965	2.36e6	9176	5.76e4	37320	2.13e6	2.19e6
bnl2	93113	1.24e7	12242	5.87e4	79538	1.22e7	1.23e7
cycle	65687	2.72e6	18376	1.33e5	53182	3.54e6	3.68e6
d2q06c	177415	2.41e7	24860	3.77e5	139274	2.54e7	2.58e7
ganges	22005	4.97e5	6627	4.71e4	11765	2.43e5	2.90e5
pilot	221610	4.20e7	58265	1.09e6	199640	5.18e7	5.29e7
pilot87	-	-	115290	2.55e6	469878	2.18e8	2.21e8
pilotnov	57614	5.35e6	9175	1.31e5	38896	3.64e6	3.77e6
sctap3	24405	5.69e5	6694	3.58e4	14767	3.89e5	4.25e5
ship12s	10656	8.44e4	2139	9.94e3	2191	1.71e4	2.70e4
sierra	20886	3.12e5	4418	2.52e4	10147	2.07e5	2.32e5
stocfor2	28119	4.11e5	9806	3.90e4	21180	3.30e5	3.69e5
woodw	56641	2.30e6	12611	1.06e5	31237	1.90e6	2.01e6

reports results of running them on several Netlib problems. It reports the number of nonzeros in triangular factors and the effort (flops) to compute them. In case of normal equations, we distinguish the effort to form AA^T and to factorize it and we report the total effort being the sum of the former two. The results for normal equations come from HOPDM code while the results for the augmented system approach have been reproduced from Table A.3 of [22]. Note that the problem PILOT87 was not solved in [22].

It would undoubtedly be advantageous to be able to pick up the preferable approach after the preliminary analysis of the sparsity structure of the LP constraint matrix A. This needs both methods to be incorporated to the implementation, which is quite unusual yet.

Theoretically the most difficult in an efficient implementation of the Cholesky decomposition is the Analyse phase, i.e., reordering for sparsity. The goal of it is to find a permutation matrix P such that the Cholesky factor of $P\tilde{A}\tilde{A}^T P^T$ is the sparsest possible. This is, unfortunately, an NP–complete problem [78]. In practice, a suboptimal solution is usually found by applying some reordering heuristics. These heuristics work very well in practice. In a marginal time spectacular density reduction of the factors is reached. The most popular of them are the *minimum degree* and the *minimum fill–in* orderings that we shall now briefly describe.

Minimum degree ordering

Markowitz [51] observed that in the kth step of an (unsymmetrical) sparse Gaussian Elimination locally the best pivot candidate a_{ij} is the one that minimizes

$$f_{ij} = (r_i - 1)(c_j - 1),$$

where r_i and c_j are the numbers of nonzero entries of row i and column j in the kth Schur complement. The value f_{ij} gives the number of flops required by the kth step of Gaussian Elimination and, at the same time, estimates the potential fill-in caused by this step. The pivot sequence found that way should thus well prevent excessive fill-ins and ensure a low cost of the factorization. Tinney and Walker [71] applied this strategy to symmetric matrices. When symmetric positive definite systems are considered, pivot selection is restricted to diagonal elements, hence the Markowitz merit function simplifies to

$$f_j = (c_j - 1)^2, \tag{23}$$

and leads to a simple rule that the best candidate for the pivot column is the one with the minimum number of nonzero entries. Interpreting this process in terms of the elimination graph [25], one sees that this is equivalent to the choice of the node that has the minimum degree, which gave the name to this heuristic.

The key point of an efficient implementation of the minimum degree ordering is the representation of fill–ins as the elimination proceeds [18, 25]. The algorithm keeps trace of the elimination process storing only pivotal cliques. A clique denotes here a set of row numbers of all, say, p rows that are active in a given pivotal step. The storage of a clique needs remembering only p row numbers, while the symmetric matrix represented by it has $p(p+1)/2$ elements. The sparsity pattern of the Schur complement at the kth step of Gaussian Elimination (needed to determine the next minimum degree column) is thus represented implicitly by the sparsity pattern of the decomposed matrix and the pivotal cliques from previous steps of the elimination.

Modern implementations of the minimum degree ordering use several enhancements of the basic algorithm [25] and are extremely powerful. Their detailed discussion is beyond the scope of this paper. However, we would like to focus on one of them that plays a particularly important role. It is a technique that takes advantage of the presence of *indistinguishable* nodes called also *supernodes* in the elimination graph (a set of columns with identical sparsity patterns). Instead of storing several identical cliques, this technique handles the whole supernode with only one clique, which leads to obvious savings in a time–consuming degree update step of the reordering algorithm. Note that the form of the normal equations matrix

$$AA^T = \sum_{j=1}^{n} a_j a_j^T,$$

where a_j denotes the jth column of matrix A, explains the natural tendency to creating many supernodes as every column a_j of A creates (subject to a symmetric row and column permutation) a dense window in AA^T and larger windows produced by denser columns often cover smaller ones.

Minimum fill–in ordering

Let us observe that, in general, the function (23) considerably overestimates the expected number of fill–ins in a given iteration of the Gaussian Elimination because it does not take into account the fact that in many positions of the predicted fill–in, nonzero entries already exist. It is possible that another node, although more expensive in terms of (23), would produce less fill–in as the elimination step would mainly update already existing nonzero entries of the Schur complement. An analysis that exactly predicts fill–in and chooses the pivot producing the minimum number of it (minimum fill–in ordering) is much more involved than the minimum degree ordering. To count predicted fill–in one has to simulate the elimination step, which is quite an expensive operation. Surprisingly, in general, this technique does not offer sufficient advantage over the minimum degree ordering to justify its use.

Our discussion concentrated by now on the Analysis phase of the decomposition. However, the dominating term in the computational effort of the interior point algorithm is the (repeated several times) numerical factorization. From the mathematical point of view, once the pivot order has been found, the effort of the numerical factorization is uniquely determined

$$FLOPS = \sum_{i=1}^{m} l_i^2, \tag{24}$$

where l_i denotes the number of nonzero elements in column i of L. However, in practice, its efficiency still depends very much on how the computations are organized and how well do they exploit the specific computer architecture like parallelism, vectorization, cache memory, the use of Basic Linear Algebra System (BLAS) routines, etc. Lower level BLAS routines offer, for example, loop unrolling technique that vectorizes very well. Higher level BLAS routines contain block versions of decomposition algorithms that take advantage of the more efficient organization of the matrix-vector and matrix-matrix products. The gain that results from the use of BLAS may vary significantly on different computers.

Detailed discussion of these issues is beyond the scope of this paper. Undoubtedly, specializations of the interior point implementations to different computer architectures will draw much attention of the LP community in the near future. The interested reader may consult preliminary results of [10, 35] and the references therein. A variant of the Cholesky decomposition that suits particularly well parallelization comes from [20].

Instead, we shall present a technique that gives considerable time savings on all computer architectures. The savings vary significantly for different computers. This technique consists in a switch to dense code near the end of decomposition [18]. It exploits the fact that triangular factors become practically completely dense in the last steps of Gaussian Elimination. Loops over nonzero entries of the pivot column can then avoid indirect addressing (needed to handle sparse columns).

Table 4 compares the efficiency of our implementation in the case when a switch to dense mode is done (default) with the case when it is disabled. It, additionally, reports the number of decomposition flops required to compute Cholesky factorization of AA^T, that part of this effort which is done in dense mode and the size of the dense window. Even on a SUN SPARC 10 computer (without any vectorization facilities) considerable savings are obtained if a dense window is large. Obviously, the savings would be remarkably larger on other computer architectures that take advantage of the specific processor features (e.g., on IBM's RISC 6000 workstation).

Our discussion in this section has naturally concentrated on *direct* approaches to solving reduced KKT systems that for general large scale LPs are definitely the methods of choice. Saunders' remark [67]: "Major Cholesky would feel proud" seems thus well summarize current state of the art of IPMs implementations.

To complete the discussion of different methods of computing projections, let us mention also *iterative* approaches. A classical method from this family, i.e., conjugate gradients algorithm has

Table 4. Advantages of switch to dense mode.

