Solution of linear and convex optimization problems with emphasis on interior-point methods

Erling Dalgaard Andersen

January 17,1996

Kapitel 1

Preface

The present dissertation is the result of my three years of Ph.D.- study. The dissertation contains five papers treating the solution of linear and convex optimization problems. Linear and convex optimization problems are used to describe many different decision problems arising in operations research and economics. Therefore, methods for solution of optimization problems are of interest in many areas of economics. The optimization problems can be large when they model the decision problem accurately where size of the problem means number of constraints and variables in the problem. Hence, if larger optimization problems can be solved, then better decisions can be made. Therefore, it is important to be able to solve large-scale optimization problems. We define a problem to be large-scale if the problem cannot or only hardly can be solved with the existing methods and computers due to its size. Clearly this is a weak definition of large-scale. However, in the fifties a 100 variable LP problem was large-scale whereas today an LP problem containing less than 100,000 constraints and variables cannot be considered large scale. This implies, it is impossible to give a stronger definition of large-scale.

The above observations motivate the underlying theme for this dissertation which is solution of large-scale linear and convex optimization problems. Recently, it has been proposed to use interior-point methods to solve this class of problems and it have been proved that the methods has good theoretical and practical properties. Therefore, each chapter of this dissertation treats different subjects related to interior-point methods and their application.

Now I will give a short introduction to each chapter in the dissertation. The titles of the chapters are:

2. A survey of interior-point methods.

3. Presolving in linear programming (jointly with Knud D. Andersen, presented at Scandinavian Workshop on Linear Programming August 1993 and at TIMS/ORSA fall conference 1993, see [7]).

4 Finding all linearly dependent rows in large-scale linear programming (see [6]).

5. Combining interior-point and pivoting algorithms for linear programming, (jointly with Y. Ye, presented at 15th International Symposium on Mathematical Programming 1994, to appear in Management Science).

6. On a homogeneous algorithm for the monotone complementarity problem (jointly with Y. Ye, presented at INFORMS fall 1995 conference and at International Conference on Complementarity Problems 1995 conference (Baltimore)).

The first paper surveys the last ten years of development in interior-point methods for linear and convex optimization problems. We also present two examples of optimization problems arising in operations research and finance to motivate the subject for this dissertation. The main purpose of the paper is to present a context for the following four papers.

The subject for the second paper is automatic reduction techniques for large-scale linear programming (LP) problems. Most large-scale LP problems contain a lot of simple redundancies and therefore it is worthwhile to remove such redundancies before submitting the problems to optimization. The chapter gives a comprehensive survey of existing methods and proposes some improvements. Also the so-called postsolve phase that is reconstructing the solution to the original problem from the solution to the reduced problem is discussed in detail. Finally numerical results are presented showing it is worthwhile to use automatic reduction techniques in combination with an interior-point algorithm.

The third paper is an extension of the work in the second paper. In an interior-point method there is a certain linear equation system solved in each iteration. It has a unique solution if there are no linearly dependent rows in the LP constraint matrix. It is therefore desirable to remove all linear dependencies. In the paper we discuss how to detect and remove all linear dependencies efficiently.

The fourth paper discuss how to combine an interior-point algorithm and a pivoting algorithm such as the simplex algorithm. In general, if an LP problem has multiple optimal solutions an interior-point algorithm does not generate an optimal solution which is also a basic solution. However, in practice a basic solution is often needed, because it facilitates a sensitivity analysis and obtains a good starting solution for perturbed problems. We present a method which purifies the interior-point solution into a basic solution. By exploiting properties of the interior-point algorithms we obtain a method that has good theoretical properties and also works well in practice. The last claim is documented with numerical results.

Finally, the fifth paper discusses solution of the monotone complementarity problem. This problem contains the class of all convex optimization problems. We propose an algorithm which in one phase detects whether the problem is infeasible or reports the optimal solution without using any regularization or "big-M" parameter. Moreover, assuming some smoothness conditions the algorithm is polynomial. We also present some numerical results suggesting the proposed algorithm works well in practice.

During my Ph.D.-study I spend two years at Department of Management Sciences, The University of Iowa, USA (summer 1993-summer 1995). I have benefitted a lot from the stay and I would like to express my gratitude towards all those who made it possible. I am grateful for the financial support I received from the Danish Research Academy and from The University of Iowa. Professor Kenneth Kortanek and his wife Irene Kortanek made me feel especially welcome in Iowa City. Irene got me the nicest apartment and filled my refrigerator before I came. I learned a lot from my work with Professor Yinyu Ye. Yinyu's supervision and cooperation with me has improved this dissertation significantly. I also enjoyed all the good discussions about optimization with my office mates Xiaojie Xu and Marcia Fampa.

I also appreciate the support I received from my Danish advisors Niels Christian Petersen and Ole Bent Olesen.

During the last 5 years I have had numerous discussions about optimization with my brother Knud D. Andersen. These discussions have been very stimulating for my work and without them I might never have started a Ph.D.-study and written this dissertation.

Indhold

Kapitel 2

A survey of interior-point methods

This paper surveys the development of interior-point methods since Karmarkar's now famous 1984 paper. Our emphasis will be on methods for linear programming, but we also discuss the methods' extension to nonlinear problems.

2.1 Introduction

Linear programming (LP) and mathematical programming in general are important tools in operations research and in many other areas of science. A few examples of applications of mathematical programming in operations research are transportation planning, portfolio selection and productivity analysis. In practice, these applications give rise to very large LP problems. Hence, it is essential to have methods which solve large problems efficiently.

The first efficient method for LPs was invented by G. B. Dantzig in the late forties, see Dantzig [37]. The method is known as the simplex algorithm and to this day the majority of all LP problems are solved using the simplex algorithm.

From a theoretical point of view, the simplex algorithm is not considered efficient, because it may, in the worst case perform a work that grows exponentially with the size of the problem. Whereas for an efficient algorithm the work only grows polynomially with the size of the problem. Even though the simplex algorithm is not polynomial ample practical evidence shows that the simplex algorithm solves most LP problems in a small number of iterations.

The first polynomial algorithm for linear programming was presented by Khachian [83] in 1979. Unfortunately, Khachian's ellipsoid algorithm is not efficient in practice and cannot compete with the simplex algorithm at all.

In 1984, Karmarkar [82] presented a new polynomial algorithm for LP. It was claimed that this algorithm was much more efficient than the simplex algorithm but it has never been scientifically documented. Nevertheless, Adler et al. [1] presents a variant of Karmarkar's algorithm that performs well in practice and since then Karmarkar's paper has generated a lot of interest and research which has produced genuine advances in the state of the art.

The main idea in Karmarkar's algorithm is to approach the optimal solution of the LP problem from the interior of the feasible region, which is the opposite strategy of the simplex algorithm which moves along the boundary of the feasible region. The interior-point approach was already an old idea in 1984. It was, for example, proposed by Frisch [54] in 1955. Indeed it has been formally shown by Gill et al. [59] that Frisch's logarithmic barrier algorithm is closely related to Karmarkar's algorithm. The interior-point approach was also studied extensively in the sixties by Fiacco and McCormick [48]. Other earlier work related to Karmarkar's algorithm is Dikin's [41, 42] affine scaling algorithm and Huard's center method [75].

Over the last 10 years, many papers have been written about interior-point methods. A lot of them are surveyed and referenced in the excellent articles by Goldfarb and Todd [61], Gonzaga [67] and Wright [147]. See also the book of Ye [154].

It is easy to extend interior-point methods to general convex nonlinear problems such as quadratic programs. This possibility is studied in detail in the book of Nesterov and Nemirovskii [117] and in the thesis of den Hertog [38].

The outline of the paper is as follows. First we present two examples of optimization problems to motivate our work. Next we present our notation and some relevant LP theory. Then we survey barrier methods for linear programs. Our emphasis is on the so-called primal-dual algorithm. Finally, we discuss the algorithms extension to nonlinear problems.

2.2 Some examples

In this section we present two examples of optimization problems arising in operations research and economics to motivate the importance of being able to solve large-scale optimization problems.

2.2.1 A linear model

In this section we will present the so-called airline crew scheduling problem, see the papers [57, 55, 74, 129] for details. The problem can be described as follows. Given a flight schedule and some governmental and labor work rules, what is the cheapest assignment of crews to flight segments? A flight segment is a nonstop trip from one airport to another airport.

Each crew is located at a base, for example Chicago, and each crew must be assigned to a flight rotation. A flight rotation is a set of flight segments such that the crew returns to its base. Two possible rotations are [(Chicago,Denver), (Denver,Phoenix), (Phoenix,Chicago)] or [(Chicago,Los Angeles), (Los Angeles,Chicago)]. In practice, there is a huge number of possible rotations for each crew. Although only rotations which satisfy work rules are feasible. To each feasible rotation a cost can be assigned, which is the incremental cost using the rotation.

In other words, the problem is to pick a set of rotations such that all flight segments are covered exactly once and the total cost is minimized.

To give an idea of the size and the importance of the problem it can be mentioned that even for medium sized problems there are millions of feasible rotations. The costs for the flying personnel for the major US carriers often exceed 1.3 billion\$, see [74]. Therefore even a small percentage savings is worthwhile.

In mathematical terms the problem can be stated as follows

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = e, \\ & x \in \{0,1\}^n, \end{array} \tag{2.1}$$

where $c \in R^n$, $A \in R^{m \times n}$, $e \in R^m$ and $x \in R^n$. Hence, m is the number of flight segments and n is the number of rotations. c_j is the cost of the jth rotation. a_{ij} is equal to 1 if the jth rotation covers the ith flight segment and otherwise 0. Finally, e is the vector of all ones that is $e = (1, \dots, 1)^T$.

In words the model minimizes the total cost such that all flight segments are covered exactly once.

It is important to note that the problem (2.1) is only an approximation to the real problem. The reason is that in real problems the number of feasible rotations are huge. Therefore, the rotations included in the problem are only a representative subset of all the feasible rotations. This subset is chosen using heuristics and random selection. In general, from a cost minimizing point of view, the more rotations that are included in the problem the better.

Even though only a representative subset of the feasible rotations is generated the number of variables in the problem may be large. In one reported case a problem with approximately 13 million variables was generated. However, the number of rows is usually small, say less than 2000.

The first step in solving the problem (2.1) is to relax the integer restrictions on the variables. The relaxed problem is an LP problem and it has the following features:

1. The problem is huge.
2. The problem is primal degenerate and the primal simplex algorithm has difficulties in solving the problem, see [22].
3. In practice, the problems are generated such that they contain a large number of duplicate columns. Duplicate columns are identical except for the cost coefficient. All of these duplicate columns, except

the one with the lowest cost coefficient can be removed from the problem. There are also some linearly dependent rows in the problem.

4. The solution strategies for a integer programming problem usually requires an optimal solution to the LP relaxation which is also an optimal basic solution.

These observations motivate study of methods for detection and removal of simple redundancies in LP problems.

Also alternatives to the primal simplex algorithm are needed to solve this class of problems. It is now accepted that interior-point methods are efficient for solution of large-scale problems. Furthermore, interior-point methods do not have problems with degeneracies. Hence, interior-point methods may make it possible to solve this class of problems more efficiently.

However, interior-point methods do not in general, produce an optimal solution which is also a basic solution. It follows that it is important to be able to generate an optimal basic solution from any solution to an LP problem.

These subjects are treated in this dissertation.

2.2.2 A nonlinear model

In economics, a linear model of the problem under study is often a sufficiently good approximation. However, in some cases a nonlinear model is required. In this section we will present one such case.

A classical problem in finance is to select a portfolio from a set of securities assumed to be risky. Inherently in this problem is a trade-off between the expected return and the risk. In general, there is a positive correlation between those two. Therefore, a portfolio should be selected such that given a expected return the risk is minimized.

Assume we have n securities and r is the vector of expected returns of the securities. Also assume the n by n variance-covariance C matrix of the expected returns is known. Typically, the vector r and the matrix C are estimated from historical data. Now we can state the problem as follows:

$$\begin{array}{ll} \text{minimize} & x^T C x \\ \text{subject to} & r^T x \geq \lambda, \\ & e^T x = 1, \\ & x \geq 0, \end{array} \tag{2.2}$$

where x is an n vector and denotes the proportion that is invested in each security. The parameter λ is the desired expected return. Hence, the model picks a portfolio such that its variance of the expected return is minimized subject to an expected return greater than or equal to λ is achieved. Ideally, the solution denoted $x(\lambda)$ for all values of λ is required. The curve $x(\lambda)$ is called the efficient frontier and it shows the Pareto optimal relation between risk and expected return. It implies that the problem (2.2) has to be solved for different values of λ.

The problem (2.2) is a convex quadratic optimization and it can easily contain from a few hundred to thousand variables.

The above model can be extended in several directions. In practice the model is too simple. Often the investor has constraints which require at least 20% of the budget should be invested in the oil industry and in the financial sector. Such restrictions lead to inclusion of general linear constraints.

The model presented above is a single period model and an obvious extension is to work with a multiperiod model. Clearly, a multiperiod model leads to a more realistic description of the decision problem and therefore to better decisions (more profitable).

Note that a k period model, usually, is at least k times larger than a single period model. Therefore, multiperiod models may be very large.

One possible multiperiod model arising in asset allocation is discussed by Carpenter et al. [31]. Their problem is to maximize a general concave function (investor's utility function) subject to general linear constraints. Solution of problems with up to 8000 constraints and 20000 variables using an interior-point algorithm and MINOS are reported. MINOS is based on active set ideas similar to the simplex algorithm and is considered one of the best algorithms for nonlinear problem. The results presented show that the interior-point algorithm outperforms MINOS on the larger testproblems[1]. Hence, the new interior-point methods make it possible to solve (some) nonlinear problems faster than before. It implies that it might be possible to solve larger problems than previously.

In finance, this has the implication that improved decisions can be made with respect to selecting a portfolio of securities.

2.3 Notation and LP theory

In this section, we will present our notation and some relevant theory for LP problems. Our notation follows that of [154].

Any LP problem can be written in the following standard form

$$\begin{array}{lll} (LPP) & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b, \\ & & x \geq 0, \end{array} \tag{2.3}$$

where $c, x \in R^n, b \in R^m$ and $A \in R^{m \times n}$. We assume A is of full row rank. The primal feasible region is defined by

$$\mathcal{F}_P = \{x \in R^n_+ : Ax = b\} \tag{2.4}$$

and its interior by

$$\dot{\mathcal{F}_P} = \{x \in R^n_{++} : Ax = b\}, \tag{2.5}$$

where $R_+ = \{x \in R : x \geq 0\}$ and $R_{++} = \{x \in R : x > 0\}$.

The dual problem corresponding to (LPP) is

$$\begin{array}{lll} (LPD) & \text{maximize} & b^T y \\ & \text{subject to} & A^T y + s = c, \\ & & s \geq 0, \end{array} \tag{2.6}$$

where $y \in R^m$ and $s \in R^n$. Similar to (2.4) and (2.5) we define

$$\mathcal{F}_D = \{(y, s) \in R^m \times R^n_+ : A^T y + s = c\} \tag{2.7}$$

and

$$\dot{\mathcal{F}_D} = \{(y, s) \in R^m \times R^n_{++} : A^T y + s = c\}. \tag{2.8}$$

We also use the notation that

$$\mathcal{F} = \mathcal{F}_P \times \mathcal{F}_D \text{ and } \dot{\mathcal{F}} = \dot{\mathcal{F}_P} \times \dot{\mathcal{F}_D}. \tag{2.9}$$

Now we can state the following two duality theorems.

Theorem 2.3.1 *(Weak duality.) Let $(x, y, s) \in \mathcal{F}$ then*

$$c^T x \geq b^T y.$$

Theorem 2.3.2 *(Strong duality.) Let $x \in \mathcal{F}_P$ then x is an optimal solution if and only if there exists $(y, s) \in \mathcal{F}_D$ such that*

$$c^T x - b^T y = x^T s = 0.$$

The proof of these two theorems can be found in any book about linear programming.

The Karush-Kuhn-Tucker (KKT) optimality conditions to (LPP) are

$$\begin{array}{rcll} Ax - b & = & 0, & x \geq 0, \\ A^T y + s - c & = & 0, & s \geq 0, \\ Xs & = & 0. & \end{array} \tag{2.10}$$

[1] Carpenter et al. [31] use an SQP algorithm. Hence, their solution algorithm is based on solving a sequence of quadratic programming (QP) problems and each QP problem is solved with an interior-point algorithm. An interior-point algorithm based on solving the problem directly might be significantly faster (3 to 5 times).

Here we introduce the new notation that if x is vector then capital X denotes

$$X = diag(x).$$

The KKT conditions state that an optimal solution (x, y, s) must be primal and dual feasible and satisfy the complementarity conditions. Hence it follows from Theorem 2.3.2 that these optimality conditions are necessary and sufficient.

We denote the duality gap

$$c^T x - b^T y \tag{2.11}$$

and the complementarity gap

$$x^T s. \tag{2.12}$$

If (x, y, s) is primal and dual feasible, then the duality and complementarity gap is identical.

We say a solution (x, y, s) to (2.10) is a strictly complementarity solution if

$$x_j + s_j > 0, \ \forall j \in \{1, \ldots, n\}$$

that is either x_j or s_j is zero but not both. All LP problems have at least one strictly complementarity solution, see Goldman and Tucker [62] and Jansen et al. [78]. Now let (x^*, y^*, s^*) be any strictly complementarity solution to (LPP), then define

$$\mathcal{P} = \{j : x_j^* > 0\} \text{ and } \bar{\mathcal{P}} = \{1, \ldots, n\} \setminus \mathcal{P}.$$

It can be shown that $\mathcal{P}$ is invariant with respect to any strictly complementarity solution.

2.4 The logarithmic barrier method

Most of the new interior-point methods can be seen as variants of the logarithmic barrier method due to Frisch [54]. This has been shown by Roos and Vial [125] and Shanno and Bagchi [127]. Therefore, we discuss barrier methods in some details.

The principle of the barrier method is to solve the problem

$$\begin{array}{lll} (LPPB) & \text{minimize} & c^T x - \mu \sum_{j=1}^n \ln(x_j) \\ & \text{subject to} & Ax = b. \end{array} \tag{2.13}$$

for a positive decreasing sequence of the barrier parameter μ instead of solving (LPP) directly. The solution to (2.13) for one value of μ is used to find the solution to (2.13) for the next value of μ faster than starting from scratch. The logarithmic barrier term in the objective function acts as a barrier and to insure that any solution to $(LPPB)$ is positive. In other words, any feasible solution to $(LPPB)$ is also feasible to (LPP). Let $x(\mu)$ be the optimal solution to $(LPPB)$ (if it exists[2]) for a given μ, then $x(\mu)$ will converge towards the optimal solution to (LPP) when $\mu \to 0^+$. Hence, it is natural to develop an algorithm that traces $x(\mu)$ to the optimal solution. The advantage of this approach is that it is not necessary to guess the optimal active set (=correct basic and non-basic partition) which is a difficult combinatorial problem. This is exactly what the simplex algorithm tries to do.

Since $(LPPB)$ is a convex optimization problem, the Karush-Kuhn-Tucker (KKT) conditions

$$\begin{array}{rcl} c - \mu X^{-1} e - A^T y & = & 0, \\ Ax - b & = & 0, \quad x > 0, \end{array} \tag{2.14}$$

are necessary and sufficient optimality conditions. Note that e denotes the vector of all ones that is $e = (1, \ldots, 1)^T$.

Assume (x, y) is a solution to (2.14) for some $\mu > 0$. Since

$$\mu X^{-1} e = c - A^T y = s > 0$$

then $(y, \mu X^{-1} e)$ is a dual feasible solution to (LPD). Moreover, the duality gap is given by

$$c^T x - b^T y = x^T s = \mu x^T X^{-1} e = n\mu. \tag{2.15}$$

[2] A sufficient condition is $\dot{\mathcal{F}} \neq \emptyset$.

Therefore, any solution to (2.14) implicitly defines a dual feasible solution with a known duality gap given by $n\mu$. It proves that $x(\mu)$ converges to the optimal solution as $\mu \rightarrow 0^+$. Before we turn to the construction of a barrier algorithm we discuss the dual and primal-dual formulation of the system (2.14).

Instead of applying the barrier transform to (LPP) we can apply it to the dual problem (LPD). In this case we obtain the KKT conditions

$$\begin{array}{rcl} Ax - b & = & 0, \\ A^T y + s - c & = & 0, \quad s > 0, \\ \mu S^{-1} e - x & = & 0. \end{array} \tag{2.16}$$

Clearly the equations are almost identical to (2.14) and an analysis of them is similar to the primal case.

If we combine the equations (2.14) and (2.16) we obtain the primal-dual equations

$$\begin{array}{rcl} A^T y + s - c & = & 0, \quad s > 0, \\ Ax - b & = & 0, \quad x > 0, \\ Xs - \mu e & = & 0. \end{array} \tag{2.17}$$

This set of nonlinear equations are identical to the KKT conditions (2.10) to (LPP) except that the complementarity conditions are perturbed by μ.

The three sets of equations (2.14), (2.16) and (2.17) are identical, that is they have the same solutions. However, informally speaking the system (2.17) is less nonlinear than the systems (2.14) and (2.16). Therefore, it may be easier to solve (2.17) using for example Newton's method than the two other systems. Hence, an algorithm based on solving (2.17), is likely to be more efficient, than a method based on solving the two other systems.

2.4.1 The central path

Given the assumption $\dot{\mathcal{F}} \neq \emptyset$, that is, both the interior of both the primal and dual feasible region are nonempty the nonlinear equations (2.17) have a solution for all non-negative μ. Therefore, the equations (2.17) define a smooth curve called the central path. The central path plays a fundamental role for the development of many interior-point algorithms. It is formally defined as follows

$$\Gamma = \{(x, y, s) \in \dot{\mathcal{F}} : Xs = \frac{x^T s}{n} e\}.$$

We can now state the following theorem adapted from Ye [154].

Theorem 2.4.1 *Let $(x(\mu), y(\mu), s(\mu)) \in \Gamma$, then*

1. *The central pair $(x(\mu), s(\mu))$ is bounded where $0 < \mu \leq \mu^0$ for any given $\mu^0 > 0$.*
2. *For $0 < \mu' \leq \mu$,*
$$c^T x(\mu') \leq c^T x(\mu) \text{ and } b^T y(\mu') \geq b^T y(\mu).$$
3. *$(x(\mu), y(\mu), s(\mu))$ converges to an optimal solution pair for (LPP) and (LPD).*

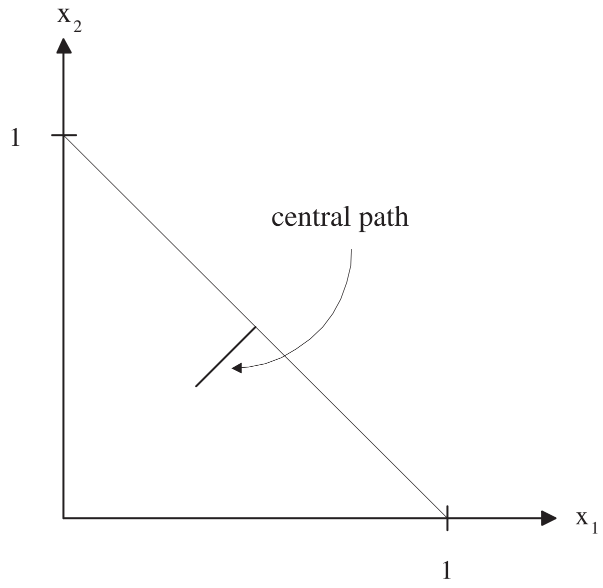
Proof: See [154].

□

The theorem shows that the central path is well defined and its limit point is an optimal solution to the problem (LPP). Now we use the example (2.18) to demonstrate some properties of the central path.

$$\begin{array}{ll} \text{minimize} & -x_1 - x_2 \\ \text{subject to} & x_1 + x_2 \leq 1, \\ & x_1, x_2 \geq 0. \end{array} \tag{2.18}$$

In Figure 2.1 we have shown the feasible region for the example (2.18). The central path is also shown. Note that we have not included the slack variable for the first inequality in (2.18). The following can be observed.



Figur 2.1: Example 2.18 shown graphically.

1. The central path is a straight line from the point (1/3,1/3) to the point (1/2,1/2). The point (1/3,1/3) is called the center. As μ tends to infinity $x(\mu)$ tends to the center. On the other hand as μ tends to zero $x(\mu)$ tends to (1/2,1/2). The point (1/2,1/2) is an optimal solution and is in the center of the optimal face. The optimal face consists of the line segment from (0,1) and to (1,0).
2. The central path is only a straight line because we have not included the slack variable in the figure. Moreover, in general, the central path is not a straight line.
3. By following the central path we stay in the center of the feasible region. Hence from points close to the central path it tends to be possible to take long steps in any direction. Therefore, an algorithm that follows the central path is not likely to get stuck at a point close to the boundary.
4. The example demonstrates that if the central path is followed to an optimal point, then a basic solution is not produced. Note only the points (0,1) and (1,0) correspond to optimal basic solutions.
5. If redundant inequalities are added to the example, then the location of the central path changes. This is an undesirable property.

These observations combined with Lemma 2.4.1 lead to construction of algorithms that generate points on the central path converging towards the optimal solution.

Unfortunately, it is more difficult to solve the system of nonlinear equations (2.17) than to solve the problem (LPP). Therefore, interior-point algorithms only generate points close to the central path in some measure. This means that the system of nonlinear equations (2.17) only has to be solved approximately. Another way to describe this solution approach is that the algorithms generate points in the neighborhood of the central path converging towards the optimal solution. Some common used neighborhoods are

$$\mathcal{N}_2(\mu,\beta) = \{(x,y,s) \in \dot{\mathcal{F}} : \|Xs - \mu e\|_2 \le \beta\mu\},$$

$$\mathcal{N}_\infty(\mu,\beta) = \{(x,y,s) \in \dot{\mathcal{F}} : \|Xs - \mu e\|_\infty \le \beta\mu\}$$

and

$$\mathcal{N}_\infty^-(\mu,\beta) = \{(x,y,s) \in \dot{\mathcal{F}} : x_j s_j \ge \beta\mu, \forall j \in 1,\ldots,n\}.$$

We have the following relationship

$$\mathcal{N}_2(\beta) \subseteq \mathcal{N}_\infty(\beta) \subseteq \mathcal{N}_\infty^-(\beta).$$

The central path is studied by for example Megiddo [103] and Sonnevend [128].

2.4.2 The primal barrier algorithm

As already mentioned Karmarkar's algorithm can be seen as a variant of the primal barrier algorithm. This has formally been proved by Gill et al. [59].

The principle of the primal barrier algorithm is as follows. First an approximate solution to (2.14) is chosen. This solution is moved closer to the central path using one or more iterations of Newton's method applied to (2.14). Afterwards, the barrier parameter is reduced and the whole process is repeated. The algorithm can be stated as follows.

Algorithm 2.4.1

1. *Pick* $x^0 \in \dot{\mathcal{F}}_P,\ y = 0$ *and* $\mu^0 \geq 0$
2. $k := 0$
3. *while stopping criteria not satisfied*
4. *Solve:*
$$\begin{bmatrix} \mu^k (X^k)^{-2} & -A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} -(c - \mu^k (X^k)^{-1} - A^T y^k) \\ 0 \end{bmatrix}$$
5. *Pick* $\alpha \in [0,1]$ *such that* $x^{k+1} := x^k + \alpha \delta x > 0$
6. $y^{k+1} := y^k + \delta y$
7. *For some* $\gamma \in [0,1]$ *let* $\mu^{k+1} := \gamma \mu^k$
8. $k := k + 1$
9. *end while*

The linear equations solved in step 4 are exactly equal to one iteration of Newton's method applied to (2.14). It can be verified if A is of full row rank, then the Newton equations system has a unique solution.

The choice of α and γ in step 5 and 7, respectively is important for the convergence of the algorithm. Gonzaga [66] shows that a certain choice implies the algorithm converge in $O(\sqrt{n}L)$ iterations assuming the initial point (x^0, y^0) is close to the central path. (L is the bit size of the problem).

Note that it is possible to pick μ equal to zero in all iterations. In this case, the barrier algorithm is identical to the primal affine scaling algorithm proposed by Dikin [41] and later rediscovered by several researchers, see Barnes [19] and Vanderbei, Freeman and Meketon [143].

An implementation of the primal barrier algorithm is presented by Gill et al. [59] and Tomlin [134].

2.4.3 The dual barrier algorithm

If we apply the primal barrier algorithm to the dual problem (LPD), then we obtain the dual barrier algorithm. This algorithm can be shown to have the same complexity as the primal variant.

The dual barrier algorithm has been implemented by Gill, Murray and Saunders [58]. Especially the affine scaling version of the dual barrier algorithm has been very popular, see for example [1, 2, 12, 14, 99, 112].

Another dual interior-point algorithm is presented by Boggs et al. [24, 43].

2.4.4 The primal-dual algorithm

In practice, the primal and dual barrier algorithm presented above is outperformed by the primal-dual algorithm. This algorithm was proposed by Kojima, Mizuno and Yoshise [87]. It is almost identical to the primal barrier algorithm, but instead of solving the nonlinear equations (2.14) the equations (2.17) are solved. Kojima et al. assumes that the primal-dual algorithm is initiated with a primal and dual feasible solution. Later, Lustig [92] proposes a primal-dual variant that relaxes this assumption. The resulting algorithm is known as the primal-dual infeasible-interior-point algorithm, but we call it the primal-dual algorithm for short.

One iteration of the primal-dual algorithm can be derived as follows. Assume an initial point (x^0, y^0, s^0) is known such that

$$(x^0, y^0, s^0) \in R^n_{++} \times R^m \times R^n_{++}. \tag{2.19}$$

It is not assumed that the initial solution is primal and dual feasible. Hence the primal residuals

$$r_P^0 = b - Ax^0$$

and the dual residuals

$$r_D^0 = c - A^T y^0 - s^0$$

need not be zero. The search direction used by the primal-dual algorithm is identical to one iteration of Newton's method applied to (2.17) that is

$$\begin{array}{rcl} A^T \delta y + \delta s & = & r_D^0, \\ A \delta x & = & r_P^0, \\ S^0 \delta x + X^0 \delta s & = & -X^0 s^0 + \gamma \mu^0 e, \end{array} \tag{2.20}$$

where

$$\mu^0 = \frac{(x^0)^T s^0}{n}.$$

$\gamma \in [0, 1]$ is a suitable chosen parameter. The new point is defined by

$$(x^1, y^1, s^1) = (x^0, y^0, s^0) + \alpha(\delta x, \delta y, \delta s), \tag{2.21}$$

where $\alpha \in [0, 1]$ is a suitable chosen stepsize such that $(x^1, s^1) > 0$. Convergence of the algorithm essentially follows from the following lemma.

Lemma 2.4.1 *Let (x^1, y^1, s^1) be defined by (2.21), then*

1. $r_P^1 = (1 - \alpha) r_P^0$.
2. $r_D^1 = (1 - \alpha) r_D^0$.
3. $(x^1)^T s^1 = (1 - \alpha(1 - \gamma))(x^0)^T (s^0) + \alpha^2 (\delta x)^T \delta s$.