Problem	No window		With dense window				
	iters	time	iters	time	Total flops	Dense flops	window
25fv47	28	26.78	28	24.05	2.55e6	1.25e6	155
80bau3b	43	69.43	43	65.55	2.13e6	7.41e5	130
bnl2	30	93.79	30	78.49	1.22e7	7.92e6	287
cycle	41	60.39	41	57.81	3.54e6	9.05e5	139
d2q06c	41	300.06	41	254.37	2.54e7	1.49e7	354
ganges	20	7.02	20	6.90	2.43e5	5.10e4	53
pilot	44	575.02	44	501.63	5.18e7	2.27e7	408
pilot87	43	2172.71	43	1696.19	2.18e8	1.51e8	768
pilotnov	19	25.75	19	23.89	3.64e6	1.25e6	155
sctap3	13	6.82	13	6.54	3.89e5	1.12e5	69
ship12s	14	2.96	14	2.96	1.71e4	-	-
sierra	18	7.39	18	7.05	2.07e5	8.53e4	63
stocfor2	19	9.07	19	8.70	3.30e5	1.62e4	36
woodw	33	40.11	33	38.28	1.90e6	4.62e5	111

been applied to solving reduced KKT systems by several researchers but, up to our knowledge, it has never proved competitive when applied to general large scale linear programs. The reason of this is an always present ill–conditioning of KKT systems. The only way to overcome extremely slow convergence of an iterative method is the use of a good preconditioner but, for general linear programs there is no way, to date, to find such a preconditioner at an acceptable computational effort. Instead, for specially structured problems good preconditioner derived from the problem interpretation can sometimes be found [65, 62].

3.4 Stepsize

Most of interior point methods require that all iterates belong to a neighborhood of the central path [31]. One way to keep this condition satisfied is to allow only small reduction of the barrier parameter with a factor

$$\beta = \frac{1}{1+\gamma/\sqrt{n}}, \tag{25}$$

where $\gamma \in (0, 0.1)$ that, in consequence, leads to very short steps (*short step methods*). Although, such an approach allows proving nice theoretical properties of the algorithm (all iterates are very close to the central path), it leads to hopelessly slow convergence as the worst case behavior becomes the practice.

In *medium step methods* (when $\gamma = \Omega(1)$) and *long step methods* (when $\gamma = \Omega(\sqrt{n})$), a more optimistic target is chosen and, consequently, the iterates are allowed to move in a much larger neighborhood of the central path but more damped Newton steps are required to get close to the new target. The complexity of the long step methods becomes $\mathcal{O}(n)$, which is worse than that of the the short step methods. In practice, these methods offer very fast convergence, significantly faster than is given by the worst case complexity analysis.

In current implementations we do not bother about getting really close to the target. The target is updated after each Newton step. Even worse, different stepsizes are used in primal and dual spaces and the only concern is preserving the nonnegativity of the variables. More precisely, maximum possible stepsizes are determined by the formulae

$$\begin{aligned} \alpha_P &:= \max \alpha > 0 : \ (x,s) + \alpha(\Delta x, \Delta s) \geq 0, \\ \text{and} \qquad \alpha_D &:= \max \alpha > 0 : \ (z,t) + \alpha(\Delta z, \Delta t) \geq 0, \end{aligned} \tag{26}$$

and these stepsizes are slightly reduced with a factor $\alpha_0 = 0.99995$ to prevent hitting the boundary. The use of such a stepsize rule saves about 40% of the iterations number compared with the case when the theory of the long step methods is strictly followed.

The use of different stepsizes in different spaces is due to Kojima et al. [45]. However, to preserve the polynomial complexity, some additional safeguards were needed.

In another paper Kojima, Megiddo and Mizuno [44] prove the global convergence of the infeasible primal–dual method. In their stepsize selection, only nonnegativity of variables is concerned like in (26) but additional conditions are imposed that ensure more uniform progress in reducing both primal and dual feasibility and approaching optimality.

The results collected in Table 5 illustrate the efficiency of our primal–dual implementation measured with the number of iterations to reach optimality in two cases: 1) always identical stepsizes are chosen in both spaces and 2) different stepsizes are allowed (default). These results clearly

Table 5. Different stepsizes in different spaces.

Problem	$\alpha_P = \alpha_D$	$\alpha_P \neq \alpha_P$
25fv47	31	28
80bau3b	44	43
bnl2	33	30
cycle	49	41
d2q06c	45	41
ganges	21	20
pilot	47	44
pilot87	45	43
pilotnov	19	19
sctap3	13	13
ship12s	15	14
sierra	20	18
stocfor2	21	19
woodw	35	33
whole Netlib	1996	1778

show that restricting primal and dual stepsizes to be the same causes remarkable loss of the efficiency (218 iterations on 90 problems).

3.5 Centering and higher order methods

Almost all interior point algorithms compose the direction step $(\Delta x, \Delta s, \Delta y, \Delta z, \Delta t)$ (denoted with Δ for short) from two parts

$$\Delta = \Delta_a + \Delta_c. \tag{27}$$

i.e., combine affine–scaling, Δ_a and centering, Δ_c components. The term Δ_a is obtained by solving (7) for $\mu = 0$ and Δ_c is the solution of equation like (7) for the right hand side

$$(0, 0, 0, \mu e - XZe, \mu e - STe)^T,$$

where $\mu > 0$ is some centering parameter ($\mu = \frac{x^T z + s^T t}{2n}$, for example, refers to centering that does not change the current complementarity gap). The term Δ_a is responsible for "optimization" while Δ_c keeps the current iterate away from the boundary.

All path–following methods can benefit from the use of the predictor–corrector technique. This technique takes into account higher order terms in the Newton's direction and estimates targets that are more likely to be achieved. Undoubtedly, one of the reasons why it proved so successful in the primal–dual method is the very good Mehrotra's heuristics defining the new target (28) that takes advantage of the knowledge of a current and predicted complementarity gap. We shall now address this technique in more detail.

Introducing higher order terms is combined in it with the adaptive choice of the barrier parameter μ based on the analysis of the reduction of the complementarity gap achievable when moving in a (predictor, $\mu = 0$) primal-dual affine scaling direction.

The affine scaling (predictor) direction Δ_a solves the linear system (7) for the right hand side equal to the current violation of the first order optimality conditions for (1)–(2), i.e., with $\mu = 0$. This direction is usually "too optimistic" — if a full step of length one could be made in it, the LP problem would be solved in one step. Predictor–corrector makes a hypothetical step in this direction. The maximum stepsizes in the primal, α_{Pa} and in the dual, α_{Da} spaces preserving nonnegativity of (x, s) and (z, t), respectively are determined and the predicted complementarity gap

$$g_a = (x + \alpha_{Pa}\Delta x)^T(z + \alpha_{Da}\Delta z) + (s + \alpha_{Pa}\Delta s)^T(t + \alpha_{Da}\Delta t)$$

is computed. It is then used to determine the barrier parameter

$$\mu = \left(\frac{g_a}{g}\right)^2 \frac{g_a}{n}, \tag{28}$$

where $g = x^T z + s^T t$ denotes current complementarity gap. The term g_a/g measures the achievable progress in the affine scaling direction. If only a small step in the affine scaling direction can be made, then g_a/g is close to one and $\mu = g/n$, which means that we only want to improve centrality. If the affine scaling direction offers considerable progress in the reduction of the complementarity gap, then more optimistic target (closer to the optimum) is chosen.

For such a μ, the corrector direction Δ_c is computed

$$\begin{bmatrix} A & 0 & 0 & 0 & 0 \\ I & 0 & I & 0 & 0 \\ 0 & A^T & 0 & I & -I \\ Z & 0 & 0 & X & 0 \\ 0 & 0 & T & 0 & S \end{bmatrix} \begin{bmatrix} \Delta x_c \\ \Delta y_c \\ \Delta s_c \\ \Delta z_c \\ \Delta t_c \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \mu e - \Delta X_a \Delta Z_a e \\ \mu e - \Delta S_a \Delta T_a e \end{bmatrix}, \tag{29}$$

and, finally, the direction Δ of (27) is determined.

A single iteration of the (second order) predictor-corrector primal-dual method needs thus two solves of the same large, sparse linear system for two different right hand sides: first to solve (7) with $\mu = 0$ and then to solve (29).

We should note here that the above presentation of the predictor–corrector technique follows the computational practice. It abuses mathematics in the sense that stepsizes α_P and α_D are not taken into account in it (one would expect this when using higher order Taylor approximations). The reader interested to see a detailed rigorous presentation of this approach is encouraged to consult [56].

Let us observe that the predictor–corrector mechanism can be applied repeatedly leading thus to higher (than two) order method. Direction Δ of (27) can be used as a predictor, and a new corrector term Δ_c' can later be computed and added to it. A method of order k would compute $k-1$ corrector terms and would require solving the same linear system for k different right hand sides.

Although the use of higher order terms usually results in the reduction of the number of iterations to reach optimality, it does not necessarily produce time savings due to the increased effort in a single iteration. To date, second order methods seem, in the average, to be the most efficient. (They are used in most implementations [23, 30, 50, 55].) No one knows yet how to benefit practically from higher order information. A possible way to take advantage of it is to promote higher order terms if the Cholesky factorization is very expensive compared with a single solve (see, e.g., [4] for details).

We end this section with a demonstration of the use of higher order techniques. Table 6 reports the number of iterations to reach 8–digits precision in the duality gap and the CPU time for several Netlib problems for different order methods. The method of order 1 is a pure primal–dual one in which the direction comes from the solution of (7). The method of order 2 is a classical predictor–corrector one in which one corrector term has been computed. The method of order k computes $k-1$ corrector terms.