Proof: See for example Kojima et al. [84].

□

Lemma 2.4.1 shows that the primal and dual residuals are reduced linearly in the stepsize α. Moreover, if $\gamma \in [0, 1)$, then the complementarity gap is also reduced for a sufficiently small α. The algorithm keeps iterating until the stopping criterion

$$\left\| r_P^k \right\| \leq \epsilon_P, \ \left\| r_D^k \right\| \leq \epsilon_D \text{ and } x^T s \leq \epsilon_C \tag{2.22}$$

is satisfied for some preselected $\epsilon_P, \epsilon_D, \epsilon_C > 0$.

Note that it is possible to choose γ equal to zero in all iterations. In this case, the primal-dual algorithm is identical to the primal-dual affine scaling algorithm.

From Lemma 2.4.1 it follows that the primal-dual algorithm is globally convergent if it is possible to choose the stepsize strictly positive in all iterations such that the complementarity gap is reduced. This is not possible without the introduction of some safeguards on the choice of stepsize. Such safeguards have been proposed by Kojima, Mizuno and Megiddo [84] (see also Mizuno [110]). The safeguards are that all iterates should remain in the neighborhood

$$\begin{array}{rl} \mathcal{N}(\beta) = \{ & (x, y, s) \in R^n_{++} \times R^m \times R^n_{++} : \\ & \left\| Ax^0 - b \right\| \frac{x^T s}{(x^0)^T s^0} \geq \left\| Ax - b \right\|, \\ & \left\| A^T y^0 + s^0 - c \right\| \frac{x^T s}{(x^0)^T s^0} \geq \left\| A^T y + s - c \right\|, \\ & x_j s_j \geq \beta (x^T s)/n, \forall j = 1, \ldots, n \}, \end{array} \tag{2.23}$$

where $\beta \in (0, 1)$. The first two conditions in the neighborhood definition force the infeasibility to be reduced faster than the complementarity gap. If the opposite is the case, we might end up with a complementarity solution that is infeasible (highly undesirable). Moreover, the last condition keeps the iterates close to the central path. How close depends on the value of β. Hence, the initial point and β

is chosen such that $(x^0, y^0, s^0) \in \mathcal{N}(\beta)$ and in all the iterations the stepsize α are chosen such that the iterates belong to the neighborhood (2.23). In this case, Kojima et al. [85] show that a strictly positive stepsize always exists which reduces the complementarity gap. Therefore, the primal-dual algorithm with safeguards is globally convergent.

One advantage of letting β be positive is that it implies that the optimal solution will be a strictly complementarity solution, see Güler and Ye [70]. A strictly complementarity solution is sometimes very useful, see [68] for some examples. One example Greenberg mentions is to identify critical jobs in a project scheduling problem. Only if a strict complementarity solution is known, the correct critical jobs can be selected.

Furthermore, it is well-known that the traditional sensitivity analysis for LP problems might lead to wrong conclusions if the LP problem is degenerate. Jansen et al. [77] study an alternative sensitivity analysis which requires knowledge of the optimal partition. They show their sensitivity analysis leads to more informative results than the classical analysis based on the basic and non-basic partition.

If the primal and dual solution is feasible, then

$$\delta x^T \delta s = 0.$$

In this case, the complementarity gap is reduced linearly in the stepsize, and the convergence analysis is simplified.

We have implicitly assumed that the LP problem is primal and dual feasible ($\mathcal{F} \neq \emptyset$). This is, of course, not always the case. Therefore, Kojima et al. [84] also terminate the algorithm if

$$\|(x, y, s)\| \geq M, \tag{2.24}$$

where M is a big positive constant. If M is chosen big enough, then the stopping criterion is only triggered if the LP problem is infeasible. We refer to the paper of Todd and Ye [133] for a proof of the statement that if an LP problem is nearly infeasible, then it has a very large solution. In practice, this stopping criterion is undesirable, because of difficulties in choosing M. Moreover, numerical problems usually arise when the primal-dual algorithm is applied to an infeasible problem, see [96].

How to detect primal and dual infeasibility is discussed in the next section.

Several important issues have been left out of the discussion above. For example how to choose the reduction γ in the barrier parameter and how to solve the linear equations (2.20). This is discussed below.

2.4.5 The homogeneous algorithm

The primal-dual algorithm presented in the previous section cannot detect an infeasible or unbounded status of an LP problem. The reason is that the primal-dual algorithm tries to solve the primal-dual equations (2.17) and these equations do not have a solution if the LP problem is either primal or dual infeasible. Clearly it is unsatisfactory that the algorithm cannot detect an infeasible status.

This problem leads Ye, Todd and Mizuno [156] to propose a homogeneous and self-dual LP model. The homogeneous model always has a solution and the solution either proves that the LP problem is infeasible or can easily be modified into the optimal solution.

Here we present a simplified version of Ye et al. [156] ideas presented by Xu, Hung and Ye [148]. Their model is

$$\begin{array}{rcl} Ax - b\tau & = & 0, \quad x,\ \tau > 0, \\ A^T y + s - c\tau & = & 0, \quad s > 0, \\ b^T y - c^T x - \kappa & = & 0, \quad \kappa > 0. \end{array} \tag{2.25}$$

The system (2.25) is a homogenization of KKT conditions to (LPP) and the system always has zero as a solution.

Ye et al. [156] proves the following two theorems.

Theorem 2.4.2 *A strictly complementarity solution $(x^*, \tau^*, y^*, s^*, \kappa^*)$ to (2.25) always exists that is*

$$x_j^* + s_j^* > 0 \textit{ and } \tau^* + \kappa^* > 0.$$

Theorem 2.4.3 *Let $(x^*, \tau^*, y^*, s^*, \kappa^*)$ be any strictly complementarity solution to (2.25), then*

1. *$\tau^* > 0$ if and only if (LPP) is feasible and bounded. In this case $(x^*/\tau^*, y^*/\tau^*, s^*/\tau^*)$ is a primal and dual optimal solution to (LPP).*

2. *$\kappa^* > 0$ if and only if (LPP) is infeasible or unbounded.*

From these two theorems it follows that we should generate an strictly complementarity solution to the model (2.25). This solution provides all the information needed.

Xu et al. [148] propose to solve the homogeneous model (2.25) using the primal-dual algorithm outlined in the previous section. Hence, they suggest the following algorithm.

Define the primal and dual residuals

$$r_P^0 = b\tau^0 - Ax^0 \text{ and } r_D^0 = c\tau^0 - A^T y^0 - s^0$$

respectively. Also define

$$r_G^0 = c^T x^0 - b^T y^0 + \kappa^0.$$

Now the search direction is given by

$$\begin{array}{rcl} A^T \delta y + \delta x - c\delta\tau &=& \eta r_D^0, \\ A\delta x - b\delta\tau &=& \eta r_P^0, \\ b^T \delta y - c^T \delta x - \delta\kappa &=& \eta r_G^0 \end{array} \tag{2.26}$$

and

$$\begin{array}{rcl} S^0 \delta x + X^0 \delta s &=& -X^0 s^0 + \gamma\mu^0, \\ \kappa^0 \delta\tau + \tau^0 \delta\kappa &=& -\tau^0 \kappa^0 + \gamma\mu^0, \end{array} \tag{2.27}$$

where $\eta, \gamma \in [0,1]$ are two parameters to be chosen. Moreover,

$$\mu^0 = \frac{(x^0)^T s^0 + \tau^0 \kappa^0}{n+1}.$$

Let

$$(x^1, \tau^1, y^1, s^1, \kappa^1) = (x^0, \tau^0, y^0, s^0, \kappa^0) + \alpha(\delta x, \delta\tau, \delta y, \delta s, \delta\kappa) \tag{2.28}$$

for a suitable chosen stepsize $\alpha \in [0,1]$ such that $(x^1, \tau^1, s^1, \kappa^1) > 0$. If we choose the stepsize α such that

$$\min(X^1 s^1, \tau^1 \kappa^1) \geq \beta\mu^1 \tag{2.29}$$

we are guaranteed that the optimal solution is a strictly complementarity solution, see [70]. β is a positive fixed constant and we use the definition

$$\min(v) = \min_j(v_j),$$

where v is a vector. Convergence of the algorithm follows from Lemma 2.4.2.

Lemma 2.4.2 *Let $(x^1, \tau^1, y^1, s^1, \kappa^1)$ be defined by (2.28), then*

1. $r_P^1 = (1 - \alpha\eta) r_P^0$.

2. $r_D^1 = (1 - \alpha\eta) r_D^0$.

3. $r_G^1 = (1 - \alpha\eta) r_G^0$.

4. $(x^1)^T s^1 + \tau^1 \kappa^1 = ((1 - \alpha(1-\gamma)) + \alpha^2 \eta(1 - \eta - \gamma))((x^0)^T s^0 + \tau^0 \kappa^0)$

Proof: See Xu et al. [148].

□

If we choose η and γ such that

$$\gamma = 1 - \eta,$$

then both the residuals and the complementarity gap decrease linearly in the stepsize α. Furthermore, they decrease at the same rate. Therefore, feasibility and optimality are reached at same time. Hence, the homogeneous algorithm behaves like the feasible primal-dual algorithm. Using arguments similar to those in Kojima et al. [84], it is easy to show that the homogeneous algorithm is globally convergent.

In summary, the main disadvantage of the homogeneous algorithm is that a slightly larger linear system must be solved in each iteration than in the primal-dual algorithm. On the other hand, we gain the advantage that a primal or dual infeasibility can be detected.

One more advantage of the homogeneous model being a whole ray is a solution to the model. This has the consequence that if the initial point is far from the optimal point in size, then the algorithm can easily correct this problem by adjusting τ. Hence, it can be expected that the homogeneous algorithm is less sensitive to the choice of starting point. Indeed, the following simple starting point

$$(x^0, \tau^0, y^0, s^0, \kappa^0) = (e, 1, 0, e, 1)$$

works well in practice.

Xu et al. [148] document that the homogeneous algorithm works well in practice. Moreover, they show the algorithm detects an infeasible status correctly.

An open research question is whether the solution produced by homogeneous algorithm to an primal or dual infeasible LP is useful. In other words, can the solution help pinpoint the faulty part of the problem? Some of the ideas presented by Greenberg [68, pp. 183-184] might help answering this question successfully.

2.5 Complexity

One major feature of the barrier algorithms presented above is that under suitable assumptions they have polynomial complexity. We will not present a detailed complexity proof in this section, but only discuss its main ingredients.

Our discussion is based on the proof techniques presented in the papers by Monteiro and Adler [113], Tseng [138] and Roos and Vial [125].

Assume an initial point is known such that

$$(x^0, y^0, s^0) \in \mathcal{N}_2(\mu^0, \beta) \tag{2.30}$$

for some $\beta \in (0, 1]$ and $\mu^0 \geq 0$. This implies

$$(x^0, y^0, s^0) \in \dot{\mathcal{F}} \tag{2.31}$$

and

$$\left\| X^0 s^0 - \mu^0 e \right\|_2 \leq \beta \mu^0. \tag{2.32}$$

Lemma 2.5.1 *Given (x^0, y^0, s^0) satisfies (2.31) and (2.32). The duality gap is equal to*

$$c^T x^0 - b^T y^0 = (x^0)^T (s^0) \leq (1 + \beta) n \mu^0.$$

It can be proved that if we pick

$$\gamma = 1 - \frac{1}{O(1)\sqrt{n}} \tag{2.33}$$

and compute $(\delta x, \delta y, \delta s)$ from (2.20) then

$$(x^1, y^1, s^1) = (x^0, y^0, s^0) + (\delta x, \delta y, \delta s) \in \mathcal{N}_2(\gamma \mu^0, \beta).$$

The exact choice of β and γ can be seen in Monteiro and Adler [113].

Repeating this procedure and using Lemma 2.5.1 we obtain the duality gap converges linearly to zero at the rate (2.33). Monteiro and Adler [113] show that assuming integer data and using a big-M construction it is possible to create an LP problem which has a known solution that satisfies (2.30) for $\mu^0 \leq 2^{O(L)}$. L is the bit size of the original problem. Furthermore, they show that if the solution (x, y, s) satisfies (2.30) for $\mu \leq 2^{-O(L)}$, then, using a rounding procedure, an exact optimal solution to (LPP) can be obtained. Hence, it can be verified that the algorithm requires $O(\sqrt{n}L)$ iterations to produce an exact optimal solution. This is currently the best iteration complexity and was first obtained by Renegar [122] for an interior-point algorithm.

Similar complexity results can be obtained for the primal and dual barrier algorithm. The computational most expensive operation in each iteration of the interior-point algorithms is ' the solution of the Newton equations. They can be solved accurately in $O(n^3)$ operations. It is possible to modify the interior-point algorithms such that only an approximate solution is needed using an idea presented by Karmarkar [82]. This approximate solution can be obtained in $O(n^{2.5})$ operations. Therefore, the theoretical best interior-point algorithms has the complexity $O(n^3 L)$.

The polynomial barrier algorithm has global linearly convergence with the rate (2.33). For large n this is a very slow convergence. Indeed, the polynomial algorithm presented above is too slow for all practical purposes.

In the complexity proof, the barrier parameter is only reduced slightly, because it must be guaranteed that one iteration of Newton's method can bring the current point back close to the central path. Now the complexity proof is a worst case analysis, and the central path might be very flat. Therefore, even if the barrier parameter is reduced a lot, one iteration of Newton's method might be able to bring the current point back close to the central path.

Hence, an obvious improvement of the complexity of the algorithms is to try to reduce the barrier parameter more in at least some iterations. The Mizuno, Todd and Ye [110] predictor-corrector algorithm exploits this idea. The predictor-corrector algorithm first takes a affine scaling step followed by a pure centering step to move the point back to the central path. In the affine scaling step, a large reduction in complementarity gap might be achieved.

Several researchers have tried to improve the complexity of the interior-point algorithms. A negative result has been obtained by Todd and Ye [132]. They prove that the primal-dual algorithm may take at least $n^{1/3}$ iterations to reduce the duality gap by a constant factor. Powell [121] also presents an example where the number of iterations for Karmarkar's original algorithm grows linearly in the number of variables.

Another stream of research is the so-called condition based complexity. This research is based on the observation that even though two LP problems are of the same size they are not necessarily equally difficult to solve. Therefore, several researchers have proposed different condition numbers for an LP problem and developed algorithms which complexity is dependent on the condition number. The best results in this direction have been obtained by Vavasis and Ye [144].

A surprising observation about interior-point methods is that they usually converge in a very low number of the iterations. The primal-dual algorithm solves most if not all LP problems in less than 100 iterations. Therefore, for practical problems the growth in number iterations is much slower than what the worst case analysis suggests.

2.6 Finite termination and cross-over

Contrary to the simplex algorithm, the interior-point methods do not generate an exact optimal solution. Instead they generate an infinite sequence converging towards the optimal solution.

To overcome this problem, Ye [151] has proposed a finite termination scheme, see also Mehrotra and Ye [108]. In this approach, information from the interior-point algorithm is used to guess the correct optimal partition. When the optimal partition is known, an exact optimal solution can be generated and the interior-point method is terminated.

Another related problem is the construction of an optimal basic solution from the interior-point solution. In general, an interior-point method does not produce an optimal solution that is also a basic solution. We will not discuss this issue in detail here, but refer the reader to Chapter 5 of this dissertation.

2.7 Asymptotic convergence

Without finite termination, the interior-point algorithms generate an infinite sequence converging towards the optimal solution. Traditionally, such algorithms have been evaluated on their rate of convergence and not on their computational complexity. For example, it is well-known that Newton's method is quadratically convergent which is considered very good. In general, any convergence, at least as fast as super-linear, is called fast convergence, whereas linear convergence usually is too slow.

In an interior-point method, rate of convergence usually means the convergence rate of the duality gap $(x^k)^T s^k$. However, the convergence rate of the solution sequence, is of course, also important. Note quadratic convergence of the duality gap does not mean the solution sequence is quadratically convergent.

Interior-point methods consist of solving a set of nonlinear equations using Newton's method and, therefore, super-linear or quadratic convergence should be expected. However, the methods do not solve **one** set of nonlinear equations, because the equations are parameterized by the barrier parameter. In the affine scaling version of the barrier algorithms, the barrier parameter is equal to zero in all iterations. Hence, the affine scaling algorithms should have quadratic convergence. This is confirmed for the primal-dual affine scaling algorithm by Ye et al. [155] and Mehrotra [107]. The caveat is that the primal-dual affine scaling algorithm is not polynomial unless extremely short steps are taken. It seems that polynomial complexity and fast convergence are contradictory. This problem is solved by the predictor-corrector algorithm [110] which, if possible, reduces the complementarity gap a lot while staying close to the central path. This algorithm has both polynomial complexity and quadratic convergence. A similar result has been obtained for another variant of the interior-point methods, see McShane [101]. Potra [119] has also obtained quadratic convergence for the infeasible predictor-corrector method.

2.8 Implementation of the primal-dual algorithm

In this section, we will discuss some issues related to implementing the primal-dual algorithm and its variants.

A lot of work has been done on implementing the primal-dual algorithm. A partial list of references is [102, 92, 93, 95, 96, 106, 148]. More references can be found in the survey papers [98, 97, 65, 9].

The first practical improvement of the primal-dual algorithm is to use different stepsizes in the primal and dual space. It means (2.21) is replaced by

$$x^1 = x^0 + \alpha_P \delta x \text{ and } (y^1, s^1) = (y^0, s^0) + \alpha_D(\delta y, \delta s), \tag{2.34}$$

where $\alpha_P, \alpha_D \in [0, 1]$. The advantage of using different stepsizes is that the infeasibility is likely to be reduced faster. This typically saves around 10% of the iterations, see [65].

The idea of the primal-dual algorithm is to follow the central path more or less closely using a first order approximation (=Newton direction) to the central path. It is therefore natural to try to use a high-order approximation to improve the efficiency of the algorithm. The most efficient high-order variant, in practice, is due to Mehrotra [106]. It can be described as follows. If we plug in $(x^0 + \delta x, y^0 + \delta y, s^0 + \delta s)$ for (x, y, s) in (2.17), then we obtain

$$\begin{array}{rcl} A(x^0 + \delta x) - b & = & 0, \\ A^T(y^0 + \delta y) + (s^0 + \delta s) - c & = & 0, \\ (X^0 + \Delta X)(s^0 + \delta s) - \gamma \mu^0 e & = & 0, \end{array} \tag{2.35}$$

where $\Delta X = diag(\delta x)$. Now this system is equal to

$$\begin{array}{rcl} A\delta x & = & r_P^0, \\ A^T \delta y + \delta s & = & r_D^0, \\ S^0 \delta x + X^0 \delta s & = & -X^0 s^0 + \gamma \mu^0 - \Delta X \delta s, \end{array} \tag{2.36}$$

where we use the definitions of Section 2.4.4. Due to the nonlinear term $\Delta X \delta s$ (2.36) cannot be solved easily. By comparing (2.20) and (2.36) we see that the plain Newton direction is obtained by setting $\Delta X \delta s = 0$ in (2.36). Mehrotra's main contribution is to propose another and better guess for the term

$\Delta X\delta s$. Mehrotra first solves (2.36) for $\gamma = 0$ and $\Delta X\delta s = 0$ and consequently computes the primal-dual affine scaling direction $(\delta x^a, \delta y^a, \delta s^a)$. Next Mehrotra computes

$$\alpha_P^{\max} = \max_{0 \le \alpha \le 1} (x^0 + \alpha \delta x^a \ge 0) \text{ and } \alpha_D^{\max} = \max_{0 \le \alpha \le 1} (s^0 + \alpha \delta s^a \ge 0)$$

and sets

$$\gamma = \left(\frac{(x^0 + \alpha_P^{\max} \delta x^a)^T (s^0 + \alpha_D^{\max} \delta s^a))}{(x^0)^T s^0} \right)^2. \tag{2.37}$$

The final direction is obtained by solving (2.36) again, where the γ is estimated by (2.37) and $\Delta X^a \delta s^a$ is used as an estimate for $\Delta X \delta s$. The estimation of γ in (2.37) is an adaptive heuristic which tries to reduce the barrier parameter a lot when the central path is flat.

Note the computational cost of Mehrotra's predictor-corrector algorithm is limited to one solve, because the linear system (2.36) is solved twice for different righthand sides instead of once as in the plain primal-dual algorithm.

In practice, Mehrotra's predictor-corrector method reduces the number of iterations by 20% to 50% at a very moderate cost, see [106, 95].

Lustig et al. [96] show how to combine Mehrotra's predictor-corrector ideas with global convergence ideas of Kojima et al. [84] such that a globally convergent algorithm is obtained. An analysis of the complexity of a variant of Mehrotra's predictor-corrector algorithm is presented in Zhang and Zhang [157].

Mehrotra's predictor-corrector algorithm uses a second order approximation to the central path. It is, of course, possible to use a high-order approximation. This possibility is studied in Mehrotra [105] and in Carpenter et al. [30]. However, they report that high-order methods beyond two are not worthwhile. Recently, Gondzio [63] proposed an alternative high-order method that shows some promise.

Mehrotra's predictor-corrector algorithm can also be used in combination with the homogeneous primal-dual algorithm, see Xu et al. [148]. Based on the homogeneous algorithm, Hung and Ye [76] presents a very nice complexity result for a high-order algorithm.

2.8.1 Solution of the Newton equations

In each iteration of the primal-dual algorithm, the following linear system

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S^0 & 0 & X^0 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \\ \delta s \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \end{bmatrix} = r \tag{2.38}$$

must be solved for different righthand-sides r. This is the single most expensive operation. Therefore, in this section we will discuss how this can be done efficiently.

Using the identity

$$\delta s = (X^0)^{-1}(r_3 - S^0 \delta x) \tag{2.39}$$

we can reduce the system (2.38) to

$$\begin{bmatrix} -(X^0)^{-1} S^0 & A^T \\ A & \end{bmatrix} \begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} r_2 - (X^0)^{-1} r_3 \\ r_1 \end{bmatrix}. \tag{2.40}$$

A further reduction is obtained using

$$\delta x = -(S^0)^{-1} X^0 (r_2 - (X^0)^{-1} r_3 - A^T \delta y). \tag{2.41}$$

Finally, we obtain the linear system

$$(A X^0 (S^0)^{-1} A^T) \delta y = r_1 + A X^0 (S^0)^{-1} (r_2 - (X^0)^{-1} r_3). \tag{2.42}$$

A solution approach based on the system (2.40) is called the augmented system approach. Whereas an approach based on the system (2.42) is called the normal equations approach.

We first discuss the normal equations approach. The matrix

$$M = (A X^0 (S^0)^{-1} A^T) \tag{2.43}$$

is a positive definite matrix given the assumption that A is of full row rank. Two important observations about M are: First, if A is a sparse matrix, then M also tends to be a sparse matrix. Second, the sparsity pattern in M is independent of the matrices X^0 and S^0. From the positive definiteness and these two observations it follows that a sparsity preserving pivot order can be chosen once and for all, and this ordering can be kept fixed in all interior-point iterations. Therefore, an approach based on solving (2.42) has been found to be highly efficient in practice for most LP problems. For details on the implementation of this approach we refer the reader to [94, 81, 97, 9]. Note that the system (2.42) can be solved by an iterative method e.g. conjugate gradients. However, nobody has reported good results for general LP problems using this option.

Two concerns have been raised about the normal equations approach. First, even though A is sparse, then M might be completely dense. This happens for example if A contains one or more dense columns. However, an efficient method for handling dense columns has been developed recently, see Andersen [15]. Unfortunately, other problems exists where M is highly sparse, but nevertheless the Cholesky decomposition becomes very dense. Such examples are reported by Resende, Ramakrishnan and Drezner [123] and Lustig et al. [97]. Second, the normal equations approach might be numerically unstable, see Saunders [126].

Both of these issues can be addressed if (2.40) is solved directly. The linear system in (2.40) is symmetric and indefinite, and therefore the Cholesky decomposition cannot be used. Based on ideas presented in Duff et al. [45], Turner [139] and Fourer and Mehrotra [50, 51] have pioneered this augmented system approach. However, except for a few special LP problems, their approach is not competitive with the normal equations approach. The main reason is that they must allow for general numerical pivoting which slows things down.

An intermediate approach between the normal equations and the augmented system approach has been proposed by Vanderbei [142, 141]. This approach is not numerically stable as the augmented system approach, but on the other hand, it might be more efficient in practice.

Finally, iterative methods for direct solution of (2.38) have been proposed by Gill et al. [60] and Freund and Jarre [52]. Since they present few numerical results, it is difficult to evaluate the value of their ideas. However, especially the QMR method studied by Freund and Jarre shows some promise.

In summary, the normal equations approach for solving the system (2.38) seems to be the best approach for most LP problems. However, it is still an open question how to solve (2.38) for some classes of LP problems.

2.8.2 Presolving

In practice, very large problems containing thousands of constraints and variables are solved. The mere size of the problems makes it difficult for the model formulator to choose an optimal formulation that minimizes redundancy in the model. It is therefore advantageous to remove redundancy from the LP problem before submitting it to optimization. The reduction phase is called a presolve and is discussed in Chapter 3.

2.9 Extension to convex problems

The interior-point methods presented in the previous sections can be applied to general convex problems such as

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \ldots, m, \end{array} \tag{2.44}$$

where f and g_i are convex functions.

Several researchers have studied solution of convex problems. Some references are [114, 38, 117, 80]. In particular, the book by Nesterov and Nemirovskii [117] presents the development of interior-point methods in a general setting.

A common feature of the proposed algorithms is that, if the functions satisfy certain smoothness conditions, then a polynomial algorithm can be developed for the problem. We refer the reader to the paper of den Hertog et al. [39] for a discussion of the different smoothness conditions.

Another active area of research is the development of interior-point algorithms for special classes of convex problems. Recently, positive semi-definite programming has received a lot of attention, see Alizadeh [4] and Vandenberghe and Boyd [140]. Positive semi-definite programming has numerous applications in control theory and in combinatorial optimization.

Another example is the problem of minimizing a sum of norms.This problem has been studied by Andersen [13, 17, 16] and Xue and Ye [149].

The extension of the primal-dual algorithm to convex problems has been studied by Vial [145, 18]. Kortanek, Xu and Ye [89] develop a primal-dual algorithm for the class of geometric problems. For this class of problems, their algorithm works well.

Finally, Andersen and Ye [10] study the extension of the homogeneous algorithm to the monotone complementarity problem which also covers the class of convex problems. For details we refer the reader to a Chapter 6 of this dissertation.

In summary, several (many) interior-point methods have been developed for convex programs. In theory, they solve a convex optimization problem in the same number of iterations as an LP problem of the same size. Hence, it can be expected that the methods are efficient. However, the development of general purpose software has lacked behind the algorithmic development. The main reasons for this are lack of good test examples, no standard input format, the difficult generation of derivatives (=gradient and Hessian) and the fact that the Newton equation system is more difficult to solve than in LP.

2.10 Extension to non-convex problems

Interior-point methods can also be applied to non-convex problems, but it is in general not possible to prove any complexity results for this class of problems.

The classic book by Fiacco and McCormick [48] presents different interior-point methods for non-convex problem.

Newer material includes the paper of El-Bakry et al. [46]. They study the extension of the primal-dual algorithm to non-convex problems and show that the algorithm with some safeguards is locally convergent and achieves quadratic convergence. They also present good numerical results for several small problems. Another primal-dual algorithm is studied by McCormick [100].

Breitfeld and Shanno [28, 27] also present a primal interior-point algorithm.

It is possible to develop polynomial algorithms for certain non-convex problems. An example is the von Neumann economic growth model

$$\begin{array}{rl} \text{maximize} & \gamma \\ \text{subject to} & (B-\gamma A)y \geq 0, \\ y \geq 0, & \end{array} \tag{2.45}$$

where A, B are two nonnegative matrices in $R^{m\times n}$. This problem has been studied by Ye [152]. Related problems have been studied by Freund and Jarre [53] and Nesterov and Nemirovskii [118].

2.11 Interior-point methods from a practical point of view

In the previous sections we have mainly presented interior-point methods from a theoretical point of view that is we have discussed derivation of the methods and their convergence properties. One might question practical applications of the methods.

The recent papers of Lustig et al. [97] and Bixby [21] document that interior-point methods are a significant contribution to the solution of practical large-scale LP problems. Indeed, for some problems, interior-point methods are, in orders of magnitude, faster than the best implementations of the simplex algorithm. The reverse might also be the case.

The interior-point methods are not only fast but they have other nice properties. First, they use a predictable number of iterations, because an interior-point algorithm usually solves an LP problem in less than 50 iterations and more than 100 iterations is never required. Hence, after the first interior-point iteration is performed it is easy to estimate the solution time and the algorithm can be terminated if the estimated solution time is considered excessive. Second, the interior-point methods rarely need any finetuning of the parameters of the algorithm. This is contrary to the simplex algorithm. All commercial optimization package based on the simplex algorithm come with several scaling options, several pricing options, and several crash options and the solution time might vary considerably depending on what option is chosen. Therefore, the interior-point methods are easier to use than the simplex algorithm.

One disadvantage of the interior-point methods is that they do not generate an optimal solution which is also a basic solution. A basic solution is needed to perform the traditional (perhaps misleading) sensitivity analysis and to do efficient warm-starts. This is not a limitation any more, because, as documented in Bixby [21] and in Chapter 5, an optimal basic solution can be cheaply recovered from the interior-point solution.

Nevertheless, there are still some cases where a simplex algorithm performs much better than the interior-point methods. Indeed, when a sequence of closely related problems is solved, the simplex algorithm is much better at exploiting information from the previous solved problem. It is still an open question how to perform warm-starts within the framework of an interior-point method. Anyway, due to the computationally expensive interior-point iteration, compared to a simplex iteration, it is unlikely that the interior-point algorithms are going to the replace the simplex algorithm when warm-starts are performed.

Even though the interior-point algorithms usually perform well LP problems exist where the cost of each interior-point iteration is prohibitively large. This happens when there are a lot of fill-in in the Cholesky factors. As discussed in Section 2.8.1, alternative solution methods for the Newton equations (2.38) might solve this problem.

2.12 Conclusion

In this paper, we have surveyed the last 10 years of development of interior-point methods. We have focused on general methods for linear programming. Our conclusion about these methods is that the current theoretical and practical knowledge about the methods are well-developed. Although improvements are always possible. We believe the main area for new research is the extension of the methods to nonlinear problems. Especially, the extension to the class of convex problems is likely to be successful.

Kapitel 3

Presolving in Linear Programming

Most modern linear programming solvers analyze the LP problem before submitting it to optimization. Some examples are the solvers WHIZARD [136], OB1 [97], OSL [49], Sciconic [91] and CPLEX [21]. The purpose of the presolve phase is to reduce the problem size and to discover whether the problem is unbounded or infeasible.

In this paper we present a comprehensive survey of presolve methods. Moreover, we discuss the restoration procedure in detail, i.e., the procedure that undoes the presolve.

Computational results on the NETLIB (see Gay [56]) problems are reported to illustrate the efficiency of the presolve methods.

This paper is co-authored with Knud D. Andersen [1].

3.1 Introduction

Even though computers and LP algorithms have been improved a lot during the last 10 years, it is still important to present an LP problem to a solver in a properly formulated form. There are some rules a properly formulated LP problem must satisfy. It should contain as few variables, constraints, and non-zeros as possible, it must be well-scaled and the constraints must be linearly independent.

These simple rules may not be satisfied in practice. Therefore, it is advantageous to implement a presolve procedure in an LP solver to detect redundancy and to remove it.

Presolving is an old idea and has been treated by Brearley, Mitra and Williams [26], Williams [146], Bradley, Brown and Graves [25] and Tomlin and Welch [136, 137]. All these authors propose the same basic approach to presolving, namely to use an arsenal of simple and fast inspection type procedures to detect various forms of redundancy and then repeat these procedures until the problem cannot be reduced further. We use the same strategy, but propose a more exhaustive presolve procedure.

Clearly, there is a trade-off between how much redundancy a presolve procedure detects and the time spent in the presolve procedure. The optimal strategy is to detect and remove the types of redundancy which lead to a reduction in the total solution time. However, this is not possible, and therefore the conservative strategy of detecting simple forms of redundancy, but doing it fast, seems to be the best strategy.

Another type of presolve procedure has been proposed by Adler et al. [1] and Chang and McCormick [32]. They propose using general linear transformations on the constraints to reduce the number of non-zeros in the problem. This option is not discussed here.

Papers [26, 146, 25, 136, 137] discuss a presolve procedure in combination with the simplex algorithm. Here we use the presolve procedure in combination with an interior-point algorithm. Papers [2, 93] already document that a presolve procedure in combination with an interior-point algorithm is beneficial.

The paper is organized as follows. In Section 3.2 we present our notation. In Section 3.3 the presolve procedure is described. In Section 3.4 we discuss how to recover an optimal primal and dual feasible solution to the original problem. In Section 3.5 we discuss our implementation of the presolve procedure. In Section 3.6 computational results are presented and finally in Section 3.7 our conclusions are presented.

[1] Odense University, Department of Mathematics and Computer Sciences, Campusvej 55, DK-5230 Odense M. The author was supported by an Danish SNF Research studentship.

l_j	u_j	z_j^*	x_j^*	Problem status
$> -\infty$	$+\infty$	> 0	l_j	-
$-\infty$	$< +\infty$	< 0	u_j	-
$-\infty$	?	> 0	-	Unbounded
?	$+\infty$	< 0	-	Unbounded
?	$< l_j$	?	-	Infeasible

Tabel 3.1: Conclusion that can be reached from the optimality conditions. (? means any value. - means unknown.)

3.2 Notation

Any LP problem can be formulated in the following form

$$\begin{array}{ll} \min & c^T x, \\ \text{s.t.} & Ax = b, \\ & l \leq x \leq u \end{array} \tag{3.1}$$

where $x, c, l, u \in R^n$, $b \in R^m$, and $A \in R^{m \times n}$. Some of the simple bounds l_j and u_j may be $-\infty$ and $+\infty$ respectively. Define $L = \{j : l_j > -\infty\}$ and $U = \{j : u_j < +\infty\}$ then the optimality conditions to (3.1) can be stated as follows

$$\left\{ \begin{array}{ll} Ax^* = b & \\ A^T y^* + z^* = c & \\ (x_j^* - l_j) z_j^* = 0, & \forall j \in L \\ (u_j - x_j^*) z_j^* = 0, & \forall j \in U \\ z_j^* \geq 0, & \forall j \in \{j \in L : x_j^* = l_j \wedge l_j < u_j\} \\ z_j^* \leq 0, & \forall j \in \{j \in U : x_j^* = u_j \wedge l_j < u_j\} \\ z_j^* = 0, & \forall j \notin L \cup U \\ l \leq x^* \leq u & \end{array} \right. \tag{3.2}$$

where $y \in R^m$ and $z \in R^n$. y and z are the Lagrange multipliers corresponding to the linear constraints and the linear inequalities in (3.1), respectively. Note that the superscript * means optimal values. In Table 3.1 we have listed the conclusions that can be deduced from the optimality conditions (3.2). For example, if it can be established that $z_j^* > 0$ then x_j is fixed at its lower bound if it is finite, otherwise the problem is unbounded.

3.3 A presolve procedure

The computational complexity of simplex and interior-point type LP algorithms is dependent on the number of constraints and variables in (3.1). Even more important in practice is the sparsity of A, because only non-zeros are stored and the work is dependent on the number of non-zeros. Therefore, a presolve procedure should reduce the size of A without creating any new non-zeros in A. This has the advantage that the storage cost cannot be increased by the presolve procedure. The disadvantage of not allowing the creation of new fill-ins is that general linear transformations on A are not allowed even though it might be beneficial in some circumstances. However, we allow the presolve procedure to modify the objective function c, the right hand side b and the simple bounds l and u.

In the following, we assume that A is sparse i.e. contains less than, say, 30% non-zeros. The presolve procedure might not be advantageous if this assumption is not fulfilled. Furthermore, we assume that all reductions on (3.1) are applied immediately and therefore we always work on the reduced problem. Whenever we write $\forall i$ and $\forall j$ it means for all rows and for all columns in the current reduced problem.

The idea of the presolve procedure is to make several passes through A and in each pass various forms of redundancies are detected and removed. In the following sections we discuss the procedure in detail.

l_j	u_j	a_{kj}	z_j^*	y_k^*
$-\infty$	$+\infty$	$\neq 0$	$=0$	$= c_j/a_{kj}$
$> -\infty$	$+\infty$	> 0	≥ 0	$\leq c_j/a_{kj}$
$> -\infty$	$+\infty$	< 0	≥ 0	$\geq c_j/a_{kj}$
$-\infty$	$< +\infty$	> 0	≤ 0	$\geq c_j/a_{kj}$
$-\infty$	$< +\infty$	< 0	≤ 0	$\leq c_j/a_{kj}$

Tabel 3.2: Bounds on the optimal Lagrange multipliers y^* implied by the column singleton a_{kj}.

3.3.1 Simple presolving methods

Here we present some standard (simple) presolve methods. All of these have been treated before, see for example [25], but are included here for the sake of completeness.

(i) *An empty row:*

$$\exists i : a_{ij} = 0, \forall j. \tag{3.3}$$

Either the ith constraint is redundant or infeasible.

(ii) *An empty column:*

$$\exists j : a_{ij} = 0, \forall i. \tag{3.4}$$

Dependent on the bounds on variable x_j and its objective coefficient c_j, variable x_j is fixed at one of its bounds or the problem is unbounded.

(iii) *An infeasible variable:*

$$\exists j : l_j > u_j. \tag{3.5}$$

The problem is trivially infeasible.

(iv) *A fixed variable:*

$$\exists j : l_j = u_j. \tag{3.6}$$

Variable x_j can be substituted out of the problem.

(v) *A singleton row:*

$$\exists (i,k) : a_{ij} = 0, \forall j \neq k, a_{ik} \neq 0. \tag{3.7}$$

The ith constraint fixes variable x_j at level b_i/a_{ik}.

It should be noted that one reduction may lead to further reductions. For example, if A is a permuted lower triangular matrix, repeated use of (3.7) solves the problem. To exploit this possibility, our presolve procedure first counts the number of non-zeros in each constraint. Then, a list containing all singleton constraints is created. As the constraints in this list are used to fix variables, the number of non-zeros for each constraint is updated and new singleton constraints are appended to the list.

3.3.2 Column singletons

Now we study how to exploit column singletons where the definition of a column singleton is

$$\exists (j,k) : a_{ij} = 0, \forall i \neq k, a_{kj} \neq 0. \tag{3.8}$$

If one of the bounds on the column singleton is infinite then the column singleton imposes a bound on one of the optimal Lagrange multipliers. All such bounds are listed in Table 3.2. Such bounds will be useful later.

(vi) *A free column singleton:*

$$\exists (j,k) : (a_{ij} = 0, \forall i \neq k, a_{jk} \neq 0) \wedge l_j = -\infty \wedge u_j = +\infty. \tag{3.9}$$

The substitution

$$x_j = (b_i - \sum_{p \neq j} a_{ip} x_p)/a_{ik} \tag{3.10}$$

can be done.

The free column singleton reduction is very advantageous, because one constraint and one variable is removed from the problem without generating any fill-ins in A, although the objective function is modified. Next we will discuss two methods for generating more free column singletons.

(vii) *A doubleton equation combined with a column singleton:*

$$\exists i, j, k : a_{ij}x_j + a_{ik}x_k = b_i, j \neq k, a_{ij} \neq 0, a_{ik} \neq 0. \tag{3.11}$$

If variable x_k is a column singleton then the bounds on the variable x_j are modified so that the feasible region is unchanged even if the bounds on x_k are removed. Afterwards variable x_k can be substituted out of the problem.

It is also possible in some cases to establish that bounds on a singleton are redundant. An example is

$$x_j - \sum_{k \neq j} x_k = 0, \quad x_j, x_k \geq 0, \tag{3.12}$$

where xj is assumed to be column singleton. Obviously, the simple bounds on variable x_j can be dropped without changing the feasible region, thereby creating a free column singleton. This example can be generalized as follows.

Whenever our presolve procedure detects a column singleton x_j, it tries to establish that it is implied free using the following technique. For each $a_{ij} \neq 0$ the following bounds on variable x_j can be computed

$$u'_{ij} = \begin{cases} (b_i - \sum_{k \in P_{ij}} a_{ik}l_k - \sum_{k \in M_{ij}} a_{ik}u_k)/a_{ij}, & a_{ij} > 0 \\ (b_i - \sum_{k \in P_{ij}} a_{ik}u_k - \sum_{k \in M_{ij}} a_{ik}l_k)/a_{ij}, & a_{ij} < 0 \end{cases} \tag{3.13}$$

and

$$l'_{ij} = \begin{cases} (b_i - \sum_{k \in P_{ij}} a_{ik}l_k - \sum_{k \in M_{ij}} a_{ik}u_k)/a_{ij}, & a_{ij} < 0 \\ (b_i - \sum_{k \in P_{ij}} a_{ik}u_k - \sum_{k \in M_{ij}} a_{ik}l_k)/a_{ij}, & a_{ij} > 0 \end{cases} \tag{3.14}$$

where $M_{ij} = \{k : a_{ik} < 0, j \neq k\}$ and $P_{ij} = \{k : \ a_{ik} > 0, j \neq k\}$. It can be verified that $l'_{ij} \leq x_j \leq u'_{ij}$ for any feasible solution x to (3.1). If these new bounds are at least as tight as the original ones the variable is said to be implied free.

(viii) *An implied free column singleton:*

$$\exists j, k : (a_{ij} = 0, \forall i \neq k, a_{kj} \neq 0) \wedge l_j \leq l'_{kj} \leq u'_{kj} \leq u_j \tag{3.15}$$

It is assumed that l'_{kj} and u'_{kj} are computed according to (3.13) and (3.14). The variable x_j can be treated as a free column singleton, (see (3.9)).

Detection of column singletons has been used in other presolve procedures ([25], pp. 149-152), but, at least to our knowledge,a systematic check whether a column singleton is implied free, has not previously been proposed. In the case of a doubleton equation, Bradley et al. [25] propose always modifying the simple bounds on one of the variables such that the other one becomes free. Afterwards, the free variable is substituted out of the problem. We have not implemented this more general procedure, because it might causes fill-ins in A.

3.3.3 Forcing and dominated constraints

In this section, the simple bounds on the primal variables are exploited to detect forcing and dominated constraints.

In order to establish whether a constraint is of the forcing type or the dominated type we compute

$$g_i = \sum_{j \in P_i} a_{ij}l_j + \sum_{j \in M_i} a_{ij}u_j \quad \text{and} \quad h_i = \sum_{j \in M_i} a_{ij}l_j + \sum_{j \in P_i} a_{ij}u_j \tag{3.16}$$

where $P_i = \{j : a_{ij} > 0\}$ and $M_i = \{j : a_{ij} < 0\}$. Clearly we have

$$g_i \leq \sum_j a_{ij} x_j \leq h_i \tag{3.17}$$

for any solution x that satisfies the simple bounds. The following possibilities can occur.

(ix) *An infeasible constraint:*

$$\exists i : h_i < b_i \vee b_i < g_i \tag{3.18}$$

In this case the problem is infeasible.

(x) *A forcing constraint*

$$\exists i : g_i = b_i \vee h_i = b_i \tag{3.19}$$

If $g_i = b_i$ then due to linearity the only feasible value of x_j is l_j (u_j) if $a_{ij} > 0$ ($a_{ij} < 0$). Therefore, we can fix all variables in the ith constraint. A similar statement can be made if $h_i = b_i$.

It is highly advantageous to get rid of forcing constraints, because all variables in them are structurally degenerate.

In the previous section, (3.13) and (3.14) were developed to compute implied bounds on variable x_j. These bounds in combination with the original bounds can be used to detect more free column singletons. A procedure using this method is called a dominated constraint procedure. (The term "dominated" is proposed by Lustig et al. [97].)

In order to limit the computational cost of the dominated constraint procedure we only compute l'_{ij} and u'_{ij} for each $a_{ij} \neq 0$ if either g_i or h_i are finite. Furthermore, g_i and h_i are computed during the forcing constraint check and they can be reused to compute the implied bounds cheaply using the following formulas

$$u'_{ij} = \begin{cases} (b_i - g_i)/a_{ij} + l_j, & a_{ij} > 0 \\ (b_i - h_i)/a_{ij} + l_j, & a_{ij} < 0 \end{cases} \tag{3.20}$$

and

$$l'_{ij} = \begin{cases} (b_i - h_i)/a_{ij} + u_j, & a_{ij} > 0 \\ (b_i - g_i)/a_{ij} + u_j, & a_{ij} < 0 \end{cases}. \tag{3.21}$$

It should be noted that the implied bounds are only used to detect more free column singletons. Dependent on the context in which the dominated constraint procedure is used, the implied bounds may of course be used to tighten or relax the bounds on the primal variables. For example, the bounds should be tightened if the problem is a mixed integer problem. On the other hand, they could also be used to relax the simple bounds. The last possibility should certainly be exploited to create free variables if the reduced problem is solved by a simplex type algorithm, see [136].

Forcing constraints have been implemented in all presolve procedures, at least to our knowledge. The first reference to a dominated constraint procedure is [26, p. 63] (see also [146]) and our procedure is similar to that, although in [26] the procedure is executed more often than we propose. We limit the number of times the procedure is executed, because we have found it to be computationally expensive for some dense problems.

The only other reference to a dominated constraint procedure, we know of, is [97], i.e., it is implemented in the LP solver OB1. However, OB1's dominated constraint procedure seems not to be as general as the procedure proposed in this section.

3.3.4 Forcing and dominated columns

The dominated constraint procedure discussed in the previous section can, of course, be used on the dual problem. We call such a procedure a dominated column procedure and it has been proposed by Williams [146].

The first step of the dominated column procedure is to find all column singletons and use them to compute implied lower and upper bounds on the optimal Lagrange multipliers y^*. How this is done has been treated in Section 3.3.2. Let $\bar{y}_i$ and $\hat{y}_i$ be the maximum and minimum respectively of all such bounds, i.e., we have

$$\bar{y}_i \leq y_i^* \leq \hat{y}_i, \forall i. \tag{3.22}$$

These simple bounds on the Lagrange multipliers can be used to fix some of the primal variables. Define

$$e_j = \sum_{i \in P_j} a_{ij}\bar{y}_i + \sum_{i \in M_j} a_{ij}\hat{y}_i \quad \text{and} \quad d_j = \sum_{i \in P_j} a_{ij}\hat{y}_i + \sum_{i \in M_j} a_{ij}\bar{y}_i \tag{3.23}$$

where $P_j = \{i : a_{ij} > 0\}$ and $M_j = \{i : a_{ij} < 0\}$. Clearly

$$e_j \leq \sum_i a_{ij} y_i^* \leq d_j. \tag{3.24}$$

From the optimality conditions (3.2) and (3.24) it can be deduced that

$$c_j - d_j \leq z_j^* = c_j - \sum_i a_{ij} y_i^* \leq c_j - e_j. \tag{3.25}$$

(xi) *A dominated column:*

$$\exists j : c_j - d_j > 0 \vee c_j - e_j < 0 \tag{3.26}$$

If $c_j - d_j > 0$ then by (3.25) $z_j^* > 0$ and according to Table 3.1 variable x_j can be fixed at its lower bound. However, if $l_j = -\infty$ the problem is unbounded. An analogue conclusion can be reached if $c_j - e_j < 0$.

(xii) *A weakly dominated column:*

$$\exists j \notin S : c_j - d_j = 0 \wedge l_j > -\infty, \tag{3.27}$$

$$\exists j \notin S : c_j - e_j = 0 \wedge u_j < +\infty \tag{3.28}$$

where $S = \{j : a_{ij}$ is a column singleton for some $i\}$. Note that the column singletons are used to generate the bounds d_j and e_j and, therefore, they cannot be dropped with this test. In the following proposition we prove that in the cases (3.27) and (3.28) the variable x_j can be fixed at its lower and upper bound, respectively.

Proposition 3.3.1 *Assume (3.1) has an optimal solution solution. If (3.27) or (3.28) holds, then there exists an optimal solution such that $x_j^* = l_j$ or $x_j^* = u_j$, respectively.*

Proof: Assume (3.27) holds. Furthermore assume without loss of generality that $l = 0$, $u = \infty$ and $j = n$. Then the dual problem of (3.1) is

$$\begin{array}{rl} \max & b^T y, \\ \text{s.t.} & A^T y + s = c, \\ & s \geq 0. \end{array} \tag{3.29}$$

Given the assumptions of the proposition, then (3.29) has an optimal solution. Therefore there exists a dual optimal solution $\bar{x} \in R^{n-1}$ to (3.29) without the last constraint which implies

$$A(\bar{x}, 0) = b \quad \text{and} \quad \bar{x} \geq 0. \tag{3.30}$$

Clearly $x = (\bar{x}, 0)$ is an optimal solution to (3.1) proving there exists an optimal solution such that $x_j^* = 0 = l_j$ is the case. The proof for (3.28) is similar.

□

(xiii) *A forcing column:*

$$\exists j \notin S : (c_j - e_j = 0 \wedge u_j = \infty) \vee (c_j - d_j = 0 \wedge l_j = -\infty). \tag{3.31}$$

Assume that $c_j - e_j = 0$ and $u_j = \infty$ which implies $0 \leq z_j^* \leq c_j - e_j = 0$. By construction we have

$$y_k^* = \left\{ \begin{array}{ll} \bar{y}_k, & a_{kj} > 0 \\ \hat{y}_k, & a_{kj} < 0 \end{array} \right. . \tag{3.32}$$

If $(-y_k^*)$ times the kth constraint is subtracted from the objective function when $a_{kj} \neq 0$, the kth constraint can be removed from the problem. We have not implemented this method because we believe it occurs rarely.

Finally the bounds e_j and d_j are used to compute new bounds on the optimal Lagrange multipliers y^* in the following manner. Assume that $l_j > \infty$ and $u_j = +\infty$. It follows from the optimality conditions (3.2) that

$$0 \leq z_j^* = c_j - \sum_i a_{ij} y_i^*. \tag{3.33}$$

If $a_{kj} \neq 0$ we have

$$y_k^* \leq (c_j - e_j)/a_{kj} + \bar{y}_k, \; a_{kj} > 0 \tag{3.34}$$

or

$$y_k^* \geq (c_j - e_j)/a_{kj} + \hat{y}_k, \; a_{kj} < 0. \tag{3.35}$$

Similarly if $l_j = -\infty$ we obtain

$$y_k^* \geq (c_j - d_j)/a_{kj} + \hat{y}_k, \; a_{kj} > 0 \tag{3.36}$$

and

$$y_k^* \leq (c_j - d_j)/a_{kj} + \bar{y}_k, \; a_{kj} < 0 \tag{3.37}$$

In the case of (3.34) we have $c_j - e_j \geq 0$. Otherwise (3.26) is the case. Observing that $(c_j - e_j)/a_{kj} \geq 0$ it follows

$$\bar{y}_k \leq y_k^* \leq (c_j - e_j)/a_{kj} + \bar{y}_k \tag{3.38}$$

so the new bounds on the Lagrange multiplier cannot be inconsistent. Similarly all the bounds (3.35), (3.37) and (3.36) must be consistent.

The new bounds on the Lagarange multipliers y^* generated by (3.34), (3.35), (3.36) and (3.37) are used to recompute e_j and d_j which in turn are used to detect more dominated columns.

3.3.5 Duplicate rows

Linearly dependent rows in A are undesirable, because in an interior-point algorithm a certain linear system must be solved in each iteration. This linear system has a singular matrix if A contains linearly dependent rows.

A simple form of linear dependent rows is when two rows are identical except for a scalar multiplier. Two such rows are called duplicate by Tomlin and Welch [137]. However, they treat slack variables separately and therefore we use a generalized version of their definition which is

$$\exists i, k : a_{ij} = v a_{kj}, i \neq k, \forall j \notin S \tag{3.39}$$

where $S = \{j : a_{ij}$ is a column singleton for some $i\}$ and v is a scalar multiplier.

Assume the ith and the kth row are duplicate and by definition we have

$$\left\{ \begin{array}{l} \sum\limits_{j \notin S} a_{ij} x_j + \sum\limits_{j \in S} a_{ij} x_j = b_i \\ \sum\limits_{j \notin S} a_{kj} x_j + \sum\limits_{j \in S} a_{kj} x_j = b_k \\ a_{ij} = v a_{kj}, \forall j \notin S \end{array} \right. . \tag{3.40}$$

If the ith row is added $(-v)$ times to the kth we obtain

$$\left\{ \begin{array}{l} \sum\limits_{j \notin S} a_{ij} x_j + \sum\limits_{j \in S} a_{ij} x_j = b_i \\ \sum\limits_{j \in S} a_{kj} x_j - v(\sum\limits_{j \in S} a_{ij} x_j) = b_k - v b_i \end{array} \right. . \tag{3.41}$$

The following situations may occur.

(xiv) *a) The kth row is empty, b) the kth row contains an implied free variable or c) the kth row is a doubleton equation containing a singleton column.*

All these cases have already been treated.

l_k	u_k	z_k^*	v	$c_j - vc_k$	z_j^*
?	$+\infty$	≥ 0	≥ 0	> 0	> 0
?	$+\infty$	≥ 0	≤ 0	< 0	< 0
$-\infty$	?	≤ 0	≥ 0	< 0	< 0
$-\infty$	?	≤ 0	≤ 0	> 0	> 0

Tabel 3.3: Bounds on the Lagrange multipliers when two columns are duplicate. (? means any value.)

(xv) *Constraint k contains only column singletons:*

$$a_{ij} = 0, \forall j \in S. \tag{3.42}$$

In this case we have two independent problems which can be solved separately. We believe this is a rare occurrence and it is not implemented in our presolve procedure.

These checks should be done twice when the ith row is added to the kth and, then vice-versa.

The main objection against finding all duplicate rows is, of course, that it may be computationally too expensive. However, if the problem is sparse we have found that not to be the case. We find all the duplicate columns with a method similar to that proposed by Tomlin and Welch [137, p. 8]. The method consists of two phases. First rows with the same sparsity pattern in non-singletons columns are found, because if two rows are duplicate their sparsity pattern must be identical. It is completed in O(nz(A)) operations and in practice this phase is very fast. In the second phase the duplicate columns are detected in each group of columns with the same sparsity pattern. The second phase is typically more expensive than the first one, because it involves floating point operations. In [137] the two phases are combined but we have separated them because on a sparse problem, typically, very few rows have the same sparsity pattern. Thereby the computational cost in the second phase is minimized.

3.3.6 Duplicate columns

The dual of duplicate rows are duplicate columns, i.e., columns that are identical except for a scalar multiplier. Even though the number of duplicate columns are expected to be modest in number there exists classes of LP problems which contain a large proportion of duplicate columns, see Anbil et al. [5].

We define column j and k to be duplicate if they satisfy the following definition

$$\exists j, k : a_{ij} = v a_{ik}, \forall i, j \neq k \tag{3.43}$$

where v is a scalar multiplier. We have that

$$\begin{aligned} z_j^* &= c_j - \textstyle\sum_i y_i^* a_{ij} \\ &= c_j - v \textstyle\sum_i y_i^* a_{ik} \\ &= c_j + v(z_k^* - c_k) \\ &= c_j - vc_k + vz_k^*. \end{aligned} \tag{3.44}$$

Using (3.44) it is possible to generate bounds on the optimal Lagrange multipliers. All such bounds are listed in Table 3.3.

(xvi) *Fixing a duplicate column:*

$$c_j - vc_k \neq 0. \tag{3.45}$$

Using Table 3.3 it is possible to establish a bound on z_j^*. This bound is used to fix variable x_j or to detect that the problem is dual infeasible.

(xvii) *Replacing two duplicate columns by one:*

$$c_j - vc_k = 0. \tag{3.46}$$

Let us study the part of the problem (3.1) which amounts to the two duplicate columns. We have

$$\begin{aligned} \cdots c_j x_j + c_k x_k \cdots, \\ \cdots a_{\cdot j} x_j + a_{\cdot k} x_k \cdots = b, \\ l_j \leq x_j \leq u_j,\ l_k \leq x_k \leq u_k. \end{aligned} \tag{3.47}$$

v	l'_k	u'_k
< 0	$l_k + vu_j$	$u_k + vl_j$
> 0	$l_k + vl_j$	$u_k + vu_j$

Tabel 3.4: Replacement bounds.

Define

$$x_k = -vx_j + x'_k \tag{3.48}$$

and if x_k in (3.47) is replaced by the right hand side of (3.48) we obtain

$$\begin{array}{l} \cdots c_k x'_k \cdots, \\ \cdots a_{.k} x'_k \cdots = b, \\ l_j \leq x_j \leq u_j, \\ l_k \leq x'_k - vx_j \leq u_k. \end{array} \tag{3.49}$$

Variable x_j and x_k can be replaced by variable x'_k if its bounds are specified as shown in Table 3.4. In summary, whenever two columns are duplicate and (3.46) holds, the duplicate column procedure modifies the bounds on variable x_k according to Table 3.4 and removes variable x_j from the problem.

The duplicate columns are found with the same algorithm as the duplicate rows.

The detection of duplicate columns has been proposed in [137] and has been used in a special case, see [5] and [22], but nowhere are general results of this possibility reported. Note that the duplicate column procedure will detect all split free variables, i.e., free variables modeled by a difference of non-negative variables. Such variables mean serious numerical difficulties in a primal-dual interior-point algorithm. The reason is that the set of primal feasible solutions is unbounded and the interior of the dual feasible region is empty. If the split free variables are combined into free variables the problem can normally be solved.

3.4 Postsolve

After problem (3.1) has been presolved, and if it is not detected infeasible or unbounded the reduced problem must be solved by an LP algorithm. In general, the optimal solution to the reduced problem is not feasible for the original problem. Therefore, a restoration procedure is needed to "undo" the presolving.

There are two kinds of restoration procedures. One of them solves the original problem by the simplex algorithm, using the optimal solution to the reduced problem as a hot-start. The other type of restoration procedure is a reverse presolve procedure. The first method has been used by Williams [146], but can be very inefficient. Observing this, Tomlin and Welch [135] proposed the second procedure, calling it a "postsolve". The first type of restoration procedure is excluded in our case because we use an interior-point based LP algorithm. We will therefore not discuss this approach further.

The approach of the postsolve procedure is first to undo the last reduction made by the presolve procedure in such a manner that, if the primal and dual solutions to the reduced problem are optimal and feasible, then the restored solution is also optimal and feasible. This process is then repeated until the optimal solution to the original problem is recovered.

To do the postsolving correctly and efficiently it is necessary that the presolve procedure store some information on a presolve "stack". In the following we assume that all the necessary information is available from the presolve stack.

It should be noted that whenever we write $\sum_i y_i^* a_{ij}$ it means that the sum is over the rows in the current problem.

An empty row: No primal postsolving is necessary and y_i^* can be set to any value, for example 0.

A singleton row: Assume a_{ij} is the only element in the ith row. No primal postsolving is necessary, because x_j is fixed at its optimal value. The value

$$y_i^* = (c_j - \sum_{k \neq i} y_k^* a_{kj})/a_{ij} \quad \text{and} \quad z_j^* = 0. \tag{3.50}$$

is dual feasible and optimal.

An empty column, or a fixed variable: The optimal Lagrange multiplier $z_j^* = c_j - \sum_i y_i^* a_{ij}$ for the variable must be computed.

A free column singleton or an implied free column singleton: Assume the last reduction made by the presolve procedure is to use the column singleton a_{ij} to eliminate variable x_j and the ith constraint. The optimal value x_j^* can be computed using (3.10). The dual optimal value is given by $y_i^* = c_j/a_{ij}$ and $z_j^* = 0$.

A doubleton equation combined with a column singleton: In this case the relations

$$a_{ij}x_j + a_{ik}x_k = b_i, \quad l_j \leq x_j \leq u_j \quad \text{and} \quad l_k \leq x_k \leq u_k \tag{3.51}$$

are replaced by

$$a_{ij}x_j + a_{ik}x_k = b_i \quad \text{and} \quad \tilde{l}_k \leq x_k \leq \tilde{u}_k \tag{3.52}$$

where a_{ij} is a column singleton. $\tilde{l}_k$ and $\tilde{u}_k$ are chosen so that the set of primal feasible solutions to (3.51) and (3.52) are identical and variable x_j is substituted out of the problem. The optimal x_j^* is given by

$$x_j^* = (b_i - a_{ik}x_k^*)/a_{ij}. \tag{3.53}$$

Let $\tilde{z}_k^*$ be the optimal Lagrange multiplier corresponding to x_k^* obtained from the reduced problem. The optimal Lagrange multiplier y_i^* is not known. However, we have

$$\begin{aligned} \tilde{z}_k = & \quad c_k - (c_j/a_{ij})a_{ik} - \sum_{p \neq i} y_p^* a_{pk} \\ = & \quad c_k - \sum_p y_p^* a_{pk} \end{aligned} \tag{3.54}$$

where $y_i^* = c_j/a_{ij}$. (3.54) follows as a result of x_j being substituted out of the problem. This value for y_i^* is not necessarily optimal and feasible if $x_k^* = \tilde{l}_k > l_k$ or $x_k^* = \tilde{u}_k < u_k$. In these cases the dual optimal value is

$$y_i^* = (c_k - \sum_{p \neq i} y_p^* a_{pk})/a_{ik} \quad \text{and} \quad z_k^* = 0. \tag{3.55}$$

A forcing constraint: A forcing constraint forces its variables to be at one of their bounds. Assume, for simplicity, that the ith constraint forces all its variables to be at their lower bound. This implies that either $a_{ij} \leq 0$ or $a_{ij} \geq 0$ for all j. The Lagrange multiplier y_i^* must be chosen such that

$$0 \leq z_j^* = c_j - \sum_{p \neq i} y_p^* a_{pj} - y_i^* a_{ij} \tag{3.56}$$

for all j, where $a_{ij} \neq 0$. If $a_{ij} > 0$ for all j, then the feasible y_i^* is given by

$$y_i^* \leq \min_{j \in \{k : a_{ik} > 0\}} ((c_j - \sum_{p \neq i} y_p^* a_{pj})/a_{ij}). \tag{3.57}$$

Otherwise, if $a_{ij} < 0$ for all j, then

$$y_i^* \geq \max_{j \in \{k : a_{ik} < 0\}} ((c_j - \sum_{p \neq i} y_p^* a_{pj})/a_{ij}). \tag{3.58}$$

Note that there always exists a feasible value for y_i^*. Indeed, there are an infinite number of feasible values which can be assigned to y_i^*. We have treated a special case, but it is easy to generalize it to arbitrary bounds and constraints. Postsolving of a forcing constraint has been treated in [135].