The analysis of results collected in Table 6 shows that the use of higher order information may reduce the number of iterations to reach optimality. However, this translates into computation time savings only in a case when Cholesky factorizations are very expensive (compare to the data of Table 4).

3.6 Stopping criteria

Interior point algorithms terminate when the relative primal and dual feasibility and the relative duality gap are reduced to a predetermined tolerance. In case of the primal–dual method, the following conditions are checked to state p–digits accurate solutions

$$\frac{||Ax-b||}{1+||x||_\infty} \leq 10^{-p} \quad \text{and} \quad \frac{||x+s-u||}{1+||x||_\infty+||s||_\infty} \leq 10^{-p}, \tag{30}$$

$$\frac{||A^Ty+z-w-c||}{1+||z||_\infty+||w||_\infty} \leq 10^{-p}, \tag{31}$$

$$\frac{|c^Tx-(b^Ty-u^Tw)|}{1+|b^Ty-u^Tw|} \leq 10^{-p}. \tag{32}$$

It is worth to note that scaling may change the left hand sides of equations (30-32) and, consequently, the measures of the quality of solutions, significantly. The denominators of these left hand sides usually decrease after scaling the problem.

An 8–digits exact solution ($p=8$) is typically required in the literature.

In practice, it is extremely rare that condition (32) is satisfied and at the same time one of the conditions (30) or (31) does not hold. The explanation of this phenomena comes from the analysis of the first order optimality conditions (5). Observe that the first three equations, that impose primal and dual feasibility, are linear. They are thus "easier" to be satisfied for the Newton's method than the last two equations that are nonlinear and, additionally, change in subsequent

Table 6. Efficiency of higher order methods.

Problem	Order 1		Order 2		Order 3		Order 4	
	iters	time	iters	time	iters	time	iters	time
25fv47	44	33.26	28	24.05	25	24.98	23	24.44
80bau3b	65	74.00	43	65.55	35	66.19	40	82.75
bnl2	58	125.54	30	78.49	25	73.20	27	80.42
cycle	63	74.15	41	57.81	33	51.88	33	55.10
d2q06c	65	364.35	41	254.37	36	243.84	33	236.53
ganges	32	7.50	20	6.90	19	8.31	19	8.45
pilot	56	617.06	44	501.63	34	405.90	32	389.20
pilot87	63	2419.63	43	1696.19	35	1431.03	33	1384.22
pilotnov	33	33.24	19	23.89	17	24.39	15	23.68
sctap3	24	8.64	13	6.54	12	7.94	11	8.69
ship12s	26	3.63	14	2.96	14	4.46	13	4.75
sierra	30	8.87	18	7.05	16	8.65	16	9.88
stocfor2	31	12.90	19	8.70	17	10.48	17	11.80
woodw	54	49.07	33	38.28	27	41.31	26	43.36

Table 7. Attaining feasibilities and optimality.

Problem	Iterations to reach			Accuracy of solution		
	Pr. Feas.	Dl. Feas.	Opt.	Pr. Inf.	Dl. Inf.	Opt. Gap
25fv47	25	13	28	3.3e-12	5.8e-9	1.9e-9
80bau3b	23	20	43	1.2e-12	9.7e-9	4.3e-9
bnl2	10	22	30	1.1e-11	4.4e-10	6.9e-9
cycle	25	41	41	1.3e-14	7.0e-9	2.7e-9
d2q06c	20	13	41	7.1e-15	5.7e-12	2.5e-10
ganges	15	14	20	3.2e-11	4.1e-9	8.3e-11
pilot	44	11	44	4.6e-9	1.6e-13	6.6e-9
pilot87	39	43	43	2.4e-11	8.8e-9	3.2e-9
pilotnov	15	7	19	1.0e-12	1.8e-10	2.0e-10
sctap3	13	13	13	2.9e-12	2.4e-12	1.0e-9
ship12s	12	13	14	2.6e-10	1.1e-12	6.9e-11
sierra	10	16	18	7.9e-13	1.8e-9	4.8e-9
stocfor2	13	18	19	3.4e-12	2.3e-13	1.6e-10
woodw	32	10	33	1.8e-12	1.2e-10	3.9e-10

interior point iterations. Consequently, the most important and perhaps the only condition that really has to be checked is (32).

Results collected in Table 7 report the numbers of primal–dual iterations at which the primal and the dual relative feasibilities and the relative optimality gap were reduced to 10^{-8} for the subset of Netlib problems. Its last three columns give the relative infeasibilities and the relative optimality gap for the 8-digits optimal solution.

Note, that for infeasible problems or for problems with unbounded level sets the scaling of infeasibilities is not appropriate. In these cases the iterates diverge, hence the denominator goes to infinity which might lead to false results. The proper normalization factor, in these cases, would be $1 + ||b||_\infty$ and $1 + ||c||_\infty$, respectively.

3.7 Theoretical vs practical worst–case complexity

Theoretical worst case complexity of the simplex method is not known to be polynomial but this never caused problems in practice. In fact, modern simplex codes rarely do more than $m + n$ iterations to reach optimality.

Although the worst case complexity of interior point methods is polynomial, the situation is somehow similar. Theoretical bound of $\mathcal{O}(\sqrt{n}\ \log \frac{1}{\epsilon})$ iterations to obtain an ϵ–exact solution to LP is still too pessimistic. In practice, the number of iterations is something like $\mathcal{O}(\log n)$. It is very rare for example that the predictor–corrector primal–dual infeasible IPM makes more that 50 iterations to reach 10^{-8}–optimality. This is never the case when the HOPDM code [30] is applied to solving 90 LPs from the Netlib suite (only 7 problems need more than 40 iterations to be solved).

4 Remarks on numerical difficulties

4.1 Degeneracy and IPMs

The main computational step in all IPMs is solving the system of normal equations

$$AD^2A^Tp \;=\; q, \tag{33}$$

for some q, where D is a diagonal matrix (11). Güler and Ye [34] showed that in all central path following methods, any limit point (x^*, z^*) of the iterates (x^k, z^k) is in the relative interior of the optimal faces. In fact, they prove that there exists a constant γ, where $0 < \gamma < 1$, such that the relations

$$\gamma \leq x_j^k \leq 1/\gamma \quad \text{for} \quad j \in P_x, \tag{34}$$

$$\gamma \leq z_j^k \leq 1/\gamma \quad \text{for} \quad j \in P_z, \tag{35}$$

are satisfied for all $k \geq 0$. Here (P_x, P_z) is again the optimal partition of the LP problem.

Note that $x_{P_z}^k \rightarrow 0$ and $z_{P_x}^k \rightarrow 0$ in any convergent IPM. Therefore, the limiting behavior of $A_{P_x}(D_{P_x}^k)^2A_{P_x}^T$ determines the asymptotic behavior of the linear systems (33). For primal–dual path-following methods we can obtain more information about the matrices $D_{P_x}^k$ as we shall now explain.

For this class of methods $(D^k)^2 = X^k(Z^k)^{-1}$, so for any i,

$$\frac{x_i}{z_i} = \frac{x_i^2}{x^Tz} \cdot \frac{x^Tz}{x_iz_i}. \tag{36}$$

If $i \in P_x$, it follows from relations (34), (35), and (36) that

$$\frac{\gamma^2}{x^Tz} \leq \frac{x_i}{z_i} \leq \frac{1}{\xi\gamma^2(x^Tz)}, \tag{37}$$

where ξ is an appropriate constant depending on the specific IPM. This shows that the condition numbers of the matrices $D_{P_x}^k$ are uniformly bounded and bounded away from zero. Thus, when matrix A_{P_x} has full rank, then matrices $A_{P_x}(D_{P_x}^k)^2A_{P_x}^T$ also have full rank, and the condition numbers of the later matrices are uniformly bounded. The rank of A_{P_x} depends on the degeneracy of the problem, which we discuss below in more detail.

1. If (P) and (D) are both non–degenerate on their respective optimal faces, then both programs have unique solutions and the matrices A_{P_x} and $A_{P_x}(D_{P_x}^k)^2A_{P_x}^T$ are all non–singular. The linear systems (33) are well conditioned, at least when ξ and γ are not too small.

2. If (P) is degenerate and (D) is non–degenerate, matrix A_{P_x} has less than m columns and so the rank of A_{P_x} is less than m and $A_{P_x}(D^k_{P_x})^2 A^T_{P_x}$ is singular. This means that the linear system (33) is ill–conditioned. Numerical problems caused by these ill–conditioning may arise and, in fact, they were early recognized [26]. However, as we shall discuss in Section 4.2 the method itself has an embedded safeguard against them.