A dominated constraint: A dominated constraint is a constraint containing one implied free column singleton. How to postsolve this case has already been treated.

A dominated column or a weakly dominated column: If column j is a dominated column it is sufficient to compute $z_j^* = c_j - \sum_i y_i^* A_{ij}$.

Duplicate rows: If two rows are duplicate one of the rows is added to the other. If the new row is empty, containing an implied free column singleton, or is a doubleton equation containing a column singleton, it can be eliminated. The method to postsolve these cases has already been treated. However, the linear transformation on A must also be postsolved. Let the non-singular matrix M represent the transformation. This transformation does not affect the primal solution but the dual solution is affected, because the new dual constraints are

$$(A^T M)\tilde{y} + z = c \quad \text{and} \quad \tilde{y} = M^{-1}y \tag{3.59}$$

where y and $\tilde{y}$ are the Lagrange multipliers in the original and transformed problem, respectively. When the optimal $\tilde{y}^*$ to the reduced problem is known y^* can be computed by multiplying $\tilde{y}$ with M. Note that M can be represented by two indices and a real, because it is an elementary matrix (one row is added to another).

Fixing a duplicate column: Same as a fixed variable.

Replacement of two duplicate columns by one: Assume that the columns $a_{.j}$ and $a_{.k}$ are duplicate and that they have been replaced by variable x'_k in the reduced problem. From (3.48) and (3.49) we have the relations

$$\left\{ \begin{array}{l} x_k^* = -vx_j^* + x_k'^* \\ l_j \leq x_j^* \leq u_j \\ l_k \leq x_k^* - vx_j^* \leq u_k \end{array} \right. \tag{3.60}$$

where $x_k'^*$ is the optimal value as obtained from the reduced problem. Any value for x_j^* and x_k^* which satisfies (3.60) is primal feasible and optimal, but the values should be chosen such that complementary slackness conditions are satisfied.

We have now outlined how to recover both primal and dual optimal feasible solutions if an optimal solution to the reduced problem is known. It should be noted that the amount of work spent in the postsolve phase in all cases is modest.

3.5 Implementation

In this section we will discuss our implementation of the presolve procedure.

The presolve procedure assumes that A is stored sparse in a row-wise and column-wise format and that the objective function c, the simple bounds l and u are stored in dense vectors. Furthermore, a Boolean flag is tagged to each constraint and variable. This flag has the value "false" if the constraint or the variable has been eliminated from the problem. Whenever a constraint or a variable is mapped off it is treated as non-existent by the presolve procedure. Therefore during the presolve phase it is not necessary to change the data structure of A.

The presolve procedure can be outlined as follows:

```
Procedure Presolve
    Remove all fixed variables.
    repeat
        Check rows
            Remove all row singletons
            Remove all forcing constraints
        Dominated constraints
            Remove all dominated constraints
        Check columns
            Remove all free, implied free column singletons and
            all column singletons in combination with a doubleton equation
        Dominated columns
        Duplicate rows
        Duplicate columns
    until( no reductions in last pass )
    Remove all empty rows and columns
end Presolve.
```

The interpretation of the above presolve procedure is that each subprocedure makes a pass through A. During the pass the redundancy is discovered and removed. The presolve procedure is terminated when it is not possible to reduce the problem further: Finally empty rows and columns are removed. Furthermore, the procedure is terminated if it discovers that the problem is either primal or dual infeasible.

During the presolve phase the procedure informs the user of the kind of reductions it makes. (This is optional.) This information may be useful for the user to improve the model formulation.

3.6 Computational results

Can the proposed presolve procedure really reduce the size of an LP problem and is the presolve phase worthwhile? These are the main questions we will answer with a test.

Let us first outline the test environment. The presolve procedure is used as a front-end procedure to Mehrotra's predictor-corrector algorithm, see Mehrotra [106] and Lustig et al. [95]. The predictor-corrector algorithm is extended with the global convergence ideas of Kojima et al. [84] (see also Lustig et al. [96]). The most expensive operation in the primal-dual algorithm is the solution of the least-squares problem in each iteration. This is solved using a supernodal column Cholesky decomposition and the ordering is obtained with the multiple minimum-degree heuristic (see Lustig et al. [94] for a discussion of details). The source code is written in ANSI C and all time results are obtained with the C procedure "clock" which measures the time in number of CPU seconds. The testing is conducted on an IBM RISC 6000/230 workstation and on a Convex 3240 vector computer both at Odense University. On the workstation the code is compiled with the standard AIX compiler using the -O option. On the Convex the Convex C compiler is used with the option -O2, meaning that the code is vectorized. Furthermore, compiler directives are inserted in the source code to obtain a higher degree of vectorization.

It should be noted that the benefit of the presolve procedure is dependent on the efficiency of the LP code, because an inefficient LP code increases the benefit of the presolve procedure. Another warning is that the results are dependent on the computer architecture, because the presolve phase consists of operations which cannot be vectorized. Therefore, the relative benefit of the presolve phase is expected to be greater in a workstation environment than in a vector computer environment.

In the first part of the test, all the small NETLIB problems [56] were solved on the IBM RISC 6000/230 computer. By "small" we mean that they can be solved fast on the workstation.

In Table 3.5 the optimal primal objective value, XFEAS and ZFEAS for each problem are shown where

$$XFEAS = \max(\|Ax^* - b\|_\infty, -\min_{j \in L}(x_j^* - l_j), -\min_{j \in U}(u_j - x_j^*)) \tag{3.61}$$

and

$$ZFEAS = \max(-\min_{j \in L \setminus U}(z_j^*), \max_{j \in U \setminus L}(z_j^*), \max_{j \notin L \cup U}(|z_j^*|), 0.0). \tag{3.62}$$

XFEAS and ZFEAS measure the maximal violation by primal and dual variables of theirs simple bounds. In the case of an equality constraint in the primal or dual problem it is assumed that the constraint is formulated with a fixed slack variable. Moreover Table 3.5 shows the number of significant figures in the optimal objective value in the column SIGF. SIGF is computed as follows

$$\text{SIGF} = \left| \log_{10}\left(\frac{|\text{primal obj. - dual obj.}|}{1.0 + |\text{dual obj.}|} \right) \right|. \tag{3.63}$$

We want to point out that the results are report after the postsolve has been done, i.e., for the original problem.

In general, at least 8 digits in the optimal primal objective value are identical with the results presented in Bixby [20]. Furthermore, XFEAS and ZFEAS are small for all the problems. The only exception is the problem GREENBEA where XFEAS is 1E-2.

Tables 3.6 and 3.7 show the number of rows, columns and non-zeros in A before and after the presolve procedure is applied to the problems. The number of columns does not include slack variables. The last three columns in the tables 3.6 and 3.7 show the number of interior-point iterations, the total solution time and the time spent in the presolve phase. The total solution time includes everything except time spent reading the problem and writing the solution.

Title	POBJ	SIGF	XFEAS	ZFEAS	Title	POBJ	SIGF	XFEAS	ZFEAS
25FV47	5.50185E+03	10	6E-11	8E-12	PEROLD	-9.3808E+03	8	1E-08	2E-10
80BAU3B	9.87224E+05	8	5E-11	1E-14	PILOT.JA	-6.1131E+03	10	2E-06	1E-12
ADLITTLE	2.25495E+05	8	9E-12	2E-13	PILOT.WE	-2.7201E+06	8	4E-07	5E-06
AFIRO	-4.6475E+02	8	3E-14	0E+00	PILOT4	-2.5811E+03	10	2E-07	4E-10
AGG	-3.5992E+07	11	2E-07	5E-11	PILOTNOV	-4.4973E+03	10	2E-08	3E-26
AGG2	-2.0239E+07	8	2E-09	1E-07	RECIPE	-2.6662E+02	9	7E-12	1E-12
AGG3	1.03121E+07	16	2E-08	2E-10	SC105	-5.2202E+01	8	2E-12	0E+00
BANDM	-1.5863E+02	9	2E-11	7E-09	SC205	-5.2202E+01	8	2E-11	0E+00
BEACONFD	3.35925E+04	8	4E-11	9E-16	SC50A	-6.4575E+01	9	3E-12	0E+00
BLEND	-3.0812E+01	9	3E-13	5E-12	SC50B	-7.0000E+01	9	1E-13	0E+00
BNL1	1.97763E+03	10	2E-09	2E-11	SCAGR25	-1.4753E+07	10	2E-08	2E-07
BNL2	1.81124E+03	10	2E-06	7E-10	SCAGR7	-2.3314E+06	9	2E-10	2E-07
BOEING1	-3.3521E+02	8	2E-12	5E-13	SCFXM1	1.84168E+04	10	8E-12	2E-10
BOEING2	-3.1502E+02	10	6E-13	0E+00	SCFXM2	3.66603E+04	8	3E-10	1E-08
BORE3D	1.37308E+03	9	5E-12	1E-10	SCFXM3	5.49013E+04	8	7E-10	6E-09
BRANDY	1.51851E+03	9	4E-10	8E-11	SCORPION	1.87812E+03	10	3E-13	3E-14
CAPRI	2.69001E+03	8	2E-10	1E-14	SCRS8	9.04297E+02	10	2E-07	4E-09
CYCLE	-5.2264E+00	9	4E-10	5E-12	SCSD1	8.66667E+00	10	6E-13	0E+00
CZPROB	2.18520E+06	9	4E-12	0E+00	SCSD6	5.05000E+01	9	1E-11	0E+00
D2Q06C	1.22784E+05	8	7E-06	3E-08	SCSD8	9.05000E+02	8	2E-13	0E+00
DEGEN2	-1.4352E+03	8	1E-11	0E+00	SCTAP1	1.41225E+03	8	2E-13	0E+00
DEGEN3	-9.8729E+02	8	6E-11	0E+00	SCTAP2	1.72481E+03	10	3E-13	0E+00
E226	-1.8752E+01	10	9E-13	4E-10	SCTAP3	1.42400E+03	10	2E-14	0E+00
ETAMACRO	-7.5572E+02	8	9E-11	8E-16	SEBA	1.57116E+04	8	5E-13	1E-15
FFFFF800	5.55680E+05	9	1E-09	5E-15	SHARE1B	-7.6589E+04	9	1E-07	7E-16
FINNIS	1.72791E+05	8	9E-08	2E-11	SHARE2B	-4.1573E+02	8	1E-12	0E+00
FIT1D	-9.1464E+03	9	0E+00	0E+00	SHELL	1.20883E+09	11	5E-08	0E+00
FIT2D	-6.8464E+04	8	4E-12	0E+00	SHIP04L	1.79332E+06	8	1E-07	0E+00
FORPLAN	-6.6422E+02	10	1E-06	0E+00	SHIP04S	1.79871E+06	8	5E-12	0E+00
GANGES	-1.0959E+05	9	1E-06	0E+00	SHIP08L	1.90906E+06	8	7E-11	0E+00
GFRD-PNC	6.90224E+06	8	5E-10	0E+00	SHIP08S	1.92010E+06	9	3E-12	0E+00
GREENBEA	-7.2555E+07	9	1E-02	4E-07	SHIP12L	1.47019E+06	10	3E-11	0E+00
GREENBEB	-4.3023E+06	8	2E-09	1E-06	SHIP12S	1.48924E+06	10	8E-12	0E+00
GROW15	-1.0687E+08	11	6E-07	0E+00	SIERRA	1.53944E+07	9	9E-12	0E+00
GROW22	-1.6083E+08	16	1E-06	0E+00	STAIR	-2.5127E+02	10	1E-10	1E-12
GROW7	-4.7788E+07	9	3E-08	0E+00	STANDATA	1.25770E+03	8	4E-15	5E-10
ISRAEL	-8.9664E+05	8	0E+00	2E-08	STANDMPS	1.40602E+03	9	2E-12	0E+00
KB2	-1.7499E+03	9	3E-11	4E-11	STOCFOR1	-4.1132E+04	9	7E-09	4E-14
LOTFI	-2.5265E+01	8	3E-08	1E-11	STOCFOR2	-3.9024E+04	9	3E-09	4E-14
MAROS	-5.8064E+04	9	9E-08	4E-06	TUFF	2.92148E-01	8	7E-10	2E-18
NESM	1.40760E+07	8	3E-08	2E-07	VTP.BASE	1.29831E+05	9	3E-11	4E-08

Tabel 3.5: Optimality results

Title	Rows	Cols.	Red. rows	Red. cols.	NNZ (A)	Red. NNZ(A)	Ite.	Total solt.	Pre-solt.
25FV47	822	1571	767	1534	11127	10106	29	22.21	0.54
80BAU3B	2263	9799	1960	8679	29063	18969	53	86.67	3.27
ADLITTLE	57	97	53	95	465	373	13	0.30	0.01
AFIRO	28	32	22	30	88	75	9	0.09	0.01
AGG	489	163	150	109	2541	794	20	1.95	0.12
AGG2	517	302	284	254	4515	2279	18	3.66	0.22
AGG3	517	302	287	254	4531	2315	19	3.83	0.22
BANDM	306	472	214	369	2659	1810	20	2.42	0.10
BEACONFD	174	262	82	138	3476	1186	10	0.74	0.11
BLEND	75	83	67	75	521	424	15	0.43	0.03
BNL1	644	1175	562	1108	6129	4823	37	12.18	0.26
BNL2	2325	3489	1697	2894	16124	12081	40	97.19	1.87
BOEING1	352	384	290	372	3865	2718	29	4.80	0.17
BOEING2	167	143	122	143	1339	794	21	1.30	0.05
BORE3D	234	315	77	105	1525	512	19	0.98	0.06
BRANDY	221	249	121	198	2150	1707	18	1.64	0.10
CAPRI	272	353	241	307	1786	1399	23	2.82	0.08
CYCLE	1904	2857	1281	2168	21322	14018	35	34.06	1.17
CZPROB	930	3523	661	2688	14173	5376	39	13.44	0.42
D2Q06C	2172	5167	2015	4999	35674	31071	36	192.20	1.63
DEGEN2	445	534	440	533	4449	3969	16	7.61	0.10
DEGEN3	1504	1818	1502	1817	26230	24644	34	146.83	0.43
E226	224	282	158	258	2767	2292	25	2.44	0.08
ETAMACRO	401	688	333	542	2489	1852	32	7.17	0.12
FFFFF800	525	854	303	646	6235	4878	32	8.20	0.19
FINNIS	498	614	356	423	2714	1468	26	4.31	0.33
FIT1D	25	1026	24	1024	14430	13386	26	6.44	0.51
FIT2D	26	10500	25	10363	138018	127760	32	80.79	4.47
FORPLAN	162	421	102	364	4916	4063	24	2.63	0.16
GANGES	1310	1681	988	1349	7021	6133	17	9.85	1.40
GFRD-PNC	617	1092	590	1066	3467	2325	17	3.71	0.14
GREENBEA	2393	5405	1848	3888	31499	23385	38	61.09	1.83
GREENBEB	2393	5405	1846	3876	31499	23289	49	72.55	2.14
GROW15	301	645	300	645	5665	5620	13	3.81	0.13
GROW22	441	946	440	946	8318	8252	14	5.85	0.18
GROW7	141	301	140	301	2633	2612	13	1.68	0.05
ISRAEL	175	142	163	141	2358	2256	26	6.48	0.08
KB2	44	41	42	40	291	284	15	0.25	0.01
LOTFI	154	308	117	282	1086	596	15	0.83	0.11
MAROS	847	1443	617	941	10006	5980	30	11.50	0.56
NESM	663	2923	599	2664	13988	12820	38	24.90	0.66
Sum	27758	67807	21886	58628	483152	390694		951.83	24.12

Tabel 3.6: Results for the small NETLIB problems.

Title	Rows	Cols.	Red. rows	Red. cols.	NNZ (A)	Red. NNZ(A)	Ite.	Total solt.	Pre-solt.
PEROLD	626	1376	590	1234	6026	5478	36	19.14	0.27
PILOT.JA	941	1988	807	1555	14706	11152	38	46.31	0.92
PILOT.WE	723	2789	700	2495	9218	8521	56	27.75	0.44
PILOT4	411	1000	390	865	5145	4813	33	13.83	0.33
PILOTNOV	976	2172	845	1870	13129	11659	25	34.06	0.88
RECIPE	92	180	61	95	752	410	12	0.36	0.03
SC105	106	103	104	103	281	280	10	0.34	0.02
SC205	206	203	203	202	552	550	11	0.72	0.03
SC50A	51	48	49	48	131	130	11	0.19	0.00
SC50B	51	48	48	48	119	118	10	0.17	0.01
SCAGR25	472	500	341	492	2029	1410	18	2.51	0.13
SCAGR7	130	140	89	132	553	366	17	0.63	0.03
SCFXM1	331	457	272	425	2612	2274	20	2.71	0.13
SCFXM2	661	914	544	850	5229	4553	22	6.31	0.28
SCFXM3	991	1371	816	1275	7846	6832	22	9.02	0.39
SCORPION	389	358	232	241	1708	1060	16	1.55	0.12
SCRS8	491	1169	418	1100	4029	2934	29	5.82	0.55
SCSD1	78	760	77	760	3148	2388	12	1.04	0.08
SCSD6	148	1350	147	1350	5666	4316	12	1.93	0.14
SCSD8	398	2750	397	2750	11334	8584	11	3.82	0.26
SCTAP1	301	480	269	452	2052	1557	19	2.20	0.15
SCTAP2	1091	1880	977	1768	8124	6159	19	9.49	0.38
SCTAP3	1481	2480	1344	2356	10734	8219	20	13.39	0.50
SEBA	516	1028	2	8	4874	11	7	1.04	0.47
SHARE1B	118	225	107	217	1182	1112	23	1.04	0.03
SHARE2B	97	79	93	79	730	691	14	0.50	0.02
SHELL	537	1775	487	1448	4900	2902	27	6.10	0.17
SHIP04L	403	2118	292	1890	8450	4275	17	3.69	0.18
SHIP04S	403	1458	216	1266	5810	2860	17	2.72	0.13
SHIP08L	779	4283	470	3099	17085	7100	18	7.23	0.35
SHIP08S	779	2387	276	1582	9501	3622	17	4.16	0.22
SHIP12L	1152	5427	609	4147	21597	9222	19	9.63	0.48
SHIP12S	1152	2763	339	1919	10941	4265	18	5.19	0.27
SIERRA	1228	2036	1194	2016	9252	7206	23	13.02	0.31
STAIR	357	467	356	384	3857	3664	18	7.04	0.11
STANDATA	360	1075	292	441	3038	1026	20	2.54	0.22
STANDMPS	468	1075	394	1017	3686	2418	37	5.97	0.21
STOCFOR1	118	111	94	96	474	359	14	0.51	0.02
STOCFOR2	2158	2031	1968	1854	9492	7064	24	19.14	0.48
TUFF	334	587	246	513	4523	4013	21	4.18	0.22
VTP.BASE	199	203	49	78	914	227	12	0.43	0.05
WOOD1P	245	2594	170	1717	70216	44573	28	32.64	1.33
WOODW	1099	8405	555	4073	37478	14599	30	27.01	1.22
Sum	23647	64643	17929	50310	343123	214972		357.07	12.56
Total	51405	132450	39815	108938	826275	605666		1308.90	36.68

Tabel 3.7: Results for the small NETLIB problems continued.

Method \Run	1	2	3	4	5	6	7
Simple methods	-	+	+	+	+	+	+
Dominated rows	-	-	-	+	+	+	+
Dominated columns	-	-	+	-	+	+	+
Duplicate rows	-	-	+	+	-	+	+
Duplicate columns	-	-	+	+	+	-	+

Tabel 3.8: Methods included in presolve runs. + means included.

Run	Red. rows	Red. cols.	Red. NNZ(A)	Prs-solt.
1	46722	127416	735685	4.7
2	41993	114601	633034	12.26
3	40912	110876	617692	29.27
4	40312	110991	613232	30.31
5	39916	108938	606038	35.61
6	40487	111939	618516	23.99
7	39815	108938	605666	35.8

Tabel 3.9: Results from 7 different level of presolve.

All problems are reduced by at least one row, because the free row is transferred to the objective function. From Table 3.7 we see that the total number of rows, colunms and non-zeros are reduced from 51405, 132450 and 826275 to 39815, 108938 and 605666, respectively. In all three cases the reduction is approximately 20%. Only the problems FIT1D, GROW* and SCSD* are not reduced at all.

From Table 3.7 we see that all the problems are solved in 1308 CPU seconds and 37 of these are spent in the presolve phase. This amounts to approximately 3% of the total solution time.

It is interesting to evaluate the efficiency of the computationally expensive dominated and duplicate rows and columns presolve subprocedures. We have therefore applied the presolve procedure seven times to the problems where different parts of the presolve procedure are turned off. In Table 3.8 we list the presolve subprocedures which are turned on in each run. In run 1 only empty rows, singleton rows, fixed variables and empty columns are detected and removed. In run 2 the complete presolve procedure is used, but without the dominated and duplicate rows and columns procedures. Finally in run 7 the complete presolve procedure is used.

Table 3.9 shows the total number of rows, columns and non-zeros in A in all the problems after each run. The last column in Table 3.9 shows the total time spent in the presolve phase.

If the results of run 1 and run 2 are compared we observe that the total number of rows are reduced by 10% if the simple presolve methods are used, compared to not using the presolve procedure. Furthermore, the time spent in the presolve procedure is only increased slightly (in absolute terms).

From Table 3.9 it follows that all parts of the presolve procedure have an effect, but the greatest reduction is obtained with the simple presolve methods. Of the computationally expensive subprocedures, the dominated rows procedure best at reducing problem size, although the dominated columns and duplicate columns procedure follow closely. Furthermore, the time spent in the presolve phase cannot be reduced significantly by turning one of the presolve subprocedures off.

If one of the presolve subprocedures should be turned off due to its computational costs it should be the duplicate columns procedure. The main justification for the duplicate columns procedure is that it discovers any split free variable, but if no such variables are discovered, the value of the procedure is questionable. In Table 3.10 the number of free variables in each problem is listed before and after the presolve phase (run 7). We have not listed results for problems where the number of free variables were unchanged. From Table 3.10 we see that the presolve procedure eliminates all free variables in the problems CAPRI, TUFF and VTP.BASE whereas it discovers some in the problems BRANDY, GREENBEA, GREENBEB, SCFXM* and STAIR. If these split free variables in the problems are not identified they typically cause serious numerical problems. Since the computational cost of the duplicate

Title	Free	Red. Free	Title	Free	Red.
BRANDY	0	4	SCFXM2	0	8
CAPRI	14	0	SCFXM3	0	12
GREENBEA	0	6	STAIR	6	7
GREENBEB	4	7	TUFF	2	0
SCFXM1	0	4	VTP.BASE	1	0

Tabel 3.10: Free variables before and after presolve.

Title \Run	1	2	3	4	5	6	7
25FV47	24,30,10	23,29,8	22,29,11	22,29,11	22,28,8	23,29,10	22,29,11
80BAU3B	78,51,8	84,52,8	82,51,8	88,53,8	86,53,8	91,56,8	86,53,8
BNL2	102,41,10	98,40,10	99,41,10	100,41,10	97,40,10	97,40,10	97,40,10
CYCLE	56,36,9	46,35,8	44,37,9	46,39,10	34,35,9	32,33,9	34,35,9
CZPROB	17,37,8	12,35,8	13,39,9	12,35,8	13,39,9	13,39,9	13,39,9
D2Q06C	201,38,10	210,38,10	197,37,11	168,36,8	189,36,10	201,38,10	192,36,8
DEGEN3	140,32,8	147,34,8	147,34,8	147,34,8	147,34,8	147,34,8	147,34,8
FIT1D	6,25,9	6,25,9	6,26,9	6,26,9	6,26,9	6,25,9	6,26,9
FIT2D	77,32,9	78,32,9	80,32,8	80,32,8	81,32,8	79,32,9	81,32,8
GREENBEA	215,86,1	132,84,2	62,39,8	61,38,10	61,38,10	68,43,2	61,38,10
GREENBEB	117,50,2	83,52,2	69,46,8	71,48,8	72,49,8	72,45,1	73,49,8
MAROS	18,31,8	12,32,9	11,29,9	11,30,9	11,30,9	12,31,10	12,30,9
NESM	22,34,8	22,34,8	23,35,8	25,38,8	24,36,8	24,37,8	25,38,8
PILOT.JA	48,39,9	50,38,10	47,36,8	46,38,10	46,38,10	45,37,8	46,38,10
PILOTNOV	37,27,8	34,25,10	32,25,8	34,25,10	34,25,10	34,25,10	34,25,10
SCSD8	4,11,8	4,11,8	4,11,8	4,11,8	4,11,8	4,11,8	4,11,8
SCTAP3	14,20,10	13,20,10	13,20,10	13,20,9	13,20,10	13,20,10	13,20,10
SHIP12L	14,20,8	9,18,8	10,19,10	10,19,10	9,18,8	10,19,10	10,19,10
WOOD1P	41,25,7	34,29,8	33,28,9	32,28,9	33,28,9	34,29,8	33,28,9
WOODW	46,28,9	30,29,9	31,29,9	32,31,9	27,30,8	30,34,9	27,30,8
Sum	1275,-,-	1126,-,-	1024,-,-	1008,-,-	1010,-,-	1034,-,-	1016,-,-

Tabel 3.11: Results of 7 different runs. (Solution time, number of iterations, significant figures in the optimal value.)

columns procedure is not large, it is a cheap way to guard against one type of bad formulation which often appears. (This is documented below).

A real test of the efficiency of the presolve procedure is, of course, to solve the problems with and without the presolve procedure and evaluate the change in the total solution time.

For this part of the test we used all the small NETLIB problems with more than 10000 non-zeros in A, but only one of the SHIP* problems. These problems were solved using the 7 different levels of presolve. The results are presented in Table 3.11.

In Table 3.11 the total solution time for each problem in each run is presented, and the number of iterations and the number of significant figures in the optimal objective value are shown. The effect of using the presolve procedure can be studied by comparing the results of runs 1 and 7. We see that the total solution time is reduced from 1274 to 1015 CPU seconds by using the presolve procedure. In addition, it can be noted that for 14 out of 20 problems the total solution time is reduced and in some cases the reduction is dramatic, e.g., CYCLE and WOODW. The solution time can be decreased in some cases by turning the presolve procedure off, but never by more than 11% (80BAU3B).

An important observation is that the standard precision, which is 8 significant figures in the optimal value, is not achieved for the problems GREENBEA, GREENBEB and WOOD1P in run 1, whereas it is achieved in run 7. That means the presolve procedure in some cases improves the formulation of the

Title	Rows	Cols.	Red. rows	Red. cols.	NNZ (A)	Red. NNZ(A)	Ite.	Total solt.	Pre-solt.
CRE-A	3517	4067	2786	3938	19054	13680	34	109.13	1.47
CRE-B	9649	72447	5324	31818	328542	107603	56	1127.72	15.08
CRE-C	3069	3678	2328	3249	16922	11296	36	100.02	1.72
CRE-D	8927	69980	4089	25188	312626	83223	54	890.23	16.22
KEN-07	2427	3602	1427	2603	11981	5494	16	25.28	0.93
KEN-11	14695	21349	10067	16715	70354	36310	23	329.70	7.02
KEN-13	28633	42659	22525	36552	139834	81168	31	1149.80	11.57
KEN-18	105128	154699	78841	128413	512719	292033	37	9208.90	42.13
OSA-07	1119	23949	1047	23015	167643	61990	34	123.62	9.43
OSA-14	2338	52460	2266	50457	367220	136870	34	276.50	20.18
OSA-30	4351	100024	4279	96119	700160	262872	30	521.47	39.92
OSA-60	10281	232966	10209	224125	1630758	584253	38	1694.27	105.61
PDS-02	2954	7535	2605	7172	21252	15543	32	97.43	1.62
PDS-06	9882	28655	9132	27852	82269	60236	46	1366.90	6.92
PDS-10	16559	48763	15609	47729	140063	103290	59	6726.97	12.88
PDS-20	33875	105728	32203	103981	304153	224418	70	51391.68	33.18
FIT2D	26	10500	25	10363	138018	127760	32	57.90	10.40
PILOT	1442	3652	1362	3334	43220	40592	45	341.25	6.47
PILOT87	2031	4883	1966	4592	73804	70375	47	993.08	12.52
Total	260903	991596	208090	847215	5080592	2319006		76531.84	355.25

Tabel 3.12: Results for the large NETLIB problems.

problem. The reason why GREENBEA and GREENBEB cannot be solved to the required precision in run 1 is that a few split free variables are not discovered.

The first part of the test shows that the presolve phase is advantageous for relatively small LP problems. In order to evaluate whether this conclusion can be extended to large sparse problems we solved the large NETLIB problems. This part of the test was conducted on the Convex vector computer.

In Table 3.12 the names of the test problems and the test results are presented. The first 13 problems are a subset of the so-called Kennington problems, which have been used to evaluate AT&T's KORBX LP solver, see [29].

It can be seen from Table 3.12 that both the total number of rows, columns and non-zeros in A can be reduced significantly, although it should be noted that especially the CRE* and KEN* problems are responsible for the reduction. The interesting question is, of course, whether the presolve phase is too computationally expensive on large sparse problems. For these test problems this is not the case, because the total solution time is 76532 CPU seconds, and the time spent in the presolve phase is 355 CPU seconds, which amounts to 0.5%. We therefore conclude that a presolve phase is also beneficial for large sparse problems. However, this might not be true for dense problems. This is documented with the problem FIT2D, because 20% of the solution time for this problem is spent in the presolve phase. The reason for the bad performance on this problem is that it is dense and that means the Convex's vector facility is exploited very well in the solution phase, whereas it is not exploited at all in the presolve phase. We have also solved FIT2D in the previous test conducted on a IBM RISC/6000 workstation and it can be verified in Table 3.6 that in this test the presolve phase is not as expensive, relative to the solution phase.

In the last test we investigate whether the presolve procedure can detect an infeasible status of an infeasible LP problem. The test problems for this part of the test were kindly provided by John W. Chinneck and they are all primal infeasible.

The test problems and our results are presented in Table 3.13. The problem GREENBEA in Table 3.13 is an infeasible version of the NETLIB problem GREENBEA. It should be noted that we do not know whether the problems should be minimized or maximized, so we have chosen to minimize all the problems except CPLEX1. This problem is dual infeasible if it is minimized. Clearly the results depend on whether the problems are minimized or maximized, because the presolve procedure uses dual information

Title	Rows	Cols	Red. rows	Red. cols.	Pre-solt.	Status
CERIA3D	3577	824	6	7	1.02	INIT
CHEMCOM	289	720	288	720	0.18	INIT
CPLEX1	3006	3221	2505	3221	1.35	INIT
FOREST6	67	95	66	95	0.03	INIT
GALENET	9	8	4	6	0.02	INFE
GREENBEA	2505	5405	2113	4730	0.67	INFE
PAPER2	10	4	7	4	0.02	INFE
PAPER1	12	8	8	8	0.00	INFE
KLEIN1	55	54	54	54	0.03	INIT
KLEIN2	478	54	476	54	0.22	INIT
KLEIN3	995	88	993	88	0.50	INIT
MONDOU	313	604	259	466	0.17	INIT
QUAL	324	464	189	312	0.20	INIT
REFINERY	324	464	189	312	0.22	INIT
VOL1	324	464	189	312	0.20	INIT
VOL2	324	464	189	312	0.20	INIT
WOODINFE	36	89	35	89	0.02	INFE

Tabel 3.13: Results for infeasible problems. (INFE means the problem is infeasible. INIT means the optimal status is unknown.)

to reduce the problem.

Unfortunately, our presolve procedure is not very succesful in this test, because only the 5 problems GALENET, GREENBEA, PAPER1, PAPER2 and WOODINFE are detected infeasible.

We conclude that the presolve procedure is not an efficient method for detecting infeasibility. Although it may discover infeasibility in some (easy) cases.

3.7 Conclusion

The numerical results presented in this paper demonstrate that for large sparse LP problems a presolve phase is worthwhile, although this conclusion might not be true for dense problems.

It should be noted that this conclusion is only valid if the presolve procedure is used in combination with an interior-point algorithm. However, with slight changes, the presolve procedure is beneficial for use in combination with a simplex algorithm, see [21].

Another possible use of the presolve procedure is as a model analyzing tool, because the procedure might reveal bad formulations such as primal and dual infeasibility. The possible detection of infeasible problems, is especially advantageous for a primal-dual interior-point algorithm, because the algorithm has some difficulties with such problems, (see Lustig et al. [96]). Our results on this possibility are not very promising and the presolve procedure should be improved in that respect.

If the presolve procedure should be extended, a computational cheap method for detecting linearly dependent constraints is desirable. Our experience shows that such constraints exist. Such a method is discussed in Andersen [6].

Acknowledgements The authors wish to thank Arne Drud for valuable comments to this paper and John W. Chinneck for providing his infeasible test problems.

Kapitel 4

Finding all linearly dependent rows in large-scale linear programming

All the new interior-point algorithms for linear programming (LP) assume that the rows of the constraint matrix are linearly independent. The assumption is necessary, because the algorithms solve a certain linear system which does not have a unique solution in the case of dependent rows. In this paper we study a pivoting based approach for finding all the linearly dependent rows. Moreover encouraging computational results are reported.

4.1 Introduction

Most of the new interior-point algorithms for LP compute a Cholesky decomposition of the matrix

$$M = ADA^T \tag{4.1}$$

where A is the LP constraint matrix and D is a positive definite diagonal matrix. If A is not of full row rank, then the matrix M is not of full rank. In this case the Cholesky decomposition is in general not unique, see for example Higham [73]. If pivoting is used during computation of the Cholesky decomposition all the dependent rows are detected.

However, in practice pivoting is not used because it destroys sparsity of the Cholesky factors. Therefore if during computation of the Cholesky decomposition the size of a diagonal element is less than a tolerance the corresponding row of A is treated as dependent. Unfortunately this dependency check is not reliable due to rounding errors. Moreover it is difficult to chose a universal good drop tolerance. It is therefore beneficial to remove the dependencies from the LP problem before submitting it to optimization.

Another benefit of removing the dependencies is the reduction in the size of the LP problem and dependent rows may create an infeasibility in the LP problem.

Even if the LP problem is optimized using the simplex algorithm dependent rows should be removed. This leads Tomlin and Welch [137] to find all duplicate rows. Duplicate rows are rows that are identical up to a scalar multiplier. Since duplicate rows can be found efficiently they are removed in the presolve phase, see Andersen and Andersen [7], Gondzio [64] and Suhl [130]. Gondzio propose a way of finding more dependent rows, but he does not try to find all of them, because it is computational too expensive, see [64, p. 11]. OB1 an interior-point LP solver has a capability to find all the linearly dependent rows by doing elementary row operations on A. This option is claimed to be useful for some large-scale LP problems arising in fleet assignment problems, see Subramanian et al. [129, p. 113].

The outline of the paper is as follows. In Section 4.2 we discuss a pivoting algorithm for finding the dependent rows. In Section 4.3 we propose an efficient implementation of this algorithm. In Section 4.4 we report numerical results on 9 large scale LP problems. In Section 4.5 we present our conclusions.

4.2 The pivoting algorithm

The problem is

$$\begin{array}{rl} \text{minimize} & c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array} \tag{4.2}$$

where $A \in R^{m \times n}$, $x, c \in R^n$ and $b \in R^m$. We use the following definition of dependent rows.

Definition 4.2.1 *Let $S \subseteq \{1, ..., m\}$ be a non-empty set. If there exists a vector $s \in R^m$ such that $s_i \neq 0$ for all $i \in S$ and*

$$\sum_{i \in S} s_i a_{i.} = 0$$

then the rows in A corresponding to the indices in S are said to be linearly dependent.

The notation $a_{i.}$ and $a_{.j}$ mean the ith row and jth column of A respectively. Now we can state Proposition 4.2.1.

Proposition 4.2.1 *Assume that the rows corresponding to the indices in S are linearly dependent. In this case either one of these rows can be removed from the problem without changing the optimal solution or the problem is infeasible.*

Note we can remove any row from a set of linearly dependent rows. In practice it is advantageous to remove the one that contains the most nonzeros.

It is well-known that all the dependent rows can be found using Gaussian elimination. This is done by reducing A to the following form

$$\begin{bmatrix} x & : & : & : & : & : \\ 0 & x & : & : & : & : \\ 0 & 0 & x & : & : & : \\ 0 & 0 & 0 & x & : & : \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.3}$$

using elementary row operations and permutations. The dependent rows will appear as zero rows at the bottom of the transformed matrix. The updated right hand side must be zero in the position corresponding to the zero rows otherwise the LP problem is infeasible.

The disadvantage of this procedure is that it does row operations on the whole A matrix. Even though A is sparse a lot of fill-ins may be generated during the computation. Instead it is possible only to work with a basis matrix and generate the parts of the updated matrix as needed. We will now describe this approach.

Assume a set of artificial columns is appended to A such that

$$A = [I, \hat{A}] \tag{4.4}$$

where $\hat{A}$ is the original LP constraint matrix. Let the set $\mathcal{V}$ contain the indices of the artificial columns. Furthermore assume A is an m by n matrix. Now the pivoting procedure can be stated as follows.

Algorithm 4.2.1

1. *Let* $\mathcal{B} = \mathcal{V}$
2. *for each* $i \in \mathcal{B} \cup \mathcal{V}$
3. $\quad \pi = e_i^T B^{-1}$
4. $\quad$ *If* $\exists j : \left|\pi^T a_{.j}\right| > 0$ *and* $j \notin \mathcal{B} \cup \mathcal{V}$
5. $\quad\quad \mathcal{B} = \mathcal{B} \cup \{j\} \setminus \{i\}$
6. $\quad$ *else*
7. $\quad\quad$ *The ith row is dependent.*
8. *end for*

We use the notation that $\mathcal{B}$ contains the indices of the basis columns and B is the corresponding matrix. e_i is the ith column of the identity matrix.

In step 4 the ith row of the updated matrix is generated. If this row is zero a linear dependency has been detected. Note that by construction $\|\pi\| \neq 0$.

Algorithm 4.2.1 is well-known, see Chvátal [35, p. 130]. However, implemented naively it may be expensive. It can be verified that the complexity of the algorithm is $O(nm^2)$. Note that reducing A to the form (4.3) using elementary row operations require a similar amount of work.

In the following section we discuss an efficient implementation of the algorithm.

4.3 Implementation

The computational cost of Algorithm 4.2.1 depend on the dimension of the basis and on the number of nonzeros in the basis. Both can be reduced by using two observations.

First observe that any zero rows in A can immediately be declared dependent and removed from the problem. Second in practice A often contains a lot of column singletons i.e. columns containing only one nonzero. For example slack variables are column singletons. Clearly a row containing a column singleton cannot be dependent. Therefore a row containing a column singleton can be removed from A before the linear dependency check. Note that when such a row is removed new column singletons may appear and more rows may be removed.

In Algorithm 4.2.1 the number of loops (step 2 to 8) depend on the number of artificial columns in the initial basis. Therefore the computational cost is decreased if some of the artificial columns are replaced by structural columns in the initial basis. It is therefore advantageous to *crash* an initial basis containing as many structural columns as possible. We use two heuristic to achieve this aim.

The first heuristic can be described as follows. We say a row is unassigned if a basic variable has not been assigned to that row. First count the number of nonzeros in each row and column. Assign an artificial variable to the row that contain the most nonzeros (ties are broken arbitrarily). Remove this row temporarily from the problem and update the nonzero count in each column. If any new column singletons appear then assign them to the row in which they occur. This may create new column singletons. Repeat this procedure until a structural or an artificial variable is assigned to each row. By construction the initial basis matrix will be upper triangular and therefore non-singular.

By assigning artificial columns to the most dense rows the heuristic tends to create as many column singletons as possible. Moreover, as a consequence the most dense rows among the dependent rows tend to be removed.

After using the first heuristic usually few rows are left unassigned. However, if more than 10% of the total number of rows are unassigned we assign the shortest nonbasic column in each unassigned row to the row. By shortest we mean containing fewest nonzeros. Afterwards we factorize the matrix and keep the non-singular part. A similar approach has been used by Bixby and Saltzman [23] in another context.

4.3.1 A specialized LU factorization

In Algorithm 4.2.1 we must solve a system of the form

$$B^T \pi = e_i \tag{4.5}$$

in each iteration. We propose to solve this linear system using a LU factorization of B where L is stored in product form. In our case it is possible to specialize the LU factorization to gain additional computational efficiency. Especially we will exploit the fact that the artificial columns can be driven out of the basis in any order.

Assume that there are r artificial columns in the basis, then we will reorder the rows and columns of B such that the last r columns are equal to the last r columns of the identity matrix. The first $m - r$ rows and columns are ordered using the Markowitz criterion, such that fill-ins are limited, see Duff et al.

[44]. After computation of the LU factorization of B, U has the form

$$U = \begin{bmatrix} x & : & : & : & 0 & 0 & 0 \\ 0 & x & : & : & 0 & 0 & 0 \\ 0 & 0 & x & : & 0 & 0 & 0 \\ 0 & 0 & 0 & x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.6}$$

Whenever we solve the system (4.5) we must first solve

$$U^T \bar{\pi} = e_i \tag{4.7}$$

and then solve the system

$$L^T \pi = \bar{\pi}. \tag{4.8}$$

If we remove the artificial column corresponding to the $(m-r)$th basic column it is easy to see that the solution to (4.7) is given by

$$\bar{\pi} = e_i, \tag{4.9}$$

i.e. the system (4.7) is trivial to solve. Afterwards we either keep the artificial column in the basis (in this case a linear dependency is detected) or the column is replaced with a structural column. In the latter case the LU decomposition must be updated. Using the idea of the Bartels-Golub LU update, the $(m-r)$th column is replaced by $L^{-1}a_{.j}$ where j is the index of the incoming column. The new U will have the form

$$U = \begin{bmatrix} x & : & : & : & : & 0 & 0 \\ 0 & x & : & : & : & 0 & 0 \\ 0 & 0 & x & : & : & 0 & 0 \\ 0 & 0 & 0 & x & : & 0 & 0 \\ 0 & 0 & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & : & 1 & 0 \\ 0 & 0 & 0 & 0 & : & 0 & 1 \end{bmatrix}. \tag{4.10}$$

before triangularity is restored. By construction the diagonal element in column $(m-r)$ is nonzero. It follows that this diagonal element can be used to eliminate sub-diagonal elements in the column. If the diagonal element is small in absolut size the row operations are potentially unstable, but in this case we can refactorize the matrix. We also refactorize the basis matrix occasionally to maintain sparsity and stability of the LU factors.

Another way to avoid numerical problems is to use the Bartels-Golub update. It should be noted that the matrix U is not used in the solve operation or in the LU update and it can therefore be discarded during the computation.

Because computations are done in finite precision Algorithm 4.2.1 should be modified slightly. Step 4 of the Algorithm 4.2.1 should be replaced by

$$\text{If} \quad \exists j : \left|\pi^T a_{.j}\right| \geq TOLAPIV \quad \text{and} \quad j \notin \mathcal{B} \cup \mathcal{V}.$$

TOLAPIV (=1.0e-8) is a parameter to be chosen. If we detect a linear dependency we have to check whether the LP problem is infeasible. Let $\bar{b}_i$ be the updated right hand side coresponding to a zero row then the problem is said to be infeasible if

$$\frac{|\bar{b}_i|}{1 + ||b||} > \text{TOLPRIMAL}$$

where TOLPRIMAL (=1.0e-8) is a parameter to be chosen.

Name	ROWS	COLS	NNZA	RROWS	RCOLS	RNNZA	LROWS	LCOLS	LNNZA
80BAU3B	2263	9799	29063	1960	8679	18969	1960	8679	18969
aa3	826	8627	79433	763	8572	67571	690	8572	58604
CRE-B	9649	72447	328542	5324	31818	107603	5316	31818	107551
KEN-13	28633	42659	139834	22525	36552	81168	22356	36552	79478
NUG12	3193	8856	44244	3192	8856	38304	2794	8856	33528
OSA-30	4351	100024	700160	4279	96119	262872	4279	96119	262872
PDS-10	16559	48763	140063	15609	47729	103290	15598	47729	103169
PILOT87	2031	4883	73804	1966	4592	70375	1966	4592	70375
WOOD1P	245	2594	70216	170	1717	44573	169	1717	44306
Sum	67750	298652	1605359	55788	244634	794725	55128	244634	778852

Tabel 4.1: Results for the test problems.

Name	ITE	TOPT	TSET	LITE	LTOPT	LTSET	RATIO
80BAU3B	43	33.57	1.30	43	33.69	1.46	1.1
aa3	26	119.47	0.52	23	83.97	3.93	7.6
CRE-B	47	296.02	3.26	47	289.51	4.46	1.4
KEN-13	24	238.87	3.36	24	187.39	15.06	4.5
NUG12	14	2482.55	0.41	26	2863.71	25.23	61.5
OSA-30	21	172.95	8.94	21	174.86	10.28	1.1
PDS-10	47	2788.63	3.24	49	2867.46	5.76	1.8
PILOT87	48	425.94	1.86	48	425.84	2.19	1.2
WOOD1P	14	10.53	0.53	14	11.12	1.62	3.1
Sum	284	6568.53	23.42	295	6937.55	69.99	3.0

Tabel 4.2: Results for the test problems continued.

4.4 Computational results

In this section we will present our numerical results. We have implemented the above procedure in C and added it to our presolve procedure described in Andersen and Andersen [7]. All the computations are performed on a HP 715/75 workstation with 64 MB ram. The code is compiled with the standard HP C compiler using +O3 optimizing option. Our code is an implementation of the primal-dual barrier algorithm. It includes a cross-over procedure, see Andersen and Ye [11]. All timing results are obtained with the C procedure "clock" which measures the time in number of CPU SEC.

As test problems we use some of the largest NETLIB problems [56] (80BAU3B, PILOT87 and WOOD1P), some of the Kennington problems (CRE-B, KEN-13, OSA-30 and PDS-10), an aircrew scheduling problem (aa3) and NUG12 (see Resende et al. [123]).

The problems are shown in Table 4.1. In the table the columns ROWS, COLS and NNZA show the number of rows, columns and nonzeros in A before presolving. RROWS, RCOLS and RNNZA show the corresponding numbers but after presolve. Finally LROWS, LCOLS and LNNZA show the corresponding numbers but after the dependent rows are removed. It should be noted that the last stage of our presolve procedure is to remove the dependent rows. Even though we remove some dependent rows we do not try to presolve the LP problem further. (We have not found this worthwhile.) Therefore the difference between RROWS and LROWS immediately shows the number of linear dependent rows in the problem. We see that the problems aa3(73), CRE-B(8), KEN-13(169), NUG12(398), PDS-10(11) and WOOD1P(1) all contain dependent rows. The number in the parenthesis shows the number of dependent rows. If we compare the columns RNNZA and LNNZA we see that usually very few nonzeros are removed by removing the dependent rows. However, for the problem aa3 and NUG12 the reduction in number of nonzeros is greater than 10%.

Checking for dependent rows is of course only worthwhile if they can be found computationally fast or we gain some computational advantages during the optimization phase. In Table 4.2 we have shown the relevant timing results for the problems.

In Table 4.2 the columns ITE, TOPT and TSET shows the number of interior-point iterations, the total solution time and the setup time when the dependent rows are not detected. LITE, LTOPT and LTSET shows the same numbers when they are detected. The last column RATIO shows the ratio LTSET/TSET. All the problems were solved using default options. They were all solved to full accuracy except NUG12. When the dependent rows were not removed from NUG12 the optimization phase terminated after 14 iterations due to numerical problems. Approximately 4 figures were correct in the objective value.

It can be seen that the setup time increases significantly for the problems aa3, KEN-13 and NUG12 when dependent rows are detected. However, in most cases the total solution time is unchanged by the dependent rows check. Although for the problem aa3 the solution time is reduced from 120 to 84 a decrease of more than 20%. Moreover the computational cost per iteration for NUG12 is reduced by 40%.

The number of iterations is virtually unchanged by removing the dependent rows. It only affects the problem aa3, NUG12 and PDS-10. It seems that our Cholesky procedure handles dependent rows well. Although the problem NUG12 demonstrates that a large number of dependent rows may create serious numerical problems.

In summary our computational results shows that finding all the linearly dependent rows may increase the setup time dramatically (e.g. NUG12). On the other hand removing the dependent rows sometimes reduces the computational cost significantly (e.g. aa3 and NUG12) and improves the numerical stability (e.g. NUG12). Moreover the total solution time is only increased slightly for those problems that do not contain any dependent rows.

4.5 Conclusion

We have shown that it is practical to check whether large-scale LP problems contain any dependent rows. Even if the LP problem does not contain any dependent rows the dependency check only increases the total solution time slightly. In other cases the detection of the linear dependent rows can decrease the solution time significantly.

In general it may not be worthwhile to include the linear dependency check in a presolve procedure. However, it is an important tool to improve numerical stability for problems containing many linear dependent rows.

Kapitel 5

Combining interior-point and pivoting algorithms for linear programming

We propose a new approach to combine linear programming (LP) interior-point and simplex pivoting algorithms. In any iteration of an interior-point algorithm we construct a related LP problem, which approximates the original problem, with a known (strictly) complementary primal-dual solution pair. Thus, we can apply Megiddo's [104] pivoting procedure to compute an optimal basis for the approximate problem in strongly polynomial time. We show that, if the approximate problem is constructed from an interior-point iterate sufficiently close to the optimal face, then any optimal basis of the approximate problem is an optimal basis for the original problem. If the LP data are rational, the total number of interior-point iterations to create such a sufficient approximate problem is bounded by a polynomial in the data size. We develop a modification of Megiddo's procedure and discuss several implementation issues in solving the approximate problem. We also report encouraging computational results for this combined approach.

This paper is co-authored with Yinyu Ye [1].

5.1 Introduction

Since Karmarkar published his paper [82] in 1984, there has been a rapid development in interior-point methods for linear programming (LP). The computational issues of this development are summarized by several researchers in a recent volume of *ORSA Journal Computing.* According to Lustig, Marsten and Shanno [97], few open questions are left open in practice. They are: a) detect infeasibility and unboundedness, b) warm-start, and c) generate an optimal basis. In this paper we will address the last issue.

Contrary to pivoting algorithms, interior-point algorithms will in general return an approximate solution in the interior of the optimal face. Except for non-degenerate cases, this solution will not be basic. In practice users sometimes want basic solutions for a couple of reasons. First of all, a basic solution has generally less nonzero components than a solution in the interior of the optimal face, which is desirable when LP relaxations of integer programming problems are solved. Secondly, an optimal basic solution provides easy warm-start for the simplex method when a sequence of closely related problems are solved. Of course, it should be mentioned that there exist many cases when an optimal interior-point solution is more desirable than a basic solution, see, Christiansen and Kortanek [34] and, more recently, Jansen et al. [77], Adler[3] and Greenberg, [68].

It has been sought to combine interior-point and simplex pivoting algorithms:

- Kortanek and Zhu [90] (also see Charnes and Kortanek [33]) proposed a "purification" pivoting

[1] Department of Management Sciences, The University of Iowa, Iowa City, IA 52242. The author is supported in part by NSF Grant DDM-9207347 and the University of Iowa Business School Summer Grant.

procedure from an interior-point to a corner-point having objective value no worse. The number of pivot steps to generate an optimal basic solution is finite but may not be polynomial.

- Given an optimal primal and dual solution pair, Megiddo proposed a strongly polynomial pivoting procedure to compute an optimal basis. One practical difficulty to use his procedure in interior-point algorithms is the fact that interior-point algorithms never generate an exact optimal solution in finite time.

- A cross-over scheme, to run an interior-point algorithm first and later switch to the simplex method, was successively implemented in a recent large-scale case study by Bixby et al. [22] and also by Bixby and Saltzman [21]. It uses a heuristic, based on information from interior-point algorithm iterates, to construct an initial basis for the simplex method to begin with. No theoretical result is established for this approach.

The goal of our paper is to develop an approach to combine interior-point and pivoting algorithms that possesses **both** theoretical guarantee and practical efficiency. We accomplish this goal using two ideas. The first idea is to construct an artificial LP problem, which approximates the original LP problem, in any iteration of an interior-point algorithm. A key feature of the approximate problem is that it has a known (strictly) complementary solution pair. Moreover, we show that, if the approximate problem is constructed from an interior-point iterate sufficiently close to the optimal face, then every optimal basis of this approximate problem is an optimal basis of the original problem. If the LP data are rational, the total number of interior-point iterations to create such a sufficient approximate problem is bounded above by $O(\sqrt{n}L)$, where L is the data size and n is the number of variables in the standard primal form.

Since a complementary solution pair is known for this approximate problem, we could simply apply Megiddo's procedure [104] to compute an optimal basis of the approximate problem in n pivot steps. Our second idea is to develop a modification (which is also strongly polynomial) of Megiddo's and Bixby and Saltzman's procedures. In fact, our modification allows either cross-over to a pivoting algorithm completely or to execute both interior-point and pivoting algorithms concurrently.

This paper is organized as follows. In the next section we introduce some relevant technical results. In Section 5.3, we show how to construct an LP problem, which approximates the original LP problem, using information from interior-point iterates. Afterwards, in Section 5.4 we develop a purification pivoting procedure, based on Megiddo and Bixby and Saltzman, for obtaining an optimal basis from a known complementary solution. In Section 5.5 we discuss implementation issues and propose a criterion on when to start the purification procedure for solving the approximate problem. Finally in Section 5.6 and 5.7 we present computational results and final remarks.

5.2 Technical review

Any LP problem can be stated in the following form

$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \tag{5.1}$$

where $c, x \in R^n$, $A \in R^{m \times n}$ and $b \in R^m$. Without lost of generality, we assume that A has full row rank. We will denote (5.1) by $LP(A, b, c)$. The dual to (5.1) is

$$\begin{array}{rl} \max & b^T y \\ \text{s.t.} & A^T y + s = c \\ & s \geq 0 \end{array} \tag{5.2}$$

where $y \in R^m$ and $s \in R^n$. Let the support

$$\sigma(v) = \{j : v_j \neq 0,\ v \in R^n\}. \tag{5.3}$$

We will use the following notation

$$\bar{\sigma}(v) = \{j : v_j = 0,\ v \in R^n\}. \tag{5.4}$$

It is well-known that if there exists an optimal solution x^* and (y^*, s^*) to (5.1) and (5.2), respectively, then the duality gap

$$(x^*)^T s^* = c^T x^* - b^T y^* = 0. \tag{5.5}$$

Moreover there always exists a strictly complementary solution such that

$$x^* + s^* > 0 \quad \text{or} \quad \sigma(x^*) = \bar{\sigma}(s^*). \tag{5.6}$$

This result was proven by Goldman and Tucker [62]. Also see a recent interesting proof by Jansen et al. [78] based on interior-point methods. Furthermore for any strictly complementary solution (x^*, y^*, s^*) we have

$$\sigma(x^*) = P^* \quad \text{and} \quad \sigma(s^*) = Z^* \tag{5.7}$$

where $P^* \cup Z^* = \{1, ..., n\}$. Thus, the partition is invariant with respect to the solution. Let us call it the optimal partition. In what follows, P and Z will also be used as matrices obtained from A corresponding to their index sets. Thus, the optimal face for the primal problem is

$$\Theta_P = \{x : \ Ax = b, \ x \geq 0, x_j = 0 \text{ for } j \in Z^*\}, \tag{5.8}$$

and the one for the dual is

$$\Theta_D = \{(y, s) : \ A^T y + s = c, \ s \geq 0, s_j = 0 \text{ for } j \in P^*\}. \tag{5.9}$$

These faces have interior points, because $\tilde{\Theta}_P = \{x : Ax = b, \ x_{P^*} > 0, \ x_j = 0 \text{ for } j \in Z^*\}$ and $\tilde{\Theta}_D = \{(y, s) : \ A^T y + s = c, \ s_{Z^*} > 0, \ s_j = 0 \text{ for } j \in P^*\}$ are nonempty. Now we define

$$\xi_P = \min_{j \in P^*} \{\max \ x_j : x \in \Theta_P\} \tag{5.10}$$

and

$$\xi_D = \min_{j \in Z^*} \{\max \ s_j : (y, s) \in \Theta_D\}, \tag{5.11}$$

and let

$$\xi = \min(1, \xi_P, \xi_D). \tag{5.12}$$

We must have $\xi > 0$. It can be verified that, if $LP(A, b, c)$ has rational data, then $\xi \geq 2^{-L}$, where L is the bit size of A, b and c. Throughout this paper, we assume that $LP(A, b, c)$ has a solution.

Many polynomial (and practical) interior-point algorithms (e.g. Kojima et al. [86], Mizuno et al. [110], Monteiro and Adler [113] and Lustig et al. [96]) generate a solution sequence (or subsequence) (x^k, s^k) such that

$$(x^{k+1})^T s^{k+1} \leq (x^k)^T s^k \text{ and } \frac{\min(X^k s^k)}{(x^k)^T s^k} \geq \Omega(1/n). \tag{5.13}$$

Given an iterate (x^k, s^k) of the sequence satisfying (5.13), define

$$P^k = \{j : x_j^k \geq s_j^k\} \quad \text{and} \quad Z^k = \{1, ..., n\} \setminus P^k. \tag{5.14}$$

Based on the result of Güler and Ye [70] (also see Mehrotra and Ye[108] [151]), we have

Theorem 5.2.1 *Let $\{x^k, s^k\}$ be generated by an $O(\sqrt{n}L)$ interior-point algorithm satisfying (5.13) and starting from $x^0 = e$ and $s^0 = e$. Then,*

1.
$$0 < e^T x^k + e^T s^k \leq 2n, \ \forall \ k. \tag{5.15}$$

2.
$$x_j^k \geq \Omega(\xi/n) \text{ and } s_j^k \leq (x^k)^T s^k/\xi, \ \forall \ k \text{ and } j \in P^* \tag{5.16}$$

and

$$s_j^k \geq \Omega(\xi/n) \text{ and } x_j^k \leq (x^k) s^k/\xi, \ \forall \ k \text{ and } j \in Z^*. \tag{5.17}$$

3. When $(x^k)^T s^k \leq O(\xi^2/n)$*, then*

$$s_j^k < O(\xi/n) \ \forall\ j \in P^* \ \textit{and}\ x_j^k < O(\xi/n) \ \forall\ j \in Z^*, \tag{5.18}$$

so that

$$P^k = P^* \ \textit{and}\ Z^k = Z^*. \tag{5.19}$$

Proof: Since

$$(x^k - x^0)^T (s^k - s^0) = (x^k - e)^T (s^k - e) = 0,$$

we have

$$e^T x^k + e^T s^k = n + (x^k)^T s^k \leq 2n$$

which proves the first statement.

Also for any complementary solution (x^*, s^*), we have

$$(x^k - x^*)^T (s^k - s^*) = 0.$$

Thus,

$$\sum_{j \in Z^*} x_j^k s_j^* + \sum_{j \in P^*} s_j^k x_j^* = (x^k)^T s^k$$

and

$$\sum_{j \in Z^*} \frac{s_j^*}{s_j^k} \frac{x_j^k s_j^k}{(x^k)^T s^k} + \sum_{j \in P^*} \frac{x_j^*}{x_j^k} \frac{s_j^k x_j^k}{(x^k)^T s^k} = 1.$$

Therefore, for any $j \in P^*$ and any complementary solution (x^*, s^*)

$$s_j^k x_j^* \leq (x^k)^T s^k \quad \text{and} \quad \frac{x_j^*}{x_j^k} \frac{s_j^k x_j^k}{(x^k)^T s^k} \leq 1.$$

Given an $j \in P^*$, we let $x^* = \arg\max\ \{x_j : x \in \Theta_P\}$. Then, using the above fact, the definition of ξ, and (5.13), we obtain

$$s_j^k \xi \leq s_j^k x_j^* < (x^k)^T s^k \quad \text{and} \quad \frac{\xi}{x_j^k} \leq \frac{x_j^*}{x_j^k} < \frac{(x^k)^T s^k}{x_j^k s_j^k} \leq O(n),$$

which proves the second statement for $j \in P^*$. Similarly, we can prove the statement for other $j \in P^*$ and for all $j \in Z^*$.

The third statement of the theorem directly follows from the second statement when $(x^k)^T s^k \leq O(\xi^2/n)$.

□

The above theorem indicates that the partition (P^k, Z^k) will be optimal after $O(\sqrt{n} \log(n/\xi))$ iterations. If $LP(A, b, c)$ has rational data, this will happen in $O(\sqrt{n}L)$ iterations. It also states that as $(x^k)^T s^k \to 0$, zero variables uniformly go to zero and positive variables are uniformly bounded away from zero. Hence, interior-point methods will converge to a solution in the interior of the optimal primal and dual faces.

Now let (B, N) denote a partition of variables into basic and non-basic variables. We actually let (B, N) denote both index sets and the corresponding matrices obtained from A. Then, B is an optimal basis if and only if B is non-singular,

$$x_B = B^{-1} b \geq 0, \tag{5.20}$$

and

$$y = B^{-T} c_B, \ s_N = c_N - N^T y \geq 0. \tag{5.21}$$

If (5.1) is consistent and bounded, then an optimal basis always exists. This is the foundation of the simplex pivoting algorithm. A basic solution is said to be primal degenerate if at least one component in x_B is zero. Similarly the dual solution is degenerate if at least one of the components in s_N is zero.