3. If (P) is non–degenerate and (D) is degenerate, then the rank of A_{P_x} is equal to m, hence $A_{P_x}(D^k_{P_x})^2 A^T_{P_x}$ is non–singular. This implies that the linear system (33) is well–conditioned, again as long as ξ and γ are not too small. Observe, that if (P) is non–degenerate, the degeneracy status of (D) matters little from the numerical point of view.

4. If both (P) and (D) are degenerate, then not much can be said about the partition (A_{P_x}, A_{P_z}) and the rank of A_{P_x}. Consequently, we can also say nothing about rank of $A(D^k)^2 A^T$ matrices.

4.2 Ill–conditioned normal equations matrix

It became a folklore that every iteration of an IPM needs inversion of the matrix that, as the optimum approaches, becomes extremely ill–conditioned. Although this is generally true, its impact on the accuracy of solutions is no as dramatic as one would expect. Ill–conditioning of the coefficient matrix is not the only factor that determines the difficulty of solving the system of linear equations and the accuracy of solutions. An equally important factor is the relation between the matrix and the right hand side. The following important result [70] theoretically justifies the common experience that numerical problems are rare even if the normal equations matrix is ill–conditioned.

Let us have a closer look at the right hand side of the normal equations system (11). Since it has the form $q = AD^2 r + h$, we may split (11) into two systems of equations

$$AD^2 A^T p_r \;=\; AD^2 r, \tag{38}$$

and

$$AD^2 A^T p_h \;=\; h, \tag{39}$$

Stewart [70] proved, that with given A and r, for any positive diagonal matrix D, the solution of equation (38) belongs to a bounded compact set. In other words, even if the normal matrix become ill–conditioned, this bad property has limited influence on the solution vector p_r.

Unfortunately, in the infeasible primal–dual method, h in (39) is nonzero and has no clear dependence on A and D. Furthermore r changes (see (10)). If the LP is both primal and dual feasible, then all infeasibilities ξ_b, ξ_u and ξ_c go rapidly to zero. Consequently, h (and p_h) vanishes while r converges to a fixed point so Stewart's result ensures the stability of p_r.

If the LP is dual infeasible, then the dual iterates diverge, ξ_c does not converge to zero and r diverges. The above stability result does not apply and, in practice, serious numerical difficulties are observed.

If the LP is primal infeasible, then the primal iterates diverge, primal infeasibilities (ξ_b, ξ_u) do not converge to zero and r diverges. Similarly to the dual infeasible case, the stability result does not apply to (38) and, eventually, as $h = \xi_b$ might not converge to zero, the solution of (39) may cause additional numerical problems.

5 Optimal basis identification

¿From now on in the rest of the paper we consider the LP problem and its dual in the standard form, e.g. the primal variables are nonnegative and have no upper bounds. Such a form simplifies the notations and makes it easier to explain and understand the underlying ideas. Hence the primal problem that we consider is

$$\begin{aligned} \text{minimize} \quad & c^T x, \\ \text{subject to} \quad & Ax = b, \\ & x \geq 0, \end{aligned} \tag{40}$$

and the dual problem is

$$\begin{aligned} \text{maximize} \quad & b^T y, \\ \text{subject to} \quad & A^T y + z = c, \\ & z \geq 0, \end{aligned} \tag{41}$$

where the vectors, matrices have the same dimensions as before.

It is known that a pair of primal and dual feasible solutions x^* and (y^*, z^*) are optimal if $X^* z^* = 0$. The optimal pair is *strictly complementary* if $x^* + z^* > 0$. In this case we have $P_x = \{i \mid x_i^* > 0\}$ and $P_z = \{i \mid z_i^* > 0\}$. Due to the complementarity property $P_x \cap P_z = \emptyset$, hence the strict complementarity property is equivalent to $P_x \cup P_z = \{1, \ldots, n\}$. The partition (P_x, P_z) is uniquely determined by the problem data and is called the *optimal partition* of the LP problem.

5.1 Do we need an optimal basis?

It is well known that if the LP problem is solved by the simplex method, then finally an optimal basis (solution) is produced as the solution of the problem. Of course if the problem is degenerate and multiple optimal solutions are present, then one of the possible optimal basis solutions is generated. In this case IPMs (except Iri–Imai's [36] method), contrary to the simplex algorithm, does not produce an optimal basis solution but a primal–dual optimal pair where the primal optimal solution is a solution from the relative interior of the primal optimal face and the dual optimal solution is a solution from the relative interior of the dual optimal face. Hence IPMs provide an optimal solution with a maximal number of nonzero coordinates in both the primal and dual problems (see e.g., Güler and Ye [34]). It is also proved that these solutions are strictly complementary and they define the optimal partition of the LP problem.

The existence of strictly complementary primal–dual optimal solutions has been proved first by Goldman and Tucker [27]. Balinski and Tucker [9] propose a pivot algorithm to generate such a strictly complementary pair. In contrast, as we will see later on, an optimal basis can be obtained from any primal–dual optimal solution pair in strongly polynomial time.

To have an optimal basis might be important for several reasons. For example, basic solutions have a minimal number of non–zero coordinates, which is advantageous when a solution with a small number of positive coordinates is required for some practical applications. Of course, there are situations when a strictly complementary optimal solution and the knowledge of the optimal partition is needed (see e.g., Greenberg [32]).

Currently, sensitivity and postoptimal analysis are based on the knowledge of an optimal bases in all commercial packages. People are used to this approach, although several papers recognized the potential mathematical errors and negative economical consequences of this approach [75, 37, 32]. Correct postoptimal analysis is possible based on solutions found by IPMs or, on a different cost, by the simplex method. The decision of which approach is preferable depends on the problem instance and is discussed in Subsection 6.1.

To our best knowledge, a basic solution is necessary for cutting plane methods in mixed integer programming. In branch and bound methods the problem has to be reoptimized after some small modification. To date simplex based solvers are more efficient than IPMs if an "almost optimal" solution is available.

For numerical reasons or if we are facing a large mixed integer problem, sometimes a crossover from the IPMs to the simplex method is needed. The efficient techniques for the crossover are similar to the basis identification techniques.

These advantages provide sufficient motivation to examine how one can generate an optimal basic solution from an optimal or near optimal solution obtained by an IPM. It is evident that this question occurs only in the case of degeneracy, since otherwise the primal and dual optimal solutions are unique and are also basic solutions.

5.2 How to get an optimal basis?

Several attempts were made to create efficient methods that produce an optimal basis if a (strictly complementary) optimal solution is available. The best theoretical algorithm, which also turned out to be very efficient in practice (see e.g., [11]) is due to Megiddo [53]. It finds an optimal basis in strongly polynomial time, provided that optimal solutions are available to both (40) and (41). Due to its theoretical and practical importance we discuss it in more detail.

Principally, IPM ends up with a strictly complementary pair of optimal solutions but the scheme is applicable to a slightly more general case. Suppose that x, primal and y, z, dual optimal solutions are available. Let $A = [A_1, A_2, A_3]$, $x = (x_1, x_2, x_3)$, $z = (z_1, z_2, z_3)$, $c = (c_1, c_2, c_3)$ where index 1 refers to the positive coordinates of x, index 2 refers to the zero coordinates of both x and z, and index 3 refers to the positive coordinates of z. Then, we have $A_1x_1 = b$, $x_1 > 0$, $x_2 = 0$, $x_3 = 0$, and $A_1^Ty = c_1$, $A_2^Ty = c_2$, $A_3^Ty < c_3$.

Starting from such a solution and partition we are looking for an optimal basis, i.e., a disjoint partition $(\mathcal{B}, \mathcal{N})$ of the indices such that submatrix of A built of columns from $\mathcal{B}$ is nonsingular, $x_B \geq 0$, $z_B = 0$, $x_N = 0$, and $z_N \geq 0$.

The process will subsequently build $\mathcal{B}$ up and, if necessary, modify x and z to satisfy the above sign requirements. Naturally, all columns whose indices enter $\mathcal{B}$ must be linearly independent to meet the nonsingularity requirement. As the process starts from optimal solution, it will not alter the objective values.

The algorithm is split into three phases. In the first one, a maximal linearly independent subset of the columns of A_1 is built up. At the same time some coordinates of x_1 are purified and so moved to A_2. This continues until the resulted A_1 contains only linearly independent columns. The indices of A_1 form then initial $\mathcal{B}$. If $\mathcal{B}$ is already a basis, we are done. Otherwise, in the second phase it is extended to a maximal independent subset of $[A_1, A_2]$. If $\mathcal{B}$ still is not a basis, then we proceed with third phase. Every step of it purifies some indices of z_3, i.e., drives the corresponding coordinates to zero. All the corresponding columns are independent on the current $\mathcal{B}$ and moved to A_2. They are immediately examined if the current $\mathcal{B}$ can be extended by them (at least one of them has to be egligible). These steps are repeated until $\mathcal{B}$ extends to a basis. The complete algorithm is presented below.