5.3 An approximate LP problem

We make an assumption that we have an $O(\sqrt{n}L)$ interior-point algorithm which generates a sequence of iterates $(x^k > 0, y^k, s^k > 0)$ for $LP(A, b, c)$. (Actually we do not need to assume that these iterates are feasible, see such an algorithm in Ye et al. [156].) In addition, we assume that the interior-point algorithm generates (P^k, Z^k) by (5.14) which is an estimate for the optimal partition (P^*, Z^*).

Using these assumptions, we define a perturbed LP problem

$$\begin{array}{rl} \min & (c^k)^T x \\ \text{s.t.} & Ax = b^k \\ & x \geq 0 \end{array} \tag{5.22}$$

where

$$b^k = P^k x^k_{P^k}, \; c^k_{P^k} = (P^k)^T y^k, \; c^k_{Z^k} = (Z^k)^T y^k + s^k_{Z^k}. \tag{5.23}$$

We can now state the following Proposition

Proposition 5.3.1 *The solutions $x = (x^k_{P^k}, 0)$ and $(y, s) = (y^k, (0, s^k_{Z^k}))$ are an optimal primal-dual pair to (5.22). In fact, they are strictly complementary.*

Proof: Feasibility follows from construction, optimality from $x^T s = 0$, and strict complementarity from $x^k_{P^k} > 0$ and $s^k_{Z^k} > 0$.

□

Due to Theorem 5.2.1, we know that $LP(A, b^k, c^k)$ will eventually share the same optimal partition with $LP(A, b, c)$. We would like to emphasize that this fact does not hold in general. If b^k and c^k are not constructed from (5.14), (5.23) and Proposition 5.3.1, then $b^k \to b$ and $c^k \to c$ does not necessarily guarantee (P^k, Z^k) converges to (P^*, Z^*) (see Theorem 3 and an example of Mizuno et al. [111]). In other words, the optimal partition for the approximate LP and the one for the original LP may be different, even if (b^k, c^k) is arbitrarily close to (b, c).

For simplicity, we assume that the iterate (x^k, y^k, s^k) is feasible. Let $\eta = \max(1, \|a_j\|)$ where a_j is the j'th column in A. (Again, if $LP(A, b, c)$ has rational data, $\eta \leq 2^L$.) Then, after $(P^k, Z^k) = (P^*, Z^*)$, we have from Theorem 5.2.1 that

$$\|b^k - b\| = \|Z^k x^k_{Z^k}\| \leq n\eta (x^k)^T s^k / \xi \tag{5.24}$$

and

$$\|c^k - c\| = \|s^k_{P^k}\| \leq \sqrt{n}(x^k) s^k / \xi \tag{5.25}$$

which shows that $b^k \to b$ and $c^k \to c$, as $(x^k)^T s^k \to 0$. Moreover, if $(x^k)^T s^k \leq \min\{O(\xi^2/n), 2^{-t}\xi/(n\eta)\}$, we have

$$(P^k, Z^k) = (P^*, Z^*), \;\; \|b^k - b\| \leq 2^{-t} \;\; \text{and} \;\; \|c^k - c\| \leq 2^{-t}. \tag{5.26}$$

Now we state the following theorem.

Theorem 5.3.1 *Let B be an optimal basis for $LP(A, b^k, c^k)$ constructed by (5.22) and (5.23). Then, there is $0 < \bar{t} < \infty$ such that B must be also an optimal basis for the original $LP(A, b, c)$ when $(x^k)^T s^k \leq 2^{-\bar{t}}$. Furthermore, if $LP(A, b, c)$ has rational data, then $\bar{t} \leq O(L)$.*

Proof: Let B be any basis of A with $B\bar{x}_B = b$ and $B^T\bar{y} = c_B$. Among those nonzero components in $\bar{x}_B$ and $\bar{s} = c - A^T\bar{y}$, let the value of the smallest absolute-value component be τ_B, which is positive. Then, we define $\tau = \min(1, \min_B(\tau_B))$. τ must be positive since the total number of possible bases in A is finite. Also let $\zeta = \max(1, \max_B(\|B^{-1}\|))$. Again, if $LP(A, b, c)$ has rational data, $\tau \leq 2^{-L}$ and $\zeta \leq 2^L$.

We first prove the following lemma

Lemma 5.3.1 *Let B be an optimal basis for $LP(A, b^k, c^k)$, where $\|b^k - b\| < \tau/\zeta$ and $\|c^k - c\| < \tau/(\eta\zeta)$. Then, B is also an optimal basis for $LP(A, b, c)$, i.e., both solutions $\bar{x}_B$ of $B\bar{x}_B = b$ and $\bar{s} = c - A^T\bar{y}$, where $B^T\bar{y} = c_B$, are nonnegative.*

Proof: Let the corresponding optimal primal basic solution be $\bar{x}^k_B \geq 0$ and the dual basic solution be $(\bar{y}^k, \bar{s}^k = c - A^T\bar{y}^k)$ for $LP(A, b^k, c^k)$, i.e., $B\bar{x}^k_B = b^k$ and $B^T\bar{y}^k = c^k_B$. Then, if $x_B \not\geq 0$, each of its negative components must be less than $-\tau$. Let the i'th component be negative, i.e., $\bar{x}_i \leq -\tau$. On the other hand, we have

$$\bar{x}^k_B - x_B = B^{-1}(b^k - b)$$

or

$$\left\|\bar{x}^k_B - \bar{x}_B\right\| = \left\|B^{-1}(b^k - b)\right\| \leq \left\|B^{-1}\right\| \left\|b^k - b\right\|.$$

Thus,

$$|\bar{x}^k_i - \bar{x}_i| \leq \left\|B^{-1}\right\| \left\|b^k - b\right\| < \tau$$

or

$$\bar{x}^k_i < \tau + \bar{x}_i \leq \tau - \tau = 0.$$

This contradicts that $\bar{x}^k_B \geq 0$.

Similarly, if $\bar{s}_j \leq -\tau$, then

$$|\bar{s}^k_j - \bar{s}_j| = |a^T_j(\bar{y}^k - y)| \leq |a^T_j B^{-T}(c^k_B - c_B)| \leq \left\|a^T_j\right\| \left\|B^{-1}\right\| \left\|c^k - c\right\| < \tau.$$

This implies that

$$\bar{s}^k_j < \tau + \bar{s}_j \leq \tau - \tau = 0,$$

which is also a contradiction.

□

Now, we just need to choose $\bar{t}$ big enough but finite such that $(x^k)^T s^k \leq 2^{-\bar{t}}$ to establish

$$(P^k, Z^k) = (P^*, Z^*)$$

and

$$\left\|b^k - b\right\| < \tau/\zeta \text{ and } \left\|c^k - c\right\| < \tau/(\eta\zeta).$$

From relations (5.24) and (5.25), this can be accomplished if

$$2^{-\bar{t}} \leq \min\{O\left(\frac{\xi^2}{n}\right), \frac{\xi\tau}{n\eta\zeta}\}.$$

If $LP(A, b, c)$ has rational data, one can verify that $\bar{t} \leq O(L)$. Thus, the theorem is proved.

□

From the above theorem, we have

Corollary 5.3.1 *Let $\{x^k, y^k, s^k\}$ be generated by an $O(\sqrt{n}L)$ interior-point algorithm satisfying (5.13) and starting from $x^0 = e$ and $s^0 = e$. Then, after at most $O(\sqrt{n}L)$ iterations, an optimal basis of $LP(A, b^k, c^k)$ constructed by (5.22) and (5.23) is also optimal for $LP(A, b, c)$.*

5.4 Purification pivoting procedures

In this section, we discuss two purification pivoting procedures to compute an optimal basis for (5.22) from a complementary solution pair. For convenience we will assume that a full set of artificial variables have been added to the problem (5.22), and let $V = \{n+1, \ldots, n+m\}$ represent the set of artificial variables. Thus, A has been augmented with an identity matrix, i.e., $A := [A, I]$. Using results described in previous sections, we now assume the following for the problem $LP(A, b^k, c^k)$:

(A1) The optimal partition (P, Z) is known and $V \subseteq Z$.

(A2) A primal optimal solution $\tilde{x}$ is known such that $A\tilde{x} = b^k$, $\tilde{x}_Z = 0$ and $\tilde{x}_P > 0$.

(A3) A dual optimal solution $(\tilde{y}, \tilde{s})$ is known such that $A^T\tilde{y} + \tilde{s} = c^k$, $\tilde{s}_{Z \setminus V} > 0$ and $\tilde{s}_P = 0$.

Here, the dual slack variable $\tilde{s}_V$, corresponding to the fixed primal artificial variables, are free of sign and can assume any value. Note that the pivoting procedures discussed below work even if $\tilde{x}_P > 0$ and $\tilde{s}_{Z \setminus V} > 0$ are relaxed to $\tilde{x}_P \geq 0$ and $\tilde{s}_{Z \setminus V} \geq 0$ in (A2) and (A3).

5.4.1 A complementarity pivoting procedure

For simplicity, we replace (b^k, c^k) by (b, c) in the rest of this section. The first pivoting procedure generates iterates which satisfy the complementary condition, but not the non-negativity constraint. Therefore we call the procedure a complementarity pivoting procedure, which is related to the method of Kortanek and Zhu [90].

The idea of the primal phase of the procedure can be outlined as follows. Let (B, N) be a basic and non-basic partition such that

$$x_B = B^{-1}b, \quad x_N = 0, \quad \text{and} \quad x_Z = 0.$$

Such a basis is easy to find, see Algorithm 5.4.1 below. In general $x_B \not\geq 0$; otherwise B is primal optimal. Therefore, there must be at least one negative component in x_B. Now define a "purified" solution

$$\hat{x} = \alpha x + (1 - \alpha)\tilde{x} \tag{5.27}$$

where

$$\alpha = 1/(1 - \rho) \quad \text{and} \quad \rho = \min_{B_i \in B} \{x_{B_i}/\tilde{x}_{B_i} : \ \tilde{x}_{B_i} > 0\}.$$

Clearly $\alpha \in (0, 1)$ and by construction $A\hat{x} = b$, $\hat{x} \geq 0$ and $\hat{x}_Z = 0$. Hence $\hat{x} \in \Theta_P$. Moreover, $\hat{x}$ has at least one more zero component than $\tilde{x}$ does. Let

$$I = \{B_i : x_{B_i}/\tilde{x}_{B_i} = \rho, \ \tilde{x}_{B_i} > 0\}. \tag{5.28}$$

It follows that at least one of the basic variables $B_i \in I$ can be replaced with one variable j where $j \in P \cap N$ and $\tilde{x}_j > 0$. If we let $\hat{x}$ replace $\tilde{x}$ then this procedure can be repeated. After at most $|P|$ pivots we must have $A\tilde{x} = Ax = b$ and $\tilde{x}_N = x_N = 0$. Thus, $\tilde{x}_B = x_B \geq 0$ is primal feasible and optimal.

We can now formally state the following procedure.

Algorithm 5.4.1

1. *Choose a basis* B, $k = 0$.
2. *while* ($\exists B_i \in Z : x^k_{B_i} \neq 0$)
3. *Exchange the ith basic variable with a variable* $j \in P \cap N$
4. $k := k + 1$
5. *end while*
6. *while* ($\exists B_i \in P : x^k_{B_i} < 0$)
7. $\rho = min\{x^k_{B_i}/\tilde{x}^k_{B_i} : \tilde{x}^k_{B_i} > 0\}$
8. $I = \{i : x^k_{B_i}/\tilde{x}^k_{B_i} = \rho, \tilde{x}^k_{B_i} > 0\}$
9. *Exchange one basic variables* $B_i \in I$ *with a variable* $j \in P \cap N, \quad \tilde{x}^k_j > 0$
10. $\alpha = 1/(1 - \rho), \quad \tilde{x}^{k+1} = \alpha x^k + (1 - \alpha)\tilde{x}^k$
12. $k := k + 1$
13. *end while*

In the above procedure we have used the convention that $x^k_B = B^{-1}b$ where B is the current basis. Steps 2 to 5 construct a basic solution such that $x_Z = 0$. Steps 6 to 13 purify the interior-point solution $\tilde{x}$ to an optimal primal basic solution using the idea outlined above.

Starting from the primal optimal basis B obtained in the primal phase, we can develop a similar dual purification procedure to purify the interior-point dual solution $(\tilde{y}, \tilde{s})$ to an optimal dual basic solution. It can be verified that these two phases together generate an optimal primal-dual basis using less than n pivots.

5.4.2 A superbasic pivoting procedure

Megiddo [104] proposed a procedure to generate an optimal basis from an optimal solution pair. His procedure was slightly modified by Bixby and Saltzman [23]. Their modification generates a basic solution that maintains non-negativity in all pivots and move iterates towards complementarity.

This procedure can be described as follows. Let (B, N) be any basic and non-basic partition of variables. Note

$$\tilde{x}_B = B^{-1}(b - N\tilde{x}_N). \tag{5.29}$$

Usually $(\tilde{x}_B, \tilde{x}_N)$ is called a primal superbasic solution, since some of its non-basic variables are not zero (e.g., Nazareth [116, p. 31]).

The primal phase of Megiddo's procedure either moves one non-basic variable in $\tilde{x}_N$ to zero, if possible, or pivots them into the basis one at a time. Each move or pivot reduces the number of non-zero and non-basic variables by one, and therefore the number of total moves and pivots is bounded by $|P|$.

This procedure is formally stated as follows:

Algorithm 5.4.2

1. *Choose a basis* B *and let* $x^0 = \tilde{x},\ k = 0$.
2. *while*$(\exists\ j \in N : x_j^k \neq 0)$
3.   *Use the primal ratio test to move variable* x_j^k *to zero if possible or pivot it into the basis*
4.   *Update* $(B, N),\ x^k,\ k := k + 1$
5. *end while*

The resulting B will be a primal optimal basis. One possible choice of B in Step 1 is V, i.e., the set of artificial variables.

Similarly, one can purify the superbasic dual solution $(\tilde{y}, \tilde{s})$ to a basic solution. The dual phase can be stated as follows:

Algorithm 5.4.3

1. *Choose a basis* B *and let* $y^0 = \tilde{y},\ s^0 = c - A^T\tilde{y},\ k = 0$
2. *while*$(\exists\ j \in B : s_j^k \neq 0)$
3.   *Use the dual ratio test to move variable* s_j^k *to zero if possible or pivot it out of the basis*
4.   *Update* $(B, N),\ y^k,\ s^k,\ k := k + 1$
5. *end while*

The resulting B will be a dual optimal basis. If the initial basis of the dual phase is primal optimal, then it will remain primal optimal, because all pivots in Algorithm 5.4.3 are primal degenerate. Therefore, the final basis will be both primal and dual optimal. Furthermore the number of pivots in the dual phase is bounded by $|Z|$. In summary the superbasic pivoting procedure generates an optimal basis in less than n pivots from a complementary solution.

5.5 Implementation issues

We now discuss several implementation issues related to the purification pivoting procedures described in the preceding section. One issue is how to estimate the optimal partition and when to start the purification procedure. The theory of Section 5.3 indicates that, if the interior-point iterate is close enough to the optimal face, then we have an accurate estimate to the optimal partition and we should likely be successful in finding an optimal basis for the original LP problem.

We propose to use the primal-dual Newton direction to determine whether or not the iterate is close to the optimal face. Mehrotra [107] and Ye et al. [155] have shown that, for any LP problem,

$$P^k = \{j : |\Delta x_j^k| / x_j^k \leq |\Delta s_j^k| / s_j^k\} \tag{5.30}$$

converge towards the optimal partition P^* where $(\Delta x, \Delta y, \Delta s)$ is the primal-dual Newton direction. This technique is closely related to Tapia's indicator, see [47]. Note that this definition of P^k is scaling invariant.

In addition, results in [107] and [155] show that, if the duality gap $(x^k)^T s^k$ is sufficiently small or the iterate is sufficiently close to the optimal face, the iterates based on using the primal-dual Newton direction will reduce the duality gap quadratically. Conversely, when fast or superlinear convergence is about to occur, we anticipate that the iterate is close to the optimal face and the partition P^k will likely equal P^*. Thus, we start purification process when the duality gap is reasonably small and the fast convergence is observed.

More precisely, we use the following criteria. Let $\alpha^{\max}$ be the maximum possible step length to the boundary along the primal-dual Newton direction. If the fast convergence occurs, this step length will approach one. We therefore try to start purification if a solution is close to being primal and dual feasible, and

$$\alpha^{\max} \geq (1-\beta^1), \quad \frac{c^Tx^k - b^Ty^k}{1+|b^Ty^k|} \leq \beta^2 \tag{5.31}$$

or

$$\frac{(c^Tx^k - b^Ty^k)}{1+|b^Ty^k|} \leq \beta^3, \tag{5.32}$$

where β^1, β^2 and β^3 are suitably chosen constants. (The second stopping criterion will be used for problems where fast convergence is hardly obtained.)

The optimal basis of $LP(A, b^k, c^k)$ may not be optimal for the original LP problem. One option is to continue one or more interior-point iterations and to obtain a more accurate approximate problem. In this case, we should use the basis resulted from the latest purification try as the initial basis for the new try. We call this option a retry. Another option is to use this basis as the starting basis for the simplex algorithm, such as the cross-over procedure proposed by Bixby and Saltzman [23]. Moreover, this basis could be used as a preconditioner for an iterative method for solving the least squares problem in interior-point algorithms. In fact, how to choose such a precondition basis and how to update it are active research topics (Gill et al. [60]).

The complementarity and superbasic pivoting procedures are fairly straight forward to implement. However, we would like to the emphasize few issues. Step 1 in both the complementarity and the superbasic pivoting procedures is to choose an initial basis. Obviously, it is desirable that the initial basis is an accurate estimate to an optimal basis. Thus, we like to choose initial basic variables from P^k. Furthermore, we also like to make the initial basis numerically stable. Hence, our initial basic variables are chosen from P^k and V (the set of artificial variables), which is similar to that described in [23]. In our implementation, we have not scaled LP problems as it has been done in [23]. Scaling might improve the efficiency and stability of purification procedures further.

In both the primal and dual phases, we choose the pivot using P.M.J. Harris's idea to enlarge the pool of possible pivots, see Harris [72] and Nazareth [116, pp. 142-143]. For difficult problems we found that this strategy is important. During the dual phase we need to compute the reduced cost, where we have used a trick described by Bixby, see [21, p. 17]. We use LU factorization to represent the basis inverse. Our factorization is a modified version of Suhl and Suhl ([131]), and the LU update is a variant of Bartels-Golub ([116, p. 118]).

The interior-point code we use is an implementation of the primal-dual barrier method. The method contains Mehrotra's predictor-corrector technique [106] and Kojima et al.'s global convergence safeguard, see [84] and [96]. Define

$$pval^k = \frac{\min(x_j^k s_j^k)}{(x^k)^T s^k / n} \tag{5.33}$$

which is a measure of closeness to the central path. Lustig et al. keep this value higher than

$$\theta_C = 1.0e-8. \tag{5.34}$$

Because we need high accuracy to determine $LP(A, b^k, c^k)$, we use instead $\theta_C = 1.0e-6$. This value may restrict the primal and dual step-length too much, if the choice of step-length is chosen as in [96, pp. 126-127]. To overcome this restriction, we calculate the primal and dual step-length as follows. Let α_P^{max} and α_D^{max} be the maximum possible step-lengths along the primal and dual directions $(\Delta x, \Delta s)$, respectively. Then, we choose maximal γ^{max} such that

$$\frac{\min(x_j^k + \gamma^{max}\alpha_P^{max}\Delta x_j)(s_j^k + \gamma^{max}\alpha_D^{max}\Delta s_j)}{(x^k + \gamma^{max}\alpha_P^{max}\Delta x)^T(s^k + \gamma^{max}\alpha_D^{max}\Delta s)/n} \geq \theta_C, \ \forall\, j, \tag{5.35}$$

and then select the primal and dual step-length given by

$$(\alpha_P, \alpha_D) = \min(\gamma^{\max}, 0.99995)(\alpha_P^{\max}, \alpha_D^{\max}). \tag{5.36}$$

Using this method we find that a relatively large θ_C rarely hurts the speed of convergence. Note that finding γ^{max} reduces to solving n quadratic inequalities in one variable, because α_P^{max} and α_D^{max} are fixed scalars.

The most expensive operation in interior-point algorithms is the computation of a least-squares problem in each iteration. We solve the problem using a supernodal (push) Cholesky decomposition and an ordering that is obtained with the multiple minimum-degree heuristic, see Andersen et al. [8] for details. Dense columns in the constraint matrix are handled by a technique discussed in Andersen [15], and all test problems are presolved by a presolver described in [6, 7].

5.6 Computational results

We now present computational results for our combined approach. Our source code is written in ANSI C and is compiled on a HP 715/75 workstation using the HP C compiler and +O3 optimization option. The workstation has 64 MB RAM. All running times are obtained with the C procedure "clock" which measures CPU time in seconds (CPU sec).

We set parameters

$$\beta^1 = 10^{-1}, \quad \beta^2 = 10^{-8} \text{ and } \beta^3 = 10^{-12}.$$

Our first set of test problems is the set of NETLIB problems ([56]). We have tried both the complementarity and superbasic pivoting procedures, and found that the latter is much faster than the former in general. Therefore, we will report results based on using the superbasic pivoting procedure. In this test we try purification once using the criteria outlined in the preceding section.

Our results are shown in Tables 5.1 and 5.2. The column RGAP shows the relative duality gap of the interior-point iterate and ITE shows the number of interior-point iterations, right before the purification procedure starts. PBI and DBI represent the maximal primal and dual bound infeasibility errors, respectively. For example, PBI is calculated as:

$$x_B = B^{-1}b$$

and

$$\text{PBI } = \max_j \{0, -x_j\}$$

where B, generated by the purification procedure, is an optimal basis to the approximate problem, and b is the right hand side of the original problem. (DBI is defined in the same manner for the dual problem.) Therefore, PBI and DBI show how good the basis is to the original problem. Clearly, if PBI and DBI are zero, then B is an exact optimal basis for the original problem. TOPT is the total CPU time spent in interior-point iterations and purification pivots. TCRO is the total CPU time spent in the purification procedure. Thus, TCRO is included in TOPT. Finally, RATIO shows the ratio TCRO/TOPT.

From Tables 4.1 and 4.2 we see that an optimal basis to the original problem, with very low PBI and DBI, is obtained for each of the test problems. PBI and DBI are usually below 10^{-10}, which is very good. Other accuracy measures are even better. Only in few cases, namely DFL001, NESM, ETAMACRO, PILOT.WE, SCSD6, we achieve an accuracy less than 10^{-8}.

In addition to accuracy, the time spent in the purification procedure is important. Again from Tables 4.1 and 4.2 we see that $TCRO$ is less than 10 % of the total solution time $TOPT$. It is more than 20% in very few cases, where these problems are either very primal or dual degenerate, or their bases are dense and expensive to factor.

NETLIB problems are relative small and may not really represent today's large-scale LP problems. Our second set of test problems is the so-called Kennington problems, see Carolan et al. [29]. (We have excluded three largest problems, KEN-18, OSA-60 and PDS-20, in our test because they require memory beyond the capacity of our workstation.) Table 5.3 shows some problem related statistics.

The columns ROWS and COLS show the numbers of rows and columns before presolving. RROWS and RCOLS show the numbers of rows and columns after presolving. $|P|$ and $|Z|$ show the numbers of variables in P and Z, respectively, in the optimal partition. $|P \cap B|$ and $|Z \cap B|$ show how many variables

	Name	RGAP	ITE	PBI	DBI	TOPT	TCRO	RATIO
1	25FV47	6.2E-09	25	0.0E+00	3.1E-13	8.27	0.45	0.05
2	80BAU3B	8.5E-14	43	8.3E-17	2.8E-14	33.54	0.75	0.02
3	ADLITTLE	9.5E-10	12	0.0E+00	1.5E-12	0.13	0.02	0.15
4	AFIRO	9.3E-11	9	0.0E+00	5.6E-17	0.05	0.01	0.20
5	AGG	1.2E-10	21	0.0E+00	6.0E-12	0.61	0.03	0.05
6	AGG2	4.9E-11	18	0.0E+00	1.2E-12	1.31	0.06	0.05
7	AGG3	3.2E-09	17	0.0E+00	1.2E-12	1.27	0.04	0.03
8	BANDM	3.0E-10	17	0.0E+00	2.4E-13	0.79	0.07	0.09
9	BEACONFD	1.5E-12	9	0.0E+00	1.3E-15	0.28	0.02	0.07
10	BLEND	5.9E-11	11	0.0E+00	1.3E-15	0.15	0.02	0.13
11	BNL1	1.2E-11	27	1.8E-14	2.7E-13	3.78	0.43	0.11
12	BNL2	5.0E-12	37	2.1E-14	9.7E-15	30.46	1.75	0.06
13	BOEING1	5.1E-10	25	0.0E+00	7.1E-15	1.54	0.08	0.05
14	BOEING2	2.1E-10	17	0.0E+00	3.6E-15	0.37	0.02	0.05
15	BORE3D	1.6E-10	17	0.0E+00	7.1E-15	0.32	0.02	0.06
16	BRANDY	1.0E-11	17	0.0E+00	5.7E-14	0.59	0.06	0.10
17	CAPRI	-3.2E-13	19	0.0E+00	1.4E-13	0.81	0.05	0.06
18	CYCLE	1.8E-11	36	1.1E-13	2.7E-16	17.64	1.98	0.11
19	CZPROB	1.7E-11	32	0.0E+00	5.7E-14	4.84	0.15	0.03
20	D2Q06C	-7.2E-13	30	4.7E-14	2.0E-09	53.63	2.48	0.05
21	D6CUBE	4.5E-11	23	1.6E-13	4.8E-14	24.94	2.82	0.11
22	DEGEN2	7.2E-09	12	2.2E-16	5.1E-15	2.68	0.68	0.25
23	DEGEN3	2.6E-11	18	1.1E-15	7.8E-15	36.21	7.19	0.20
24	DFL001	4.5E-10	44	6.5E-11	1.1E-07	3142.71	37.54	0.01
25	E226	1.3E-11	22	0.0E+00	2.7E-15	0.81	0.05	0.06
26	ETAMACRO	1.6E-15	36	0.0E+00	3.0E-08	2.71	0.07	0.03
27	FFFFF800	5.9E-13	31	0.0E+00	1.5E-11	3.14	0.08	0.03
28	FINNIS	-1.8E-13	23	0.0E+00	1.1E-13	1.31	0.08	0.06
29	FIT1D	6.8E-09	22	0.0E+00	7.1E-15	2.44	0.05	0.02
30	FIT1P	1.6E-10	18	0.0E+00	7.3E-12	3.75	0.12	0.03
31	FIT2D	2.1E-13	22	0.0E+00	7.5E-14	41.95	0.49	0.01
32	FIT2P	5.2E-13	19	0.0E+00	1.8E-11	32.89	1.18	0.04
33	FORPLAN	2.6E-10	26	0.0E+00	2.6E-14	1.08	0.04	0.04
34	GANGES	3.7E-12	16	0.0E+00	8.9E-14	4.36	0.42	0.10
35	GFRD-PNC	3.2E-09	18	0.0E+00	1.5E-11	1.27	0.10	0.08
36	GREENBEA	1.9E-10	42	9.0E-14	3.1E-10	33.66	1.49	0.04
37	GREENBEB	3.2E-13	41	4.2E-12	3.6E-13	28.73	1.71	0.06
38	GROW15	1.9E-10	15	0.0E+00	1.9E-09	2.27	0.54	0.24
39	GROW22	1.4E-10	16	0.0E+00	9.1E-09	4.47	1.70	0.38
40	GROW7	2.8E-09	14	0.0E+00	3.7E-14	0.85	0.17	0.20
41	ISRAEL	1.2E-13	25	0.0E+00	2.0E-13	1.24	0.04	0.03
42	KB2	6.4E-10	17	0.0E+00	7.1E-15	0.12	0.01	0.08
43	LOTFI	1.7E-11	20	2.1E-11	1.5E-16	0.37	0.02	0.05
44	MAROS-R7	2.0E-11	13	0.0E+00	2.0E-10	112.91	3.21	0.03
45	MAROS	1.0E-10	26	3.5E-10	6.3E-12	4.16	0.26	0.06
46	MODSZK1	-2.3E-07	24	9.1E-13	5.7E-14	3.21	0.16	0.05
Sum			1042			3654.62	68.71	

Tabel 5.1: Results for the NETLIB problems.

	Name	RGAP	ITE	PBI	DBI	TOPT	TCRO	RATIO
45	NESM	1.6E-11	40	0.0E+00	2.0E-08	11.33	0.34	0.03
46	PEROLD	3.5E-13	37	1.3E-09	2.7E-11	7.92	0.60	0.08
47	PILOT	-7.1E-12	44	0.0E+00	1.0E-09	130.73	5.79	0.04
48	PILOT.JA	2.0E-12	38	1.4E-14	1.5E-09	19.59	2.00	0.10
49	PILOT.WE	-2.0E-13	43	0.0E+00	1.7E-07	10.06	0.46	0.05
50	PILOT4	2.3E-12	39	0.0E+00	4.4E-10	5.83	0.30	0.05
51	PILOT87	2.2E-14	50	0.0E+00	1.8E-08	493.85	22.98	0.05
52	PILOTNOV	1.9E-09	23	2.9E-12	4.4E-16	14.60	2.86	0.20
53	RECIPE	1.4E-09	10	0.0E+00	2.2E-16	0.13	0.01	0.08
54	SC105	1.4E-09	11	0.0E+00	1.1E-16	0.15	0.02	0.13
55	SC205	1.5E-11	13	0.0E+00	2.5E-16	0.30	0.04	0.13
56	SC50A	8.6E-13	10	0.0E+00	1.7E-16	0.08	0.01	0.13
57	SC50B	6.7E-09	8	0.0E+00	2.8E-17	0.07	0.01	0.14
58	SCAGR25	1.7E-10	20	0.0E+00	1.6E-12	0.99	0.06	0.06
59	SCAGR7	8.1E-12	16	0.0E+00	2.7E-13	0.20	0.01	0.05
60	SCFXM1	8.7E-13	18	0.0E+00	1.7E-14	0.93	0.05	0.05
61	SCFXM2	5.0E-09	18	0.0E+00	1.9E-14	2.05	0.13	0.06
62	SCFXM3	3.2E-10	19	0.0E+00	2.0E-14	3.44	0.22	0.06
63	SCORPION	7.8E-09	10	0.0E+00	3.4E-13	0.44	0.05	0.11
64	SCRS8	1.2E-09	20	0.0E+00	9.1E-13	1.64	0.12	0.07
65	SCSD1	7.2E-09	9	0.0E+00	2.6E-09	0.40	0.04	0.10
66	SCSD6	2.5E-10	11	1.9E-16	2.0E-08	0.90	0.09	0.10
67	SCSD8	2.0E-10	10	0.0E+00	3.5E-13	2.19	0.45	0.21
68	SCTAP1	5.7E-11	15	0.0E+00	8.5E-14	0.69	0.11	0.16
69	SCTAP2	3.9E-09	13	0.0E+00	1.4E-14	2.92	0.26	0.09
70	SCTAP3	2.4E-09	15	0.0E+00	3.6E-15	4.46	0.34	0.08
71	SEBA	1.3E-11	7	0.0E+00	0.0E+00	0.44	0.03	0.07
72	SHARE1B	2.6E-11	23	0.0E+00	1.2E-13	0.40	0.03	0.08
73	SHARE2B	4.5E-12	12	0.0E+00	5.7E-14	0.21	0.02	0.10
74	SHELL	9.3E-12	23	0.0E+00	0.0E+00	2.04	0.09	0.04
75	SHIP04L	1.5E-10	14	0.0E+00	6.8E-13	1.42	0.08	0.06
76	SHIP04S	1.9E-11	15	2.8E-14	4.5E-13	0.99	0.06	0.06
77	SHIP08L	5.5E-11	17	4.4E-16	1.3E-12	3.32	0.18	0.05
78	SHIP08S	5.8E-11	15	3.6E-15	9.1E-13	1.70	0.11	0.06
79	SHIP12L	1.8E-09	19	2.8E-15	5.5E-13	5.00	0.27	0.05
80	SHIP12S	6.8E-11	18	4.6E-15	5.5E-13	2.32	0.16	0.07
81	SIERRA	9.3E-10	18	2.3E-13	7.3E-12	4.19	0.21	0.05
82	STAIR	5.1E-12	17	0.0E+00	4.7E-15	2.45	0.29	0.12
83	STANDATA	5.8E-13	15	0.0E+00	0.0E+00	0.81	0.06	0.07
84	STANDMPS	4.6E-10	22	0.0E+00	1.8E-15	1.51	0.10	0.07
85	STOCFOR1	3.6E-10	15	1.3E-17	5.7E-13	0.19	0.01	0.05
86	STOCFOR2	1.2E-12	21	1.8E-15	2.1E-13	7.38	0.73	0.10
87	STOCHFOR	6.9E-11	33	3.2E-14	6.4E-14	128.72	32.65	0.25
88	TRUSS	2.6E-12	18	5.7E-14	7.9E-12	16.38	1.91	0.12
89	TUFF	2.8E-10	23	0.0E+00	1.7E-17	1.72	0.09	0.05
90	VTP.BASE	5.6E-11	11	0.0E+00	2.3E-13	0.15	0.01	0.07
91	WOOD1P	2.3E-09	14	1.3E-16	1.7E-13	11.39	0.50	0.04
92	WOODW	4.6E-10	23	2.3E-18	3.6E-15	11.41	0.62	0.05
Sum			953			920.03	75.56	
Total			1995			4574.65	144.27	

Tabel 5.2: Results for the NETLIB problems continued.

Name	ROWS	COLS	RROWS	RCOLS	$\|P\|$	$\|P \cap B\|$	$\|Z\|$	$\|Z \cap B\|$
CRE-A	3517	4067	2781	3938	822	701	5662	2080
CRE-B	9649	72447	5316	31818	901	751	35469	4565
CRE-C	3069	3678	2323	3249	823	719	4542	1604
CRE-D	8927	69980	4081	25188	933	743	27655	3338
KEN-07	2427	3602	1378	2603	1236	1235	1367	143
KEN-11	14695	21349	9946	16715	7182	7182	9533	2764
KEN-13	28633	42659	22356	36552	15395	15394	21157	6962
OSA-07	1119	23949	1047	23015	1049	1042	23013	5
OSA-14	2338	52460	2266	50457	2345	2261	50378	5
OSA-30	4351	100024	4279	96119	4440	4274	95958	5
PDS-02	2954	7535	2594	7172	1617	1078	5718	1516
PDS-06	9882	28655	9121	27852	5415	3736	23033	5385
PDS-10	16559	48763	15598	47729	8104	5872	40637	9726
Sum	108120	479168	83086	372407				

Tabel 5.3: The Kennington problems.

Name	RGAP	ITE	PBI	DBI	TOPT	TCRO	RATIO
CRE-A	1.1E-09	34	1.4E-12	5.8E-11	21.19	1.98	0.09
CRE-B	2.4E-13	46	4.2E-14	5.8E-11	291.11	15.84	0.05
CRE-C	4.4E-11	30	2.8E-14	1.4E-10	16.30	1.48	0.09
CRE-D	2.2E-12	46	1.1E-13	4.4E-11	248.29	10.19	0.04
KEN-07	2.0E-09	13	0.0E+00	3.6E-12	4.24	0.33	0.08
KEN-11	3.6E-09	22	0.0E+00	3.6E-12	63.95	5.86	0.09
KEN-13	5.5E-12	25	0.0E+00	3.6E-12	192.09	28.63	0.15
OSA-07	1.6E-11	17	0.0E+00	6.8E-13	32.33	0.73	0.02
OSA-14	-2.9E-14	20	0.0E+00	9.1E-13	84.62	2.36	0.03
OSA-30	-2.1E-14	22	0.0E+00	8.0E-13	183.97	8.36	0.05
PDS-02	4.2E-12	29	0.0E+00	0.0E+00	22.66	1.99	0.09
PDS-06	8.3E-13	38	0.0E+00	0.0E+00	507.41	28.24	0.06
PDS-10	4.4E-10	46	0.0E+00	0.0E+00	2953.51	79.16	0.03
Sum		388			4621.67	185.15	0.04

Tabel 5.4: Results for the Kennington problems.

from P and Z included in the final optimal basis. An important observation from Table 5.3 is that the optimal basic solution is typically highly degenerate. This can be observed from the fact that a large portion, up to 30% to 50%, of the basic variables come from Z. We should therefore expect that the dual purification phase is quite expensive for solving these problems. Table 5.4 shows the computational results for solving the Kennington problems. We see once again that measures PBI and DBI are very good. Although the problems are highly degenerate, the purification is done in less than 15% of the total solution time.

Finally in Table 5.5 we have compared our purification procedure to the cross-over technique implemented in the state of the art LP code CPLEX ([21]). CPLEX results were produced by R.E. Bixby on our HP 715 workstation in April 1995 using the most recent development code. CROWS and CCOLS show the numbers of rows and columns after CPLEX's presolve, and CTIME is the cross-over time reported by CPLEX. AROWS and ACOLS show the numbers of rows and columns after our presolve, and ATIME is the purification time of our code. Considering the difference in problem sizes after presolve, it seems that CPLEX and our code produce optimal bases in a compatible amount of time.

Name	CROWS	CCOLS	CTIME	AROWS	ACOLS	ATIME
PILOT87	1890	4511	17.30	1966	4592	22.98
CRE-B	5529	31723	10.30	5316	31818	15.84
CRE-D	4027	25126	7.00	4081	25188	10.19
KEN-13	11024	24900	10.40	22356	36552	28.63
PDS-06	7141	25861	9.80	9121	27852	28.24
STOCHFOR	10740	9786	10.40	15336	14382	32.65

Tabel 5.5: A comparison between CPLEX and APOS.

5.7 Conclusions

We have in this paper proposed a new approach to combine interior-point and simplex pivoting algorithms. In any iteration of an interior-point algorithm we construct a related LP problem, which approximates the original problem, with a known (strictly) complementary primal-dual solution pair. Thus, we can apply several purification pivoting procedures to compute an optimal basis for the approximate problem in strongly polynomial time. Theoretically, we have shown that, if the approximate problem is constructed from an interior-point iterate sufficiently close to the optimal face, then any optimal basis of the approximate problem is an optimal basis for the original problem. Moreover, if the LP data are rational, the total number of interior-point iterations to create such a sufficient approximate problem is bounded by a polynomial in the data size.

Practically, we have developed a modification of Megiddo's procedure and discussed several implementation issues in solving the approximate problem. We have also reported computational results for our combined method, which indicate that the method works well in practice.

An interesting topic for future research is whether or not the method could be used to do warm-start for an interior-point LP algorithm. One possibility is to iteratively solve the least-squares problem of interior-point algorithms, using the optimal basis generated from the purification procedure as a preconditioner.

a

Kapitel 6

On a homogeneous algorithm for the monotone complementarity problem

We present a generalization of a homogeneous self-dual linear programming (LP) algorithm to solving the monotone complementarity problem (MCP). The algorithm does not need to use any "big-M" parameter or two-phase method, and it generates either a solution converging towards feasibility and complementarity simultaneously or a certificate proving infeasibility. Moreover, if the MCP is polynomially solvable with an interior feasible starting point, then it can be polynomially solved without using or knowing such information at all. To our knowledge, this is the first interior-point and infeasible-starting algorithm for solving the MCP that possesses these desired features. Preliminary computational results are presented.

This paper is co-authored with Yinyu Ye [1].

6.1 Introduction

Consider the monotone complementarity problem (MCP) in the standard form:

$$(MCP) \quad \begin{array}{ll} \text{minimize} & x^T s \\ \text{subject to} & s = f(x), \ (x, s) \geq 0, \end{array}$$

where $f(x)$ is a continuous *monotone* mapping from $R^n_+ := \{x \in R^n : \ x \geq 0\}$ to R^n, $x, s \in R^n$, and ^{T} denotes transpose. In other words, for every $x^1, x^2 \in R^n_+$, we have

$$(x^1 - x^2)^T (f(x^1) - f(x^2)) \geq 0.$$

Denote by ∇f the Jacobian matrix of f, which is positive semi-definite in R^n_+ (see Cottle et al. [36]).

Note any convex optimization problem can be formulated as monotone complementarity problem using the KKT conditions. See Section 6.5 below.

(MCP) is said to be (asymptotically) feasible if and only if there is a *bounded* sequence $(x^t > 0, s^t > 0)$, $t = 1, 2, ...,$ such that

$$\lim_{t \to \infty} \quad s^t - f(x^t) \to 0,$$

where any limit point $(\hat{x}, \hat{s})$ of the sequence is called an (asymptotically) feasible point for (MCP). (MCP) has an interior feasible point if it has an (asymptotically) feasible point $(\hat{x} > 0, \hat{s} > 0)$. (MCP) is said to be (asymptotically) solvable if there is an (asymptotically) feasible $(\hat{x}, \hat{s})$ such that $\hat{x}^T \hat{s} = 0$, where $(\hat{x}, \hat{s})$ is called the "optimal" or "complementary" solution for (MCP). (MCP) is (strongly) infeasible if and only if there is no sequence $(x^t > 0, s^t > 0)$, $t = 1, 2, ...,$ such that

$$\lim_{t \to \infty} \quad s^t - f(x^t) \to 0.$$

---

[1]The University of Iowa, Department of Management Sciences, Iowa City, Iowa 52242, USA. The author is supported in part by NSF Grant DDM-9207347.

Denote the feasible set of (MCP) by $\mathcal{F}$ and the solution set by $\mathcal{S}$. Note that (MCP) is feasible does not imply that (MCP) has a solution. If (MCP) has a solution, then it has a maximal solution (x^*, s^*) where the number positive components in (x^*, s^*) is maximal. Note that the indices of those positive components are invariant among all maximal solutions for (MCP) (see Güler [69]).

Consider a class of (MCP) where f satisfies the following condition. Let

$$\upsilon : (0,1) \rightarrow (1, \infty)$$

be a monotone increasing function such that

$$\|X(f(x+d_x) - f(x) - \nabla f(x) d_x)\|_1 \leq \upsilon(\alpha) d_x^T \nabla f(x) d_x \tag{6.1}$$

whenever

$$d_x \in R^n, \quad x \in R^n_{++} := \{x \in R^n : \ x > 0\}, \quad \left\|X^{-1} d_x\right\|_\infty \leq \alpha < 1.$$

Then, f is said to be scaled Lipschitz in R^n_{++}. Such a condition was proposed by Zhu [158] for linearly constrained convex minimization, and later extended by Potra and Ye [120] for the monotone complementary problem. This condition is included in a more general condition analyzed by Nesterov and Nemirovskii [117] and Jarre [79]. Given $x^0 > 0$ and $s^0 = f(x^0) > 0$ one can develop an interior-point algorithm that generates a maximal complementary solution of the scaled Lipschitz (MCP) in $O(\sqrt{n} \log(1/\epsilon))$ interior-point iterations, where ϵ is the complementarity error.

However, the initial point x^0 is generally unknown. In fact, we don't even know whether such a point exists or not, that is, (MCP) might be infeasible or feasible but have no positive feasible point. To overcome this difficulty, Ye, Todd, and Mizuno [156] recently developed a homogeneous linear programming (LP) algorithm based on the construction of a homogeneous and self-dual LP model (see Xu et al. [148] for a simplification of the model). More recently, Ye [153] extended the model to solving the monotone linear complementarity problem, where f is an affine mapping. However, unlike the original LP homogeneous model, the model constructed in [153] may not be a one-phase model, that is, it may have to find a feasible point first and then search for a solution. This two-phase approach is undesirable in practical implementations.

In this paper, we present a *one-phase* homogeneous model to solving the *general* monotone complementarity problem. To our knowledge, the result is the first MCP algorithm possessing the following desired features:

- It achieves $O(\sqrt{n} \log(1/\epsilon))$-iteration complexity if f satisfies the scaled Lipschitz condition.
- It solves the problem without any regularity assumption concerning the existence of optimal, feasible, or interior feasible points.
- It can start at a positive point, feasible or infeasible, near the central ray of the positive orthant (cone), and it does not need to use any big-M penalty parameter or lower bound.
- If (MCP) has a solution, the algorithm generates a sequence that approaches feasibility and optimality *simultaneously*; if the problem is (strongly) infeasible, the algorithm generates a sequence that converges to a certificate proving infeasibility.

For other interior-point infeasible-starting algorithms for solving (MCP), see Anstreicher and Vial [18], den Hertog et al. [40], Kojima et al. [84, 88], Kortanek et al. [89], Mizuno et al. [109], Monteiro and Wright [115], El-Bakry et al. [46] and references therein.

6.2 A homogeneous MCP model

Let τ and κ be two scalars then consider an augmented homogeneous model related to (MCP):

$$(HMCP) \quad \text{minimize} \quad x^T s + \tau\kappa$$

$$\text{subject to} \quad \begin{pmatrix} s \\ \kappa \end{pmatrix} = \begin{pmatrix} \tau f(x/\tau) \\ -x^T f(x/\tau) \end{pmatrix}, \quad (x, \tau, s, \kappa) \geq 0,$$

where a similar augmented transformation was discussed in Ye [150] and it is closely related to the recession function in convex analyses of Rockafellar [124]. Let

$$\psi(x,\tau) = \begin{pmatrix} \tau f(x/\tau) \\ -x^T f(x/\tau) \end{pmatrix} : \ R^{n+1}_{++} \to R^{n+1}. \tag{6.2}$$

Then, it is easy to verify that $\nabla\psi$ is positive semi-definite as shown in the following lemma.

Lemma 6.2.1 . *Let ∇f be positive semi-definite in R^n_+. Then $\nabla\psi$ is positive semi-definite in R^{n+1}_{++}, i.e. given $(x;\tau) > 0$*

$$(d_x; d_\tau)^T \nabla\psi(x,\tau)(d_x; d_\tau) \geq 0$$

for any $(d_x; d_\tau) \in R^{n+1}$, where

$$\nabla\psi(x,\tau) = \begin{pmatrix} \nabla f(x/\tau) & f(x/\tau) - \nabla f(x/\tau)(x/\tau) \\ -f(x/\tau)^T - (x/\tau)^T \nabla f(x/\tau)^T & (x/\tau)^T \nabla f(x/\tau)(x/\tau) \end{pmatrix}. \tag{6.3}$$

Proof:

$$\begin{aligned} (d_x; d_\tau)^T \nabla\psi(x,\tau)(d_x; d_\tau) = \ & d_x^T \nabla f(x/\tau) d_x - d_x^T \nabla f(x/\tau) x (d_\tau/\tau) \\ & -(d_\tau/\tau) x^T \nabla f(x/\tau)^T d_x + d_\tau^2 x^T \nabla f(x/\tau) x/\tau^2 \\ = \ & (d_x - d_\tau x/\tau)^T \nabla f(x/\tau)(d_x - d_\tau x/\tau) \end{aligned} \tag{6.4}$$

□

Furthermore, we have the following theorem.

Theorem 6.2.1 . *Let ψ be given by (6.2). Then,*

i. *ψ is a continuous homogeneous function in R^{n+1}_{++} with degree 1 and for any $(x;\tau) \in R^{n+1}_{++}$*

$$(x;\tau)^T \psi(x,\tau) = 0$$

and

$$(x;\tau)^T \nabla\psi(x,\tau) = -\psi(x,\tau)^T$$

ii. *If f is a continuous monotone mapping from R^n_+ to R^n, then ψ is a continuous monotone mapping from R^{n+1}_{++} to R^{n+1}.*

iii. *If f is scaled Lipschitz with $\upsilon = \upsilon_f$, then ψ is scaled Lipschitz, that is, it satisfies condition (6.1) with*

$$\upsilon = \upsilon_\psi(\alpha) = \left(1 + \frac{2\upsilon_f(2\alpha/(1+\alpha))}{1-\alpha}\right)\left(\frac{1}{1-\alpha}\right).$$

iv. *$(HMCP)$ is (asymptotically) feasible and every (asymptotically) feasible point is an (asymptotically) complementary solution.*

Now, let $(x^*, \tau^*, s^*, \kappa^*)$ be a maximal complementary solution for $(HMCP)$. Then

v. *(MCP) has a solution if and only if $\tau^* > 0$. In this case, $(x^*/\tau^*, s^*/\tau^*)$ is a complementary solution for (MCP).*

vi. *(MCP) is (strongly) infeasible if and only if $\kappa^* > 0$. In this case, $(x^*/\kappa^*, s^*/\kappa^*)$ is a certificate to prove (strong) infeasibility.*

Proof: The proof of (i) is straightforward.

We now prove (ii). First, ψ is continuous in R^{n+1}_{++} if f is in R^n_{++}. Let $(x^1;\tau^1)$ and $(x^2;\tau^2)$ both in R^{n+1}_{++}. Then, since both x^1/τ^1 and x^2/τ^2 in R^n_{++} and (i), we have

$$\begin{aligned} & (x^1 - x^2; \tau^1 - \tau^2)^T (\psi(x^1,\tau^1) - \psi(x^2,\tau^2)) \\ = \ & -(x^1;\tau^1)^T \psi(x^2,\tau^2) - (x^2;\tau^2)^T \psi(x^1,\tau^1) \\ = \ & -(\tau^2 (x^1)^T f(x^2/\tau^2) - \tau^1 (x^2)^T f(x^2/\tau^2) + \tau^1 (x^2)^T f(x^1/\tau^1) - \tau^2 (x^1)^T f(x^1/\tau^1)) \\ = \ & \tau^1\tau^2 (x^1/\tau^1 - x^2/\tau^2)^T (f(x^1/\tau^1) - f(x^2/\tau^2)) \\ \geq \ & 0. \end{aligned}$$

We now prove (iii). Assume $(x;\tau) \in R^{n+1}_{++}$ and let $(d_x; d_\tau)$ be given such that $\left\|(X^{-1}d_x; \tau^{-1}d_\tau)\right\|_\infty \le \alpha < 1$. To prove ψ is scaled Lipschitz we must bound

$$\left\| \begin{pmatrix} X & 0 \\ 0 & \tau \end{pmatrix} \left(\psi(x+d_x, \tau+d_\tau) - \psi(x,\tau) - \nabla\psi(x,\tau) \begin{pmatrix} d_x \\ d_\tau \end{pmatrix} \right) \right\|_1 . \tag{6.5}$$

From (6.2) and (6.3), the upper part in (6.5) is identical to

$$\begin{aligned} & X\left(f(y+d_y)(\tau+d_\tau) - f(y)\tau - (\nabla f(y)d_x + f(y)d_\tau - \nabla f(y)xd_\tau/\tau)\right) \\ = \; & (\tau+d_\tau)X\left(f(y+d_y) - f(y) - \nabla f(y)d_y\right) \\ = \; & \tau(\tau+d_\tau)Y\left(f(y+d_y) - f(y) - \nabla f(y)d_y\right) \end{aligned} \tag{6.6}$$

where

$$y = x/\tau \quad \text{and} \quad y + d_y = \frac{x+d_x}{\tau+d_\tau}, \tag{6.7}$$

that is,

$$d_y = \frac{\tau d_x - x d_\tau}{\tau(\tau+d_\tau)} = \frac{d_x - (d_\tau/\tau)x}{\tau+d_\tau}. \tag{6.8}$$

Note

$$\begin{aligned} \left\|Y^{-1}d_y\right\|_\infty & = \left\|\tau X^{-1}(\tau d_x - d_\tau x)/(\tau(\tau+d_\tau))\right\|_\infty \\ & = \left\|(\tau X^{-1}d_x - d_\tau e)/(\tau+d_\tau)\right\|_\infty \\ & \le (\left\|X^{-1}d_x\right\|_\infty + \alpha)/(1-\alpha) \\ & \le 2\alpha/(1-\alpha). \end{aligned} \tag{6.9}$$

Per assumption that f is scaled Lipschitz with $\upsilon = \upsilon_f$. It follows for $\alpha \in [0,1)$ that

$$\begin{aligned} & \|\tau(\tau+d_\tau)Y(f(y+d_y) - f(y) - \nabla f(y)d_y)\|_1 \\ \le \; & \tau(\tau+d_\tau)\upsilon_f(2\alpha/(1-\alpha))d_y^T\nabla f(y)d_y \\ = \; & \tfrac{\tau\upsilon_f(2\alpha/(1-\alpha))}{\tau+d_\tau}(d_x - xd_\tau/\tau)^T\nabla f(y)(d_x - xd_\tau/\tau) \\ = \; & \tfrac{\upsilon_f(2\alpha/(1-\alpha))}{1+d_\tau/\tau}(d_x;d_\tau)^T\nabla\psi(x,\tau)(d_x;d_\tau) \\ \le \; & \tfrac{\upsilon_f(2\alpha/(1-\alpha))}{1-\alpha}(d_x;d_\tau)^T\nabla\psi(x,\tau)(d_x;d_\tau) \end{aligned} \tag{6.10}$$

Next we bound the lower part of (6.5). This part is equal to

$$\begin{aligned} & \tau\left(-f(y+d_y)^T(x+d_x) - (-f(y)^Tx) - [-f(y)^Td_x - x^T\nabla f(y)^Td_x/\tau + x^Tf(y)xd_\tau/\tau^2]\right) \\ = \; & \tau\left((x+d_x)^T(-f(y+d_y) + f(y) + \nabla f(y)d_y) - (x+d_x)^T\nabla f(y)d_y + (x/\tau)^T\nabla f(y)d_y(\tau+d_\tau)\right) \\ = \; & \tau\left((x+d_x)^T(-f(y+d_y) + f(y) + \nabla f(y)d_y) - (d_x - d_\tau x/\tau)^T\nabla f(y)d_y\right) \\ = \; & \tau^2(e + X^{-1}d_x)^TY\left(-f(y+d_y) + f(y) + \nabla f(y)d_y\right) - \tau(\tau+d_\tau)d_y^T\nabla f(y)d_y. \end{aligned}$$

Thus, using (6.4) and (6.9)

$$\begin{aligned} & \left|\tau\left(-f(y+d_y)^T(x+d_x) - (-f(y)^Tx) - [-f(y)^Td_x - x^T\nabla f(y)^Td_x/\tau + x^Tf(y)xd_\tau/\tau^2]\right)\right| \\ \le \; & \tau^2\left\|e + X^{-1}d_x\right\|_\infty \|Y(-f(y+d_y) + (-f(y) - \nabla f(y)d_y))\|_1 + |\tau(\tau+d_\tau)|d_y^T\nabla f(y)d_y \\ \le \; & \left(\tau^2(1+\alpha)\upsilon_f(2\alpha/(1-\alpha)) + \tau(\tau+d_\tau)\right)d_y^T\nabla f(y)d_y \\ = \; & \tfrac{\tau^2(1+\alpha)\upsilon_f(2\alpha/(1-\alpha)) + \tau(\tau+d_\tau)}{(\tau+d_\tau)^2}(d_x^T - d_\tau x/\tau)^T\nabla f(y)(d_x - d_\tau x/\tau) \\ = \; & \tfrac{(1+\alpha)\upsilon_f(2\alpha/(1-\alpha)) + (1+d_\tau/\tau)}{(1+d_\tau/\tau)^2}(d_x;d_\tau)^T\nabla\psi(x,\tau)(d_x;d_\tau) \\ \le \; & \left(\tfrac{(1+\alpha)\upsilon_f(2\alpha/(1-\alpha))}{(1-\alpha)^2} + \tfrac{1}{1-\alpha}\right)(d_x;d_\tau)^T\nabla\psi(x,\tau)(d_x;d_\tau). \end{aligned} \tag{6.11}$$

The sum of (6.10) and (6.11) is equal to

$$\upsilon_\psi(\alpha)(d_x;d_\tau)^T\nabla\psi(x,\tau)(d_x;d_\tau)$$

and it bounds the term in (6.5) leading to the desired result.

We now prove (iv). Take $x^t = (1/2)^te$, $\tau^t = (1/2)^t$, $s^t = (1/2)^te$, and $\kappa^t = (1/2)^t$. Then we see, as $t \to \infty$,

$$s^t - \tau^tf(x^t/\tau^t) = (1/2)^t(e - f(e)) \to 0$$

and

$$\kappa^t + (x^t)^T f(x^t/\tau^t) = (1/2)^t(1 + e^T f(e)) \to 0.$$

Thus, the system is (asymptotically) feasible. Let $(\hat{x}, \hat{\tau}, \hat{s}, \hat{\kappa}) \geq 0$ be any (asymptotically) feasible point for $(HMCP)$, then we clearly see from (i)

$$\hat{x}^T \hat{s} + \hat{\tau}\hat{\kappa} = (\hat{x}; \hat{\tau})^T \psi(\hat{x}, \hat{\tau}) = 0.$$

We now prove (v). If $(x^*, \tau^*, s^*, \kappa^*)$ is a solution for $(HMCP)$ and $\tau^* > 0$. Then we have

$$s^*/\tau^* = f(x^*/\tau^*) \quad \text{and} \quad (x^*)^T s^*/(\tau^*)^2 = 0,$$

that is, $(x^*/\tau^*, s^*/\tau^*)$ is a solution for (MCP). Let $(\hat{x}, \hat{s})$ be a solution to (MCP). Then for any $\tau = 1$, $x = \hat{x}$, $s = \hat{s}$, $\kappa = 0$ is a solution for $(HMCP)$. Thus, every maximal solution of $(HMCP)$ must have $\tau^* > 0$.

Finally, we prove (vi). Consider the set

$$R_{++} = \{s - f(x) \in R^n : \quad (x, s) > 0\}.$$

As proved by Güler [69] and Kojima et al. [88], R_{++} is an open convex set. If (MCP) is strongly infeasible, then we must have $0 \notin \bar{R}_{++}$ where $\bar{R}_{++}$ represents the closure of R_{++}. Thus, there is a hyperplane separates 0 and $\bar{R}_{++}$, that is, there are a vector $a \in R^n$ with $\|a\| = 1$ and a positive number ξ such that

$$a^T(s - f(x)) \geq \xi > 0 \quad \forall\, x \geq 0,\ s \geq 0. \tag{6.12}$$

For $j = 1, 2, ..., n$, set s_j sufficiently large, but fix x and the rest of s, it must be true $a_j \geq 0$. Thus,

$$a \geq 0, \quad \text{or} \quad a \in R^n_+.$$

On the other hand, for any fixed x, we set $s = 0$ and see that

$$-\, a^T f(x) \geq \xi > 0 \quad \forall\, x \geq 0. \tag{6.13}$$

In particular,

$$-\, a^T f(ta) \geq \xi > 0 \quad \forall\, t \geq 0. \tag{6.14}$$

From the monotone property of f, for every $x \in R^n_+$ and any $t \geq 0$ we have

$$(tx - x)^T (f(tx) - f(x)) \geq 0.$$

Thus,

$$x^T f(tx) \geq x^T f(x) \tag{6.15}$$

and

$$\lim_{t\to\infty} x^T f(tx)/t \geq 0. \tag{6.16}$$

Thus, from (6.14) and (6.16)

$$\lim_{t\to\infty} a^T f(ta)/t = 0.$$

For an $x \in R^n_+$, denote

$$f^\infty(x) := \lim_{t\to\infty} f(tx)/t,$$

where $f^\infty(x)$ represents the limit of any subsequence and its values may include ∞ or $-\infty$.

We now prove $f^\infty(a) \geq 0$. Suppose that $f^\infty(a)_j < -\delta$. Then consider the vector $x = a + \epsilon e_j$ where e_j is the vector with the jth component being 1 and zeros everywhere. Then, for ϵ sufficiently small and t sufficiently large we have

$$\begin{aligned}
x^T f(tx)/t &= (a + \epsilon e_j)^T f(t(a + \epsilon e_j))/t \\
&= a^T f(t(a + \epsilon e_j))/t + \epsilon e_j^T f(t(a + \epsilon e_j))/t \\
&< \epsilon e_j^T f(t(a + \epsilon e_j))/t \quad \text{(from (6.13))} \\
&= \epsilon \frac{f(t(a + \epsilon e_j))_j - f(ta)_j}{t} + \epsilon \frac{f(ta)_j}{t} \\
&\leq \epsilon \left(O(\epsilon) + \frac{f(ta)_j}{t} \right) \quad \text{(from continuity of } f\text{)} \\
&= \epsilon(O(\epsilon) - \delta/2) \\
&\leq -\epsilon\delta/4.
\end{aligned}$$

$\tau^* \setminus \kappa^*$	=0	>0
$=0$	?	Infeasible(finite certificate)
>0	Solvable(finite solution)	Impossible

Figur 6.1: Four possible combinations of τ and κ.

But this contradicts to relation (6.16). Thus, we must have

$$f^{\infty}(a) \geq 0.$$

We now further prove that $f^{\infty}(a)$ is bounded. Consider

$$\begin{array}{rcl} 0 & \leq & (ta-e)^T(f(ta)-f(e))/t \\ & = & a^Tf(ta) - e^Tf(ta)/t - a^Tf(e) + e^Tf(e)/t \\ & < & -e^Tf(ta)/t - a^Tf(e) + e^Tf(e)/t. \end{array}$$

Take a limit as $t \to \infty$ from both sides, we have

$$e^Tf^{\infty}(a) \leq -a^Tf(e).$$

Thus, $f^{\infty}(a) \geq 0$ is bounded. Again, we have $a^Tf(ta) \leq -\xi$ from (6.14) and $a^Tf(ta) \geq a^Tf(a)$ from (6.15). Thus, $\lim a^Tf(ta)$ is bounded. To summarize, $(HMCP)$ has an asymptotical solution $(x^* = a, \tau^* = 0, s^* = f^{\infty}(a), \kappa^* = \lim -a^Tf(ta) \geq \xi)$.