Optimal Basis Identification Algorithm

Initialization

Suppose that x primal and y, z dual optimal solutions are available and that x and z are initially partitioned as explained above.

Reduce the positive part of x:

While the columns of A_1 are dependent do
begin
Find (e.g., by pivoting) a vector t such that $A_1 t = 0$ (this implies $c_1^T t = 0$). Using t, eliminate a positive coordinate (say j) from x_1, while preserving the non–negativity of x_1 (ratio test). Remove column a_j from A_1 and add it to A_2.
end

Let $B = A_1$. (Note that the columns of B are independent at this stage.)

Extend B to a basis:

Extend B using A_2:

While rank$(B) <$ rank$[A_1, A_2]$ do
begin
If rank$[A_1, A_2] >$ rank(B) and column a_j of $[A_1, A_2]$ is independent from B, add a_j to B.
end

Extend B using A_3:

While rank$(B) < m$ do
begin
Find (e.g., by pivoting) a vector u such that $B^T u = 0$ (this implies $A_1^T u = 0$, $A_2^T u = 0$) and $A_3^T u \neq 0$. Note that u satisfies $b^T u = 0$ (since $u^T A_1^T x_1 = b^T u$). Using u, eliminate a positive coordinate (say j) from z_3, while preserving the dual feasibility of z (ratio test). Remove a_j from A_3, and add it to A_2 and B.
end

(We now have an optimal complementary pair (x, z), where rank$(B) = m$. Using the formulae $Bx_B = b$ and $B^T y = c_B$, we see that basis B is optimal.)

Note that only Gaussian Elimination steps (pivoting) are necessary to perform this algorithm. The amount of work involved depends on the degree of degeneracy of the LP problem. In the worst case, not more than n pivots (the dimension of the space) are necessary to identify an optimal basis. The algorithm uses both primal and dual information and generates optimal basic solutions both to the primal and the dual problems.

Megiddo also proves the surprising result that the complexity of identifying an optimal bases if just a primal optimal solution or just a dual optimal solution is available is equivalent to solving an LP problem. This result also demonstrates that the primal–dual approach has an important advantage compared to pure primal or dual methods.

Although the above algorithm is clear and elegant, there are several problems to be solved in its practical implementation. One has, for example, only approximate optimal and approximate complementary solutions, hence tolerances and some safeguards are needed in implementations. Another problem consists in finding the partition P_x, P_z. That is why applying Megiddo's algorithm initialized from an IPM optimal pair, the (x_2, z_2) part usually refers to those variables for which no clear decision can be made if the variable index belongs to P_x or P_z. Special care must also be taken to select a stable basis. Megiddo's algorithm has been efficiently implemented [11, 47, 5]. In the last reference it is theoretically clarified when a guaranteed crossover can take place.

6 Problems to be solved

Although great progress is made concerned with the efficiency of IPM implementations, some relevant questions still remain open. We shall concentrate on only three of them which seem the most important, namely:

- Implementing postoptimal analysis in a correct way.
- Handling problems which have unbounded optimal faces.
  Detecting primal/dual infeasibility.
- Warm start.

6.1 Correct postoptimal analysis

In implementing algorithms for LP one has to consider methods to produce shadow prices and ranges. Currently, implementations of sensitivity and postoptimal analysis in all commercial codes are based on the knowledge of an optimal basis. People are used to this approach, although several papers recognized the potential mathematical errors and negative economical consequences of it ([75, 37, 32]). The ranges and shadow prices obtained from analyzing an optimal bases are usually incorrect as one would like to know the linearity intervals of the optimal value function and its left/right derivatives which are the correct shadow prices. This correct information is obtainable at a different cost. It is proved [3, 37] that there is a one to one correspondence between the optimal partitions and the linearity intervals, break points of the value function. These values can be obtained in the cost of the solution of some smaller LPs. To define these LPs one needs a description of the optimal face. Once a strictly complementary solution is found, the optimal partiton (P_x, P_z) is also known. The primal optimal face is then given by

$$\mathcal{P}^* = \{x \mid Ax = b, \;\; x \geq 0, \; x_{P_z} = 0\},$$

while the dual optimal face is

$$\mathcal{D}^* = \{(y,z) \mid A^T y + z = c, \ \ z \geq 0, \ z_{P_x} = 0\}.$$

Alternatively, if the optimal partition is not known just a primal optimal solution x^* and a dual optimal solution (y^*, z^*) is available then the optimal faces are characterizable as follows.

$$\mathcal{P}^* = \{x \mid Ax = b, \ \ c^T x = c^T x^*, \ \ x \geq 0\},$$

$$\mathcal{D}^* = \{(y,z) \mid A^T y + z = c, \ \ b^T y = b^T y^*, \ \ z \geq 0\}.$$

As mentioned above, to get the linearity intervals and correct shadow prices, the solution of some auxiliary LPs is needed. As illustration, first, we define the two LPs to find the left and right end of the actual linearity interval $[\epsilon_{left}, \epsilon_{right}]$ when a single objective coefficient c_j changes. If the optimal partition (P_x, P_z) is known, one need to solve the following two linear programs:

$$\epsilon_{left} = \min\{\epsilon \mid A^T y + z = c + \epsilon e_j, \ z \geq 0, \ z_{P_x} = 0\},$$

$$\epsilon_{right} = \max\{\epsilon \mid A^T y + z = c + \epsilon e_j, \ z \geq 0, \ z_{P_x} = 0\}.$$

If the optimal partition is not available, just optimal solutions are known, the solution of the two LPs

$$\epsilon_{left} = \min\{\epsilon \mid A^T y + z = c + \epsilon e_j, \ b^T y = c^T x^* + \epsilon x_j^*, \ z \geq 0\},$$

and

$$\epsilon_{right} = \max\{\epsilon \mid A^T y + z = c + \epsilon e_j, \ b^T y = c^T x^* + \epsilon x_j^*, \ z \geq 0\}$$

is necessary. Note, that the validity of the resulting ranges follows from the concavity or convexity, respectively of the optimal value functions.

Similarly, the correct left and right shadow prices can be obtained by solving some auxiliary LP problems. It is obvious that if we are in the interior of a linearity interval of the value function as c_j varies, then the left and right shadow prices are equal and both equals to x_j. If c_j is a breakpoint of the optimal value function, then the left and right shadow prices $(x_{j-left}, x_{j-right})$ can be computed by solving the two LP problems

$$x_{j-left} = \max\{x_j \mid Ax = b, \ x \geq 0, \ x_{P_z} = 0\},$$

$$x_{j-right} = \min\{x_j \mid Ax = b, \ x \geq 0, \ x_{P_z} = 0\}$$

in the optimal partition approach. If the optimal partition is not available the following two LPs are to be solved:

$$x_{j-left} = \max\{x_j \mid Ax = b, \ c^T x = c^T x^*, \ x \geq 0\},$$

$$x_{j-right} = \min\{x_j \mid Ax = b, \ c^T x = c^T x^*, \ x \geq 0\}.$$

If b_i varies then the corresponding values can be obtained similarly on the cost of the solution of some analogous LPs. We do not go into further details here. For a complete analysis based on the optimal partition see [3, 37, 40]; the alternative approaches when the optimal partition is not available are discussed in [75, 57, 40].

To close this section we point out that to get the correct ranges and shadow prices for all the primal and dual variables needs substantially more time than the incomplete (incorrect) optimal basis approach. Hence it is advisable to select a smaller set of "important" variables and to perform the necessary calculations just for this set.

6.2 Unbounded optimal faces, infeasible problems

As it is discussed in Section 2 the current implementations are based on infeasible IPMs. One of the theoretical disadvantages of the implemented infeasible IPMs is that their theoretical complexity is $\mathcal{O}(nL)$ instead of $\mathcal{O}(\sqrt{n}L)$, the best complexity to date. If the optimal face(s) are unbounded, then the duality gap converges to zero while the sequence(s) of solutions (primal, dual or both according to the unboundedness of the optimal faces, respectively) diverges, hence it is difficult to identify optimal solution(s). Similar difficulties occur if the primal or dual or both problems are infeasible. This case always manifests with the divergence of the iteration sequence in infeasible IPMs, which, in practice, might not be easy to identify. Resolving these difficulties is one of the most interesting problems nowadays.

The so–called skew-symmetric self–dual embedding (SSSD) might be a remedy. The SSSD was first introduced by Ye, Todd and Mizuno [79] using the standard form problems (40) and (41). They discussed most of the advantages of this embedding and showed that Mizuno, Todd and Ye's [59] predictor–corrector algorithms solve the LP problem in $\mathcal{O}(\sqrt{n}L)$ iterations, yielding the first infeasible IPM with this complexity. Somewhat later Jansen, Roos and Terlaky [38] presented the SSSD problem for the symmetric form primal–dual LP pair in a concise introduction to the theory of LP based on IPMs. Before presenting the SSSD embedding, the surprisingly nice properties of it are summarized.

1. Self–duality, the dual problem is identical to the primal one. One can solve it as an LP or as an LCP.
2. SSSD is always feasible. Furthermore, the interior of the feasible sets is also non-empty, hence the optimal faces are bounded. IPMs applied to SSSD always converge to an optimal solution.
3. Optimality of the original problem is detected by convergence, independently from the boundedness/unboundedness of the optimal faces of the original problem.
4. Infeasibility of the original problem is detected by convergence. Primal, dual or primal and dual rays for the original problems are identified to prove dual, primal or dual and primal infeasibility.
5. A perfectly centered initial pair can always be constructed for SSSD.
6. Polynomial convergence with the best known complexity and local quadratic convergence [79, 76] are guaranteed.

Self-dual embedding

To exploit fully the symmetry of the SSSD embedding first the problems (40) and (41) are transformed to the symmetric form. This can be done without increasing the number of variables or the number of constraints. We may assume that in problem (40) $\mathrm{rank}(A) = m$, otherwise the redundant constraints can be eliminated. Let B be any basis of A. Let $A = [B, N]$, $c^T = [c_B^T, c_N^T]$ and $x^T = [x_B^T, x_N^T]$. Then $Ax = b,\ x \geq 0$ can be written as $x_B + B^{-1}Nx_N = B^{-1}b,\ x \geq 0$ or equivalently $-B^{-1}Nx_N \geq -B^{-1}b,\ x_N \geq 0$. Likewise $c^Tx = c_B^Tx_B + c_N^Tx_N = c_B^TB^{-1}b + (c_N^T - c_B^TB^{-1}N)x_N$, hence $(\overline{P})$ can be written equivalently in the symmetric form as

$$\min\left\{(c_N^T - c_B^TB^{-1}N)x_N \mid -B^{-1}Nx_N \geq -B^{-1}b,\ x_N \geq 0\right\}.$$

Note that these transformations need a modified preprocessing of the problem that chooses sparse B and, hopefully, produces sparse $B^{-1}N$.

It is not obvious how such a basis can be constructed and, to the best of our knowledge, this has never been implemented yet. The problem of finding a basis that is optimal in the sense that $B^{-1}N$ is the sparsest possible, imposes some additional requirements to the sparsity problem (see equation (17)). Hence it seems practically more difficult to find efficient heuristics to solve it. Those heuristics should naturally generalize the techniques used to solve the sparsity problem.

Having done the above transformations, we redefine the objective vector, the coefficient matrix and the right hand side vector and denote them again by c, A and b, respectively. This produces the LP problem in the symmetric form

$$(P) \qquad \min\left\{c^Tx \mid Ax \geq b,\ x \geq 0\right\},$$

where A is an $m \times n$ matrix, $c, x \in \mathbb{R}^n$, and $b \in \mathbb{R}^m$. The linear program (P) has associated with it the dual program

$$(D) \qquad \max\left\{b^Ty \mid A^Ty \leq c,\ y \geq 0\right\},$$

As usual, a pair (x^*, y^*) will be called *strictly complementary* if $Ax^* \geq b,\ x^* \geq 0,\ A^Ty^* \leq c,\ y^* \geq 0$, $(Ax^* - b)^Ty^* = (c - A^Ty^*)^Tx^* = 0$ and, moreover, $y^* + (Ax^* - b) > 0$ and $x^* + (c - A^Ty^*) > 0$.

To formulate the SSSD embedding we need some further vectors. Let $x^0, z^0, \bar{c} \in \mathbb{R}^n$, $y^0, r^0, \bar{b} \in \mathbb{R}^m$, $\vartheta^0, \tau^0, \kappa^0, \nu^0 \in \mathbb{R}$ and $\alpha, \beta \in \mathbb{R}$ be given as follows:

$$x^0 > 0,\ \ r^0 > 0,\ \ y^0 > 0,\ \ z^0 > 0,\ \ \vartheta^0 > 0,\ \ \tau^0 > 0,\ \ \kappa^0 > 0,\ \ \nu^0 > 0,$$

$$\bar{b} = \frac{1}{\vartheta^0}(b\tau^0 - Ax^0 + r^0), \qquad \bar{c} = \frac{1}{\vartheta^0}(c\tau^0 - A^Ty^0 - z^0),$$

$$\alpha = \frac{1}{\vartheta^0}(c^Tx^0 - b^Ty^0 + \kappa^0),$$

$$\beta = \alpha\tau^0 + \bar{b}^Ty^0 - \bar{c}^Tx^0 + \nu^0 = \frac{1}{\vartheta^0}[(y^0)^Tr^0 + (x^0)^Tz^0 + \tau^0\kappa^0] + \nu^0 > 0.$$

It is worthwhile to note that if x^0 is feasible for (P), $\tau^0 = 1$ and $r^0 = Ax^0 - b$, then $\bar{b} = 0$. Also if y^0 is feasible for (D), $\tau^0 = 1$ and $z^0 = c - A^Ty$, then $\bar{c} = 0$. With some abuse of mathematics, the vectors $\bar{b}$ and $\bar{c}$ measure the amount of scaled infeasibility of the given vectors x^0, r^0, y^0 and z^0.

Now consider the following skew symmetric self–dual LP problem

$$
\begin{array}{lllllll}
\text{(SSSD)} & \min & & & \beta\vartheta & & \\
 & \text{s.t.} & & Ax & +\bar{b}\vartheta & -b\tau & \geq 0 \\
 & & -A^Ty & & -\bar{c}\vartheta & +c\tau & \geq 0 \\
 & & -\bar{b}^Ty & +\bar{c}^Tx & & -\alpha\tau & \geq -\beta \\
 & & b^Ty & -c^Tx & +\alpha\vartheta & & \geq 0 \\
 & & y \geq 0 & x \geq 0, & \vartheta \geq 0, & \tau \geq 0. &
\end{array}
$$

Due to the selection of the parameters the positive solution $x = x^0,\ y = y^0,\ \vartheta = \vartheta^0,\ \tau = \tau^0$ is interior feasible for the SSSD problem. Let us denote the slack variables for the problem SSSD by r, z, ν and κ respectively. Also note that if one chooses $x = x^0 = e,\ r = r^0 = e,\ y = y^0 = e,\ z = z^0 = e,\ \kappa = \kappa^0 = 1,\ \nu = \nu^0 = 1,\ \vartheta = \vartheta^0 = 1,\ \tau = \tau^0 = 1$, then this solution is a perfectly centered initial solution for the SSSD problem. The following theorem holds (see [79, 38]).

Theorem 1 *For the given problems (P) and (D) the SSSD embedding is made. Then one has:*

(i) The SSSD problem is feasible, hence both primal and dual feasible and has an optimal solution.

(ii) For any optimal solution of SSSD $\vartheta^* = 0$.

(iii) SSSD always has a strictly complementary optimal solution $(x^*, y^*, \vartheta^*, \tau^*)$.

(iv) If $\tau^* > 0$, then $\frac{x^*}{\tau^*}$ and $\frac{y^*}{\tau^*}$ are strictly complementary optimal solutions of (P) and (D), respectively.

(v) If $\tau^* = 0$, then either (P) or (D) or both are infeasible.

The skew–symmetric self–dual embedding has been implemented [77]. Impressive numerical results are reported that show this approach to be only slightly less efficient than the primal–dual method on feasible problems. Additionally, as the experience of the above mentioned paper shows, this method detects problem's infeasibility really by a convergence (not by a divergence). Although it needs slightly different preprocessing and it has to deal with two dense rows and columns bordered to the matrix of the augmented system (that require special care when solving (11)), the SSSD approach may become a computationally attractive alternative to the infeasible primal-dual method.

6.3 Warm start

Many practical problems need the solution of a sequence of similar linear programs where small perturbations are made to b and/or c. As long as these perturbations are small, we naturally expect that the optimal solutions are not far from each other and restarting the optimization from the solution of the old problem (warm start) should be very efficient. This is the case in practice when the simplex method is used. In contrast, we still do not have efficient implementation of warm start in IPMs, that exploits the information contained in the old optimal solution.

Several attempts have been made to solve this problem like using shifted barriers to allow infeasibility of the original variables [24]; applying modified barrier functions [61]; perturbing the problem to throw it away from the boundary of the positive orthant [50]. However, we may say that, in such a general context, IPM warm start is still far from working satisfactorily. On the other hand, computational experience of [8] shows that applying (feasible) projective algorithm [66] to restart from a well centered, almost optimal solution that is sufficiently far away from the boundary, works well even if problems that have to subsequently be solved differ considerably.

The difficulty of the IPM warm start comes from the fact that the old optimal solution is very close to the boundary and well centered. This point in the perturbed problem still remains close to the boundary but is very badly centered. Consequently, IPM makes long sequence of short steps due to the fact that the iterates cannot get rid of the boundary (boundary behavior). Therefore one needs well centered point close to the old optimal one or an efficient centering method (leaving the boundary) to overcome these difficulties. These two possibilities are discussed below.

Independently on the approach chosen it would be wise to save a well centered almost optimal solution (say, with 10^{-2} relative duality gap) that is still sufficiently far away from the boundary.

- **Efficient centering**. The so called target following method offers much flexibility in choosing achievable targets. Path following methods define targets on the central path that, due to boundary behavior, are too optimistic in the case of warm start. Using a sequence of traceable targets that improves centrality allows larger steps, therefore speeds up the centering and, finally, the optimization process. Practical methods of defining such a sequence of targets are not well studied yet although the theory is well established (see, e.g., Jansen, Roos, Terlaky and Vial [39] and Roos and Vial [66] in this volume).

- **Centered solutions for warm start in SSSD embedding**. Among the spectacular properties of the SSSD embedding listed in the previous section, the ability to always construct perfectly centered initial point was mentioned. The old well centered optimal or almost optimal solution x^*, r^*, y^*, z^*, ϑ^*, τ^*, ν^*, κ^*, can be used as x^0, r^0, y^0, z^0, ϑ^0, τ^0, ν^0, κ^0, the initial point to embed the perturbed problem. As we have seen in Section 6.2, $\bar{b}, \bar{c}, \alpha$ and β can always be redefined so that the above solution stays well centered. The construction allows simultaneous perturbations of b and c. Additionally, it extends to handling new constraints or variables added to the problem (e.g., in build–up or cutting plane schemes). In these cases, we can keep the solution unchanged for the old coordinates (let μ be the actual barrier parameter) and define the initial value of the new complementary variables to $\sqrt{\mu}$ as μ was the barrier parameter of the old centered solution. This results in a perfectly centered initial solution.

References

[1] Adler I., Karmarkar, N., Resende, M.G.C. and Veiga, G. (1989) An Implementation of Karmarkar's Algorithm for Linear Programming, *Mathematical Programming* 44, 297–336.

[2] Adler I., Karmarkar, N. Resende, M.G.C. and Veiga, G. (1989) Data Structures and Programming Techniques for the Implementation of Karmarkar's Algorithm, *ORSA Journal on Computing* 1, 84–106.

[3] Adler, I. and Monteiro, R. D. C. (1992) A Geometric View of Parametric Linear Programming, *Algorithmica* 8, 161–176.

[4] Altman, A. and Gondzio, J. (1993) An Efficient Implementation of a Higher Order Primal–dual Interior Point Method for Large Sparse Linear Programs, *Archives of Control Sciences* 2 (XXXVIII) 1/2, 23-40.

[5] Andersen E.D. and Y. Ye (1994) Combining Interior–Point and Pivoting Algorithms for Linear Programming, Research Report, May 1994, Department of Management Studies, The University of Iowa, Iowa City IA, USA.

[6] Arioli, M., Demmel, J.W. and Duff, I.S. (1989) Solving Sparse Linear Systems With Sparse Backward Error, *SIAM Journal on Matrix Analysis and Applications* 10, 165–190.

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

[8] Bahn O., Goffin J.-L., Vial J.-P. and du Merle O. (1994) Experimental behaviour of an interior point cutting plane algorithm for convex programming: an application to geometric programming, *Discrete Applied Mathematics* 49, 3–23.

[9] Balinski, M. L. and Tucker, A. W. (1969) Duality theory of linear programs: A constructive approach with applications. *SIAM Review* 11, 499–581.

[10] Bisseling, R.H., Doup, T.M. and Loyens, L.D.J.C. (1993) A Parallel Interior Point Algorithm for Linear Programs on a Network of Transputers, *Annals of Operations Research* 43, 51–86.

[11] Bixby, R. E. (1994) Progress in Linear Programming, *ORSA Journal on Computing* 6, 15–22.

[12] Brearley, A.L., Mitra, G. and Williams, H.P. (1975) Analysis of Mathematical Programming Problems Prior to Applying the Simplex Algorithm, *Mathematical Programming* 15, 54–83.

[13] Bunch J. R. and Prlett B. N. (1971) Direct Methods for Solving Symemtric Indefinite Systems of Linear Equations, *SIAM Journal on Numerical Analysis* 8, 639–655.

[14] Chang, S.F. and McCormick, S.T. (1992) A Hierarchical Algorithm for Making Sparse Matrices Sparser, *Mathematical Programming* 56, 1–30.

[15] Choi, I.C., Monma, C.L. and Shanno, D.F. (1990) Further Development of a Primal–Dual Interior Point Method, *ORSA Journal on Computing* 2, 304–311.

[16] Dantzig, G.B. (1963) *Linear Programming and Extensions*, Princeton University Press, Princeton N.J.

[17] Dikin, I.I. (1967) Iterative Solution of Problems of Linear and Quadratic Programming, *Doklady Akademii Nauk SSSR* 174, 747–748. Translated in : *Soviet Mathematics Doklady* 8, 674–675.

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

[19] Duff I.S., Gould N.I.M., Reid J.K., Scott J.A. and Turner K. (1991) The Factorization of Sparse Symmetric Indefinite Matrices, *IMA Journal of Numerical Analysis* 11, 181-204.

[20] Duff, I.S. and Reid, J.K. (1983) The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations, *ACM Transactions on Mathematical Software* 9, 302–325.

[21] Fiacco, A.V. and McCormick, G.P. (1968) *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, Wiley, New York.

[22] Fourer, R. and Mehrotra, S. (1993) Solving Symmetric Indefinite Systems in an Interior Point Method for Linear Programming, *Mathematical Programming*, 62, 15–39.

[23] Forrest, J.J.H. and Tomlin, J.A. (1992) Implementing Interior Point Linear Programming Methods in the Optimization Subroutine Library, *IBM Systems Journal* 31, 26–38.

[24] Freund, R. M. (1991) Theoretical Efficiency of a Shifted Barrier Function Algorithm for Linear Programming, *Linear Algebra and Its Applications* 152, 19–41.

[25] George, A. and Liu, J.W.H. (1989) The Evolution of the Minimum Degree Ordering Algorithm, *SIAM Review* 31, 1–19.

[26] Gill, P.E., Murray, W., Saunders, M.A., Tomlin, J.A. and Wright, M.H. (1986) On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method, *Mathematical Programming* 36, 183–209.

[27] Goldman, A. J. and Tucker, A. W. (1956) Theory of Linear Programming, in *Linear Inequalities and Related Systems* (H. W. Kuhn and A. W. Tucker, eds.), Annals of Mathematical Studies, No. 38, Princeton University Press, Princeton, New Jersey, 53–97.

[28] Golub G.H. and Van Loan, C. (1989) *Matrix Computations*, (2nd ed.) The Johns Hopkins University Press, Baltimore and London.

[29] Gondzio, J. (1992) Splitting Dense Columns of Constraint Matrix in Interior Point Methods for Large Scale Linear Programming, *Optimization* 24, 285–297.

[30] Gondzio, J. (1994) Presolve Analysis of Linear Programs Prior to Applying the Interior Point Method, Technical Report 1994.3, Department of Management Studies, University of Geneva, Switzerland.

[31] Gonzaga, C. C. (1992) Path–Following Methods for Linear Programming, *SIAM Review* 34, 167–224.

[32] Greenberg, H. J. (1993) The Use of the Optimal Partition in a Linear Programming Solution for Postoptimal Analysis, Technical Report, Mathematics Department, University of Colorado, to appear in *Operations Research Letters*.

[33] Güler, O., den Hertog, D., Roos, C., Terlaky, T. and Tsuchiya, T. (1993) Degeneracy in Interior Point Methods for Linear Programming: a Survey, *Annals of Operations Research* 46, 107–138.

[34] Güler, O. and Ye, Y. (1993) Convergence Behavior of Some Interior–Point Algorithms, *Mathematical Programming* 60, 215–228.

[35] Hafsteinsson, H., Levkovitz, R. and Mitra, G. (1993) Solving Large Scale Linear Programming Problems Using an Interior Point Method on a Massively Parallel SIMD Computer, in *Applied Parallel Computing*, D.J. Evans, ed., Gordon and Breach Publishers, to appear in Vol.4, No. 3 & 4.

[36] Iri, M. and Imai, H. (1986) A Multiplicative Barrier Function Method for Linear Programming. *Algorithmica* 1, 455–482.

[37] Jansen, B., Roos, C. and Terlaky, T. (1992) An Interior Point Approach to Postoptimal and Parametric Analysis in Linear Programming, Technical Report 92–21, Faculty of Technical Mathematics and Informatics, Technical University Delft, Delft, The Netherlands.

[38] Jansen, B., Roos, C. and Terlaky, T. (1992) The Theory of Linear Programming: Skew–Symmetric Self–Dual Problems and the Central Path, Technical Report 93–15, Faculty of Technical Mathematics and Informatics, Technical University Delft, Delft, The Netherlands, to appear in *Optimization*.

[39] Jansen, B., Roos, C., Terlaky, T. and Vial, J.–P. (1993), Primal–Dual Target Following Algorithm for Linear Programming, Technical Report 93–107, Faculty of Technical Mathematics and Informatics, Technical University Delft, Delft, The Netherlands.

[40] Jong, J. J. de, Jansen, B., Roos, C. and Terlaky, T. (1994) Sensitivity Analysis in Linear Programming: Just be Careful!, Technical Report AMER.93.022, KSLA, Amsterdam, The Netherlands.

[41] Karmarkar, N. K. (1984) A New Polynomial–Time Algorithm for Linear Programming, *Combinatorica* 4, 373–395.

[42] Karmarkar, N. K., Lagarias, J. C., Slutsman, L. and Wang, P. (1989) Power Series Variants of Karmarkar–Type Algorithms, *AT&T Technical Journal*, 68, 20–36.

[43] Kojima, M., Mizuno, S. and Yoshise, A. (1989) A Primal–Dual Interior Point Algorithm for Linear Programming, in N. Megiddo, ed., *Progress in Mathematical Programming: Interior Point and Related Methods*, Springer–Verlag, New York, 29–48.

[44] Kojima, M., Megiddo, N. and Mizuno, S. (1993) A Primal–Dual Infeasible Interior Point Algorithm for Linear Programming, *Mathematical Programming*, 61, 263–280.

[45] Kojima, M., Megiddo, N. and Mizuno, S. (1993) Theoretical Convergence of Large–Step–Primal–Dual Interior Point Algorithms for Linear Programming, *Mathematical Programming*, 59, 1–21.

[46] Kranich, E. (1991) Interior Point Methods for Mathematical Programming: A Bibliography, Discussion Paper 171, Institute of Economy and Operations Research, Fern Universitat Hagen, P.O.Box 940, D-5800 Hagen 1, Germany.

[47] Levkovitz, R., Mitra, G. and Tamiz, M. (1994) Experimental Investigations in Combining Primal Dual Interior Point Method and Simplex Based LP Solvers, under revision for *Annals of Operations Research*.

[48] Lustig, I.J., Marsten, R.E. and Shanno, D.F. (1991) Computational Experience with a Primal–Dual Interior Point Method for Linear Programming, *Linear Algebra and its Applications* 152, 191-222.

[49] Lustig I.J., Marsten, R.E. and Shanno, D.F. (1992) On Implementing Mehrotra's Predictor–Corrector Interior Point Method for Linear Programming, *SIAM Journal on Optimization* 2, 435-449.

[50] 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, 1–14.

[51] Markowitz, H.M. (1957) The Elimination form of the Inverse and Its Application to Linear Programming, *Management Science* 3, 255–269.

[52] McShane, K.A., Monma, C.L. and Shanno, D.F. (1989) An Implementation of a Primal-Dual Interior Point Method for Linear Programming, *ORSA Journal on Computing* 1, 70-89.

[53] Megiddo, N. (1991), On Finding Primal– and Dual–Optimal Bases, *ORSA Journal on Computing* 3, 63–65.

[54] Megiddo, N. (1989) Pathways to the Optimal Set in Linear Programming, in N. Megiddo, ed., *Progress in Mathematical Programming: Interior Point and Related Methods*, Springer–Verlag, New York, 131–158.

[55] Mehrotra, S. (1992) On the Implementation of a Primal–Dual Interior Point Method, *SIAM Journal on Optimization* 2, 575–601.

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

[57] Mehrotra, S. and Monteiro, R. D. C. (1992) Parametric and Range Analysis for Interior Point Methods, Technical Report, April 1992, Dept. of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721, USA.

[58] Mizuno, S. (1992) Polynomiality of Kojima–Megiddo–Mizuno Infeasible Interior Point Algorithm for Linear Programming, Technical Report 1006, School of Operations Research and Industrial Engineering, Cornell University, Ithaca NY.

[59] Mizuno, S., Todd, M. J. and Ye, Y. (1992) On Adaptive–Step Primal–Dual Interior Point Algorithms for Linear Programming, *Mathematics of Operations Research* 18, 964–981.

[60] Monma, C. L. and Morton, A. J. (1987) Computational Experience With a Dual Affine Variant of Karmarkar's Method for Linear Programming, *Operations Research Letters* 6, 261–267.

[61] Polyak, R. (1992) Modified Barrier Functions (Theory and Methods), *Mathematical Programming* 54, 177–222.

[62] Portugal, L., Bastos, F., Judice, J., Paixõ, J. and Terlaky, T. (1993) An Investigation of Interior Point Algorithms for the Linear Transportation Problems, *Report No. 93–100*, Faculteit der Technische Wiskunde en Informatica, Technische Universiteit Delft, Nederlands.

[63] Potra, F. A. (1992) An Infeasible Interior–Point Predictor–Corrector Algorithm for Linear Programming, Technical Report 26, Department of Mathematics, University of Iowa, Iowa City IA, USA.

[64] Reid J.K. (1982) A Sparsity–exploiting Variant of the Bartels–Golub Decomposition for Linear Programming Bases, *Mathematical Programming* 24, 55–69.

[65] Resende, M.G.C. and Veiga, G. (1992) An Efficient Implementation of a Network Interior Point Method, Technical Report February 1992, AT&T Bell Laboratories, Murray Hill, NJ, USA.

[66] Roos, C. and Vial, J.–Ph. (1994) Interior Point Methods, in: *Advances in Linear and Integer Programming*, Beasley, J.E. (ed.), Chapter 3, Oxford University Press, Oxford, England.

[67] Saunders, M.A. (1994) Major Cholesky Would Feel Proud, *ORSA Journal on Computing* 6, 23–27.

[68] Shanno, D. F., Bagchi, A. (1990) A Unified View of Interior Point Methods for Linear Programming, *Annals of Operations Research* 22, 55–70.

[69] Sonnevend, G. (1986) An "Analytic Center" for Polyhedrons and New Classes of Global Algorithms for Linear (Smooth, Convex) Programming, in: *System Modelling and Optimization: Proceedings of the 12th IFIP–Conference*, Prékopa, A., Szelezsán, J. and Strazicky, B. (eds.), Lecture Notes in Control and Information Sciences, Vol. 84, pp. 866–876, Springer Verlag, Berlin, Germany.

[70] Stewart, G. W. (1989) On Scaled Projections and Pseudoinverses., *Linear Algebra and Its Applications* 112, 189–193.

[71] Tinney, W.F., Walker, J.W. (1967) Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization, *Proceedings of IEEE* 55, 1801–1809.

[72] Turner K. (1991) Computing Projections for the Karmarkar Algorithm, *Linear Algebra and its Applications* 152, 141-154.

[73] Vanderbei, R.J. (1991) Splitting Dense Columns in Sparse Linear Systems, *Linear Algebra and its Applications* 152, 107–117.

[74] Vanderbei R. and Carpenter T.J. (1991) Symmetric Indefinite Systems for Interior Point Methods, Technical Report SOR 91-7, Department of Civil Engineering and Operations Research, Princeton University, Princeton, New Jersey, 1991.

[75] Ward, J. E., Wendell, R. E. (1990) Approaches to Sensitivity Analysis in Linear Programming. *Annals of Operations Research* 27, 3–38.

[76] Wu, F., Wu, S. and Ye, Y. (1992) On Quadratic Convergence of the $\mathcal{O}(\sqrt{n}L)$–Iteration Homogeneous and Self–Dual Linear Programming Algorithm, Technical Report, Department of Management Studies, The University of Iowa, Iowa City IA, USA.

[77] Xu, X., Hung, P.-F. and Ye, Y. (1993) A Simplified Homogeneous and Self-Dual Linear Programming Algorithm and its Implementation, Technical Report, Department of Management Sciences, The University of Iowa, USA.

[78] Yannakakis, M. (1981) Computing the minimum fill–in is NP–complete, *SIAM Journal on Algebraic Discrete Methods* 2, 77–79.

[79] Ye, Y., Todd, M. J. and Mizuno, S. (1994) An $\mathcal{O}(\sqrt{n}L)$–Iteration Homogeneous and Self–Dual Linear Programming Algorithm, *Mathematics of Operations Research* 19, 53–67.

[80] Zhang, Y. (1992) On the Convergence of an Infeasible Interior–Point Algorithm for Linear Programming and Other Problems, Research Report, 92–07, Dept. of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21228, USA.