Conversely, if there is a bounded sequence $(x^k > 0, \tau^k > 0, s^k > 0, \kappa^k > 0)$

$$\lim s^k = \lim \tau^k f(x^k/\tau^k) \geq 0, \quad \lim \kappa^k = \lim -(x^k)^Tf(x^k/\tau^k) \geq \xi > 0.$$

Then, we claim that there is no feasible point $(x \geq 0, s \geq 0)$ such that $s - f(x) = 0$. We prove this fact by contradiction. If there is one. Then,

$$\begin{array}{rcl} 0 & \leq & ((x^k;\tau^k)-(x;1))^T(\psi(x^k,\tau^k)-\psi(x,1)) \\ & = & (x^k-x)^T(\tau^kf(x^k/\tau^k)-f(x)) + (\tau^k-1)^T(xf(x)-(x^k)^Tf(x^k/\tau^k)). \end{array}$$

Therefore,

$$(x^k)^Tf(x^k/\tau^k) \geq (x^k)^Tf(x) + \tau^kx^Tf(x^k/\tau^k) - \tau^kx^Tf(x).$$

Since the first two terms at the right-hand side are positive and $\lim \tau^k = 0$, we must have

$$\lim (x^k)^Tf(x^k/\tau^k) \geq 0,$$

which is a contradiction to $\kappa^k = -(x^k)^Tf(x^k/\tau^k) \geq \xi > 0$. Also, any limit of x^k is a separating hyperplane, i.e., a certificate proving infeasibility.

□

Assume $(x^*, \tau^*, y^*, s^*, \kappa^*)$ is a maximal complementarity solution, then in Figure 6.1 we show the four possible combinations of τ^* and κ^*. If τ^* and κ^* are both zero, then there do not exists a finite solution or a finite certicate proving infeasibility. This cannot happen if f is an affine function.

6.3 Central path of the homogeneous model

Due to Theorem 6.2.1, we can solve (MCP) by finding a maximal complementary solution of $(HMCP)$. Select $x^0 > 0$, $s^0 > 0$, $\tau^0 > 0$ and $\kappa^0 > 0$ and let the residual vectors

$$r^0 = s^0 - \tau^0 f(x^0/\tau^0), \quad z^0 = \kappa^0 + (x^0)^Tf(x^0/\tau^0).$$

Also let

$$\bar{n} = (r^0)^Tx^0 + z^0\tau^0 = (x^0)^Ts^0 + \tau^0\kappa^0.$$

For simplicity, we set

$$x^0 = e,\ \tau^0 = 1,\ s^0 = e,\ \kappa^0 = 1,\ \theta^0 = 1,$$

with

$$X^0 s^0 = e \quad \text{and} \quad \tau^0 \kappa^0 = 1,$$

where X^0 is the diagonal matrix of x^0. Note that $\bar{n} = n + 1$ in this setting.

We present the next theorem.

Theorem 6.3.1 . *Consider* $(HMCP)$.

i. *For any* $0 < \theta \le 1$, *there exists a strictly positive point* $(x > 0, \tau > 0, s > 0, \kappa > 0)$ *such that*

$$\begin{pmatrix} s \\ \kappa \end{pmatrix} - \psi(x,\tau) = \begin{pmatrix} s - \tau f(x/\tau) \\ \kappa + x^T f(x/\tau) \end{pmatrix} = \theta \begin{pmatrix} r^0 \\ z^0 \end{pmatrix}. \tag{6.17}$$

ii. *Starting from* $(x^0 = e, \tau^0 = 1, s^0 = e, \kappa^0 = 1)$, *for any* $0 < \theta \le 1$ *there is a unique strictly positive point* $(x(\theta), \tau(\theta), s(\theta), \kappa(\theta))$ *that satisfies equation (6.17) and*

$$\begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \theta e. \tag{6.18}$$

iii. *For any* $0 < \theta \le 1$, *the solution* $(x(\theta), \tau(\theta), s(\theta), \kappa(\theta))$ *in [ii] is bounded. Thus,*

$$\begin{aligned} \mathcal{C}(\theta) := \quad & \left\{ (x,\tau,s,\kappa) : \begin{pmatrix} s \\ \kappa \end{pmatrix} - \psi(x,\tau) = \theta \begin{pmatrix} r^0 \\ z^0 \end{pmatrix}, \right. \\ & \left. \qquad \begin{pmatrix} Xz \\ \tau\kappa \end{pmatrix} = \theta e, \quad 0 < \theta \le 1 \right\} \end{aligned} \tag{6.19}$$

is a continuous bounded trajectory.

iv. *The limit point* $(x(0), \tau(0), s(0), \kappa(0))$ *is a maximal complementary solution for* $(HMCP)$.

Proof. We prove [i]. Again, the set

$$H_{++} := \left\{ \begin{pmatrix} s \\ \kappa \end{pmatrix} - \psi(x,\tau) : \quad (x,\tau,s,\kappa) > 0. \right\}$$

is open and convex. (Note that ψ is a continuous monotone mapping from R^{n+1}_{++}, rather than R^{n+1}_{+}, to R^{n+1}. Nevertheless, the proof is a straightforward extension of Kojima et al. [88].) We have $(r^0; z^0) \in H_{++}$ by construction. On the other hand, $0 \in \bar{H}_{++}$ from Theorem 6.2.1. Thus,

$$\theta \begin{pmatrix} r^0 \\ z^0 \end{pmatrix} \in H_{++}.$$

The proof of [ii] is due to Güler [69] and Kojima et al. [88].

We now prove [iii]. Again, the existence is due to Güler [69] and Kojima et al. [88] We prove the boundedness. Assume $(x, \tau, s, \kappa) \in \mathcal{C}(\theta)$ then

$$\begin{aligned}
&(x;\tau)^T (r^0; z^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) - (x;\tau)^T \psi(x^0;\tau^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - (s;\kappa)^T (x^0;\tau^0) - (x;\tau)^T \psi(x^0;\tau^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - (x^0;\tau^0)^T (\theta(r^0;z^0) + \psi(x,\tau)) \\
& -(x;\tau)^T \psi(x^0;\tau^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - \theta(x^0;\tau^0)^T (r^0;z^0) - (x^0;\tau^0)^T \psi(x,\tau) \\
& -(x;\tau)^T \psi(x^0;\tau^0) \\
\ge \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - \theta(x^0;\tau^0)^T (r^0;z^0) - (x;\tau)^T \psi(x,\tau) \\
& -(x^0;\tau^0)^T \psi(x^0;\tau^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - \theta(x^0;\tau^0)^T (r^0;z^0) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) \\
& -\theta(x^0;\tau^0)^T ((s^0;\kappa^0) - \psi(x^0,\tau^0)) \\
= \;& (x;\tau)^T (s^0;\kappa^0) + (s;\kappa)^T (x^0;\tau^0) - \theta(x^0;\tau^0)^T (s^0;\kappa^0).
\end{aligned}$$

Also for $0 < \theta \le 1$,

$$\begin{array}{rcl} \theta(x;\tau)^T(r^0;z^0) & & \\ & = & (x;\tau)^T((s;\kappa) - \psi(x,\tau)) \\ & = & (x;\tau)^T(s;\kappa) = \theta(n+1) \\ & = & \theta(x^0;\tau^0)^T(s^0;\kappa^0). \end{array}$$

From the above two relations, we have

$$(x;\tau)^T(s^0;\kappa^0) + (s;\kappa)^T(x^0;\tau^0) \le (1+\theta)(x^0;\tau^0)^T(s^0;\kappa^0).$$

Thus, $(x;\tau;s;\kappa)$ is bounded.

Finally, we prove (iv). Let $(x^*,\tau^*,s^*,\kappa^*)$ be any maximal complementarity solution for $(HMCP)$ such that

$$(s^*;\kappa^*) = \psi(x^*;\tau^*) \quad \text{and} \quad (x^*)^T s^* + \tau^*\kappa^* = 0,$$

and it is normalized by

$$\begin{array}{rcl} (r^0;z^0)^T(x^*;\tau^*) & = & (r^0;z^0)^T(x^0;\tau^0) \\ & = & (s^0;\kappa^0)^T(x^0;\tau^0) = (n+1). \end{array}$$

For any $0 < \theta \le 1$, let (x,τ,s,κ) be the solution on the path. Then, we have

$$\begin{array}{rl} \multicolumn{2}{l}{((x;\tau) - (x^*;\tau^*))^T((s;\kappa) - (s^*;\kappa^*))} \\ = & ((x;\tau) - (x^*;\tau^*))^T(\psi(x;\tau) - \psi(x^*;\tau^*)) + \theta(r^0;z^0)^T((x;\tau) - (x^*;\tau^*)) \\ \ge & \theta(r^0;z^0)^T((x;\tau) - (x^*;\tau^*)). \end{array}$$

Therefore,

$$\begin{array}{rcl} (x;\tau)^T(s^*;\kappa^*) + (s;\kappa)^T(x^*;\tau^*) & \le & (x;\tau)^T(s;\kappa) - \theta(r^0;z^0)^T((x;\tau) - (x^*;\tau^*)) \\ & = & (x;\tau)^T(s;\kappa) - (x;\tau)^T(s;\kappa) + \theta(r^0;z^0)^T(x^*;\tau^*) \\ & = & \theta(r^0;z^0)^T(x^*;\tau^*) \\ & = & \theta(n+1). \end{array}$$

Using $x_j s_j = \theta$ we obtain,

$$\begin{array}{rcl} (x;\tau)^T(s^*;\kappa^*) + (s;\kappa)^T(x^*;\tau^*) & = & \theta \sum \frac{s_j^*}{s_j} + \frac{\kappa^*}{\tau} + \sum \frac{x_j^*}{x_j} + \frac{\tau^*}{\kappa} \\ & \le & \theta(n+1). \end{array}$$

Thus, we have

$$\frac{s_j^*}{s_j} \le (n+1), \quad \text{and} \quad \frac{\kappa^*}{\kappa} \le (n+1)$$

and

$$\frac{x_j^*}{x_j} \le (n+1), \quad \text{and} \quad \frac{\tau^*}{\tau} \le (n+1).$$

Thus, the limit point, $(x(0),\tau(0),s(0),\kappa(0))$, is a maximal complementarity solution for $(HMCP)$.

□

We now present an interior-point algorithm that generates iterates within a neighborhood of $\mathcal{C}(\theta)$. For simplicity, in what follows we let $x := (x;\tau) \in R^{n+1}$, $s := (s;\kappa) \in R^{n+1}$, and $r^0 := (r^0;z^0)$. Recall that, for any $x, s > 0$

$$x^T\psi(x) = 0 \quad \text{and} \quad x^T\nabla\psi(x) = -\psi(x)^T. \tag{6.20}$$

Furthermore, ψ is monotone and satisfies the scaled Lipschitz. We will use these facts frequently in our analyses.

6.4 The homogeneous interior-point algorithm

At iteration k with iterate $(x^k, s^k) > 0$, the algorithm solves a system of linear equations for direction (d_x, d_s) from

$$d_s - \nabla\psi(x^k)d_x = -\eta r^k \tag{6.21}$$

and

$$X^k d_s + S^k d_x = \gamma\mu^k e - X^k s^k, \tag{6.22}$$

where η and γ are proper given parameters between 0 and 1, and

$$r^k = s^k - \psi(x^k) \quad \text{and} \quad \mu^k = \frac{(x^k)^T s^k}{n+1}.$$

First we prove the following lemma.

Lemma 6.4.1 . *The direction (d_x, d_s) satisfies*

$$d_x^T d_s = d_x^T \nabla\psi(x^k) d_x + \eta(1-\eta-\gamma)(n+1)\mu^k.$$

Proof: Premultiplying each side of (6.21) by d_x^T gives

$$d_x^T d_s - d_x^T \nabla\psi(x^k) d_x = -\eta d_x^T (s^k - \psi(x^k)) \tag{6.23}$$

Multiplying each side of (6.21) by x^k and using (6.20) give

$$\begin{aligned} (x^k)^T d_s + \psi(x^k) d_x &= -\eta(x^k)^T r^k \\ &= -\eta(x^k)^T(s^k - \psi(x^k)) \\ &= -\eta(x^k)^T s^k \\ &= -\eta(n+1)\mu^k \end{aligned} \tag{6.24}$$

These two equalities in combination with (6.22) implies

$$\begin{aligned} d_x^T d_s &= d_x^T \nabla\psi(x^k) d_x - \eta(d_x^T s^k + d_s^T x^k + \eta(n+1)\mu^k) \\ &= d_x^T \nabla\psi(x^k) d_x - \eta(-(n+1)\mu^k + \gamma(n+1)\mu^k + \eta(n+1)\mu^k) \\ &= d_x^T \nabla\psi(x^k) d_x + \eta(1-\gamma-\eta)(n+1)\mu^k \end{aligned}$$

□

For a step-size $\alpha > 0$, let the new iterate

$$x^+ := x^k + \alpha d_x > 0, \tag{6.25}$$

and

$$\begin{aligned} s^+ &:= s^k + \alpha d_s + \psi(x^+) - \psi(x^k) - \alpha\nabla\psi(x^k) d_x \\ &= \psi(x^+) + (s^k - \psi(x^k)) + \alpha(d_s - \nabla\psi(x^k) d_x) \\ &= \psi(x^+) + (s^k - \psi(x^k)) - \alpha\eta(s^k - \psi(x^k)) \\ &= \psi(x^+) + (1-\alpha\eta) r^k. \end{aligned} \tag{6.26}$$

The last two equalities come from (6.21) and the definition of r^k. Also let

$$r^+ = s^+ - \psi(x^+).$$

Then, we have

Lemma 6.4.2 . *Consider the new iterate (x^+, s^+) given by (6.25) and (6.26).*

a). $r^+ = (1-\alpha\eta) r^k$

b). $(x^+)s^+ = (x^k)^T s^k (1 - \alpha(1-\gamma)) + \alpha^2 \eta(1-\eta-\gamma)(n+1)\mu^k$

Proof: From (6.26)

$$\begin{array}{rcl} r^+ &=& s^+ - \psi(x^+) \\ &=& (1-\alpha\eta)r^k. \end{array}$$

Next we prove b). Using (6.20), (6.22), and Lemma 6.4.1, we have

$$\begin{array}{rcl} (x^+)^T s^+ &=& (x^+)^T(s^k + \alpha d_s + \psi(x^+) - \psi(x^k) - \alpha\nabla\psi(x^k)d_x) \\ &=& (x^+)^T(s^k+\alpha d_s) - (x^+)^T(\psi(x^k) + \alpha\nabla\psi(x^k)d_x) \\ &=& (x^+)^T(s^k+\alpha d_s) - (x^k+\alpha d_x)^T(\psi(x^k) + \alpha\nabla\psi(x^k)d_x) \\ &=& (x^+)^T(s^k+\alpha d_s) - \alpha(x^k)^T\nabla\psi(x^k)d_x - \alpha d_x^T\psi(x^k) - \alpha^2 d_x^T\nabla\psi(x^k)d_x \\ &=& (x^+)^T(s^k+\alpha d_s) - \alpha^2 d_x^T\nabla\psi(x^k)d_x \\ &=& (x^k+\alpha d_x)^T(s^k+\alpha d_s) - \alpha^2 d_x^T\nabla\psi(x^k)d_x \\ &=& (x^k)^T s^k + \alpha(d_x^T s^k + d_s^T x^k) + \alpha^2(d_x^T d_s - d_x^T\nabla\psi(x^k)d_x) \\ &=& (x^k)^T s^k + \alpha(d_x^T s^k + d_s^T x^k) + \alpha^2\eta(1-\eta-\gamma)(n+1)\mu^k \\ &=& (1-\alpha(1-\gamma))(x^k)^T s^k + \alpha^2\eta(1-\eta-\gamma)(n+1)\mu^k. \end{array}$$

□

This lemma shows that, for setting $\eta = 1-\gamma$, the infeasibility residual and the complementarity gap are reduced at an exactly same rate, which is used to be the case in the homogeneous linear programming algorithm. Now we prove the following.

Theorem 6.4.1 . *Assume that ψ is scaled Lipschitz with $\upsilon = \upsilon_\psi$ and at iteration k*

$$\|X^k s^k - \mu^k e\| \le \beta\mu^k, \quad \mu^k = \frac{(x^k)^T s^k}{n+1}$$

where

$$\beta = \frac{1}{3+4\upsilon_\psi(\sqrt{2}/2)} \le 1/3.$$

Furthermore, let $\eta = \beta/\sqrt{n+1}$, $\gamma = 1-\eta$, and $\alpha = 1$ in the algorithm. Then, the new iterate

$$x^+ > 0, \quad s^+ = \psi(x^+) + (1-\eta)r^k > 0,$$

and

$$\|X^+ s^+ - \mu^+ e\| \le \beta\mu^+, \quad \mu^+ = \frac{(x^+)^T s^+}{n+1}$$

Proof: It follows from lemma 6.4.2 that $\mu^+ = \gamma\mu^k$. From (6.22) we further have

$$S^k d_x + X^k d_s = -X^k s^k + \mu^+ e.$$

Hence,

$$D^{-1}d_x + Dd_s = -(X^k S^k)^{-1/2}(X^k s^k - \mu^+ e)$$

where $D = (X^k)^{1/2}(S^k)^{-1/2}$. Note

$$d_x^T d_s = d_x^T\nabla\psi(x^k)d_x \ge 0$$

from Lemma 6.4.1 and $\gamma = 1-\eta$. This together with the assumption of the theorem imply

$$\|D^{-1}d_x\|^2 + \|Dd_s\|^2 \le \|(X^k S^k)^{-1/2}(X^k s^k - \mu^+ e)\|^2 \le \frac{\|X^k s^k - \mu^+ e\|^2}{(1-\beta)\mu^k}.$$

Also note

$$\begin{array}{rcl} \|X^k s^k - \mu^+ e\|^2 &=& \|X^k s^k - \mu^k e + (1-\gamma)\mu^k e\|^2 \\ &=& \|X^k s - \mu^k e\|^2 + ((1-\gamma)\mu^k)^2\|e\|^2 \\ &\le& (\beta^2 + \eta^2(n+1))(\mu^k)^2 \\ &=& 2\beta^2(\mu^k)^2. \end{array}$$

Thus,

$$\left\|(X^k)^{-1}d_x\right\| = \left\|(X^kS^k)^{-1/2}D^{-1}d_x\right\| \le \frac{\left\|D^{-1}d_x\right\|}{\sqrt{(1-\beta)\mu^k}}$$

$$\le \frac{\sqrt{2}\beta\mu^k}{(1-\beta)\mu^k} = \frac{\sqrt{2}\beta}{1-\beta} \le \frac{\sqrt{2}}{2},$$

since $\beta \le 1/3$. This implies that $x^+ = x^k + d_x > 0$. Furthermore, we have

$$\begin{aligned}
\|D_x d_s\| &= \left\|D^{-1}D_x D d_s\right\| \\
&\le \left\|D^{-1}d_x\right\| \|Dd_s\| \\
&\le (\left\|D^{-1}d_x\right\|^2 + \|Dd_s\|^2)/2 \\
&\le \left\|(X^kS^k)^{-1/2}(X^ks^k - \mu^+)\right\|^2 /2 \\
&\le \frac{\left\|X^ks^k - \mu^+ e\right\|^2}{2(1-\beta)\mu^k} \\
&\le \frac{2\beta^2(\mu^k)^2}{2(1-\beta)\mu^k} \\
&= \frac{\beta^2\mu^k}{1-\beta},
\end{aligned}$$

and

$$d_x^T d_s = d_x^T D^{-1} D d_s \le \left\|D^{-1}d_x\right\| \|Dd_s\| \le \frac{\beta^2\mu^k}{1-\beta}.$$

Consider

$$\begin{aligned}
X^+s^+ - \mu^+ e &= X^+(s^k + d_s + \psi(x^+) - \psi(x^k) - \nabla\psi(x^k)d_x) - \mu^+ e \\
&= (X^k + D_x)(s^k + d_s) - \mu^+ e + X^+(\psi(x^+) - \psi(x^k) - \nabla\psi(x^k)d_x) \\
&= D_x d_s + X^+(\psi(x^+) - \psi(x^k) - \nabla\psi(x^k)d_x).
\end{aligned}$$

Using that ψ is the scaled Lipschitz, $d_x^T\nabla\psi(x^k)d_x = d_x^T d_s$ and the above four relations we obtain

$$\begin{aligned}
\left\|X^+s^+ - \mu^+ e\right\| &= \left\|D_x d_s + X^+(\psi(x^+) - \psi(x^k) - \alpha\nabla\psi(x^k)d_x)\right\| \\
&= \left\|D_x d_s + (X^k)^{-1}X^+X^k(\psi(x^+) - \psi(x^k) - \alpha\nabla\psi(x^k)d_x)\right\| \\
&\le \|D_x d_s\| + \left\|(X^k)^{-1}X^+\right\|_\infty \left\|X^k(\psi(x^+) - \psi(x^k) - \nabla\psi(x^k)d_x)\right\|_1 \\
&\le \|D_x d_s\| + 2\left\|X^k(\psi(x^+) - \psi(x^k) - \nabla\psi(x^k)d_x)\right\|_1 \\
&\le \|D_x d_s\| + 2v_\psi(\sqrt{2}/2)d_x^T\nabla\psi(x^k)d_x \\
&= \|D_x d_s\| + 2v_\psi(\sqrt{2}/2)d_x^T d_s \\
&\le \frac{\beta^2\mu^k}{1-\beta} + 2v_\psi(\sqrt{2}/2)\frac{\beta^2\mu^k}{1-\beta} \\
&\le \frac{(1+2v_\psi(\sqrt{2}/2))\beta^2\mu^k}{1-\beta}.
\end{aligned}$$

Finally, $\beta = 1/(3 + 4v_\psi(\sqrt{2}/2))$ implies that

$$\frac{(1+2v_\psi(\sqrt{2}/2))\beta^2\mu^k}{1-\beta} \le \frac{\beta}{2}$$

and

$$\left\|X^+s^+ - \mu^+ e\right\| \le \beta\mu^k/2 < \beta\gamma\mu^k = \beta\mu^+.$$

It is easy to verify that $x^+ > 0$ and $\|X^+s^+ - \mu^+ e\| < \beta\mu^+$ implies $s^+ > 0$.

□

The above theorem shows that the homogeneous algorithm will generate a sequence $(x^k, s^k) > 0$ with $(x^{k+1}, s^{k+1}) := (x^+, s^+)$ such that $s^k = \psi(x^k) + r^k$ and $\|X^k s^k - \mu^k\| \leq \beta\mu^k$, where both $\|r^k\|$ and $(x^k)^T s^k$ converge to zero at a global rate $\gamma = 1 - \beta/\sqrt{n+1}$. We see that if $\upsilon_\psi(\sqrt{2}/2)$ is a constant, or $\upsilon_f(2/(1+\sqrt{2}))$ is a constant in (MCP) due to (iii) of Theorem 6.2.1, then it results in an $O(\sqrt{n}\log(1/\epsilon))$ iteration algorithm with error ϵ. It generates a maximal solution for $(HMCP)$, which is either a solution or a certificate proving infeasibility for (MCP), due to (v) and (vi) of Theorem 6.2.1.

One more comment is that our results should hold for the case that $f(x)$ is a continuous *monotone* mapping from R^n_{++} to R^n. In other words, $f(x)$ may not exist at the boundary of R^n_+. In general, many convex optimization problems can be solved, either obtaining a solution or proving infeasibility or unboundedness, by solving their Karush-Kuhn-Tucker (KKT) system which is a monotone complementarity problem.

6.5 A preliminary implementation

In this section we discuss a preliminary implementation of the homogeneous algorithm proposed in the previous sections. We restrict our implementation at this moment to the linearly constrained convex programming problem which has the form

$$\begin{array}{rl} \text{minimize} & c(x) \\ \text{subject to} & Ax = b, \\ & x \geq 0, \end{array} \tag{6.27}$$

where $A \in R^{m\times n}$ and $b \in R^m$. Without loss of generality we will assume $n > m$ and A is of full row rank. We assume c is convex and at least twice differentiable on R^n_{++}. (Others have proposed algorithms for solving this class of problems, see for example Vial [145] and Kortanek et al. [89].)

The Wolfe dual to (6.27) is

$$\begin{array}{rl} \text{maximize} & c(x) - \nabla c(x)x + b^T y \\ \text{subject to} & \nabla c(x)^T - A^T y - s = 0, \\ & x, s \geq 0. \end{array} \tag{6.28}$$

The KKT optimality conditions to (6.27) are

$$\begin{array}{rcll} \nabla c(x)^T - A^T y - s & = & 0, & s \geq 0, \\ Ax & = & b, & x \geq 0, \\ Xs & = & 0, & \end{array} \tag{6.29}$$

where y and s are the Lagrange multipliers corresponding to the equality and inequality constraints in (6.27), respectively. Now the system (6.29) is an (MCP) problem with free variables y. Introducing the homogenization variable τ and slack variable κ into (6.29) we obtain an $(HMCP)$ problem

$$\begin{array}{rcll} \tau\nabla c(x/\tau)^T - A^T y - s & = & 0, & s \geq 0,\ y \text{ free}, \\ Ax - b\tau & = & 0, & x \geq 0,\ \tau \geq 0, \\ -x^T\nabla c(x/\tau)^T + b^T y - \kappa & = & 0, & \kappa \geq 0, \\ Xs & = & 0, & \\ \tau\kappa & = & 0. & \end{array} \tag{6.30}$$

It should be noticed that all previous $(HMCP)$ results hold for system (6.30), although it has free variables y.

6.5.1 The search direction

The search direction used in our algorithm is the same as that proposed in Section 6.4. In other words, we solve a perturbed version of (6.30) using Newton's method. Let $(x^k > 0; \tau^k > 0; y^k; s^k > 0; \kappa^k > 0)$

be the current point on iteration k. Now define the residual vectors

$$\begin{array}{rcl} r_D^k &=& s^k + A^T y^k - \tau^k \nabla c(x^k/\tau^k)^T, \\ r_G^k &=& \kappa^k + (x^k)^T \nabla c(x^k/\tau^k)^T - b^T y^k, \\ r_P^k &=& b\tau^k - Ax^k, \end{array}$$

and let

$$H^k = \nabla^2 c(x^k/\tau^k), \quad g^k = \nabla c(x^k/\tau^k), \quad \text{and} \quad \xi^k = x^k/\tau^k.$$

Then, the Newton search direction is given by

$$\begin{bmatrix} H^k & (g^k)^T - H^k\xi^k & -A^T \\ -g^k - (\xi^k)^T H^k & (\xi^k)^T H^k \xi^k & b^T \\ A & -b & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_\tau \\ d_y \end{bmatrix} - \begin{bmatrix} d_s \\ d_\kappa \\ 0 \end{bmatrix} = \eta \begin{bmatrix} r_D^k \\ r_G^k \\ r_P^k \end{bmatrix} \tag{6.31}$$

and

$$\begin{array}{rcl} S^k d_x + X^k d_s &=& -X^k s^k + \gamma\mu^k e, \\ \kappa^k d_\tau + \tau^k d_\kappa &=& -\tau^k\kappa^k + \gamma\mu^k, \end{array} \tag{6.32}$$

where $\mu^k = ((x^k)^T s^k + \tau^k\kappa^k)/(n+1)$, and η and γ are parameters to be chosen lately. If we use (6.32) to eliminate $(d_s; d_\kappa)$ we obtain the reduced Newton equation system

$$\begin{array}{l} \begin{bmatrix} H^k + (X^k)^{-1}S^k & (g^k)^T - H^k\xi^k & -A^T \\ -g^k - (\xi^k)^T H^k & (\xi^k)^T H^k \xi^k + (\tau^k)^{-1}\kappa^k & b^T \\ A & -b & 0 \end{bmatrix} \begin{bmatrix} d_x \\ d_\tau \\ d_y \end{bmatrix} \\ = \begin{bmatrix} \eta r_D^k + (X^k)^{-1}(-X^k s^k + \gamma\mu^k e) \\ \eta r_G^k + (\tau^k)^{-1}(-\tau^k\kappa^k + \gamma\mu^k) \\ \eta r_P^k \end{bmatrix} \end{array} \tag{6.33}$$

The coefficient matrix in (6.33) is in general not symmetric and cannot be made symmetric by a simple row or column scaling. (The matrix is skew-symmetric if c is a linear function.)

Let

$$K = \begin{bmatrix} H^k + (X^k)^{-1}S^k & -A^T \\ A & 0 \end{bmatrix}. \tag{6.34}$$

It is easy to verify that if A is of full row rank then K is a non-singular matrix (see Lemma 6.5.1 below). Now define h^k and p^k as solutions to the linear systems

$$Kh^k = \begin{bmatrix} (g^k)^T - H^k\xi^k \\ -b \end{bmatrix} \quad \text{and} \quad Kp^k = \begin{bmatrix} \eta r_D^k + (X^k)^{-1}(-X^k s^k + \gamma\mu^k e) \\ \eta r_P^k \end{bmatrix}$$

then $(d_x; d_\tau)$ can be computed as follows

$$d_\tau = \frac{\eta r_G^k + (\tau^k)^{-1}(-\tau^k\kappa^k + \gamma\mu^k e) + (g^k + (\xi^k)^T H^k; -b^T)p^k}{(\tau^k)^{-1}\kappa^k + (\xi^k)^T H^k \xi^k + (g^k + (\xi^k)^T H^k; -b^T)h^k}$$

and

$$(d_x; d_y) = p^k - h^k d_\tau.$$

Given $(d_x; d_\tau)$ then (d_s, d_κ) can easily be computed from relation (6.32). In the following lemma we will show that d_τ is always well defined.

Lemma 6.5.1 . *Let K be defined by (6.34) then*

$$K^{-1} = \begin{bmatrix} Q^{-1} - Q^{-1}A^T(AQ^{-1}A^T)^{-1}AQ^{-1} & Q^{-1}A^T(AQ^{-1}A^T)^{-1} \\ -(AQ^{-1}A^T)^{-1}AQ^{-1} & (AQ^{-1}A^T)^{-1} \end{bmatrix},$$

where $Q = H^k + (X^k)^{-1}S^k$. Moreover,

$$(\xi^k)^T H^k \xi^k + (g^k + (\xi^k)^T H^k; -b^T)h^k \geq 0.$$

Proof: It is easy to verify that $KK^{-1} = I$. Note that

$$Q^{-1} - Q^{-1}A^T(AQ^{-1}A^T)^{-1}AQ^{-1}$$

is symmetric and positive semi-definite. By definition

$$h^k = K^{-1}((g^k)^T - H^k\xi^k; -b).$$

In the rest of the proof we use the definition $B := (AQ^{-1}A^T)^{-1}$. Thus, we have per definition that

$$h^k = \begin{bmatrix} Q^{-1} - Q^{-1}A^TBAQ^{-1} & Q^{-1}A^TB \\ -BAQ^{-1} & B \end{bmatrix} \begin{bmatrix} (g^k)^T - H^k\xi^k \\ -b \end{bmatrix}.$$

Hence

$$\begin{array}{ll} & (g^k + (\xi^k)^TH^k; -b^T)h^k \\ = & (g^k + (\xi^k)^TH^k)(Q^{-1} - Q^{-1}A^TBAQ^{-1})((g^k)^T - H^k\xi^k) \\ & +b^TBAQ^{-1}((g^k)^T - H^k\xi^k) - (g^k + (\xi^k)^TH^k)Q^{-1}A^TBb + b^TBb \\ = & g^k(Q^{-1} - Q^{-1}A^TBAQ^{-1})(g^k)^T - (\xi^k)^TH^k(Q^{-1} - Q^{-1}A^TBAQ^{-1})H^k\xi^k \\ & +(b - AQ^{-1}H^k\xi^k)^TB(b - AQ^{-1}H^k\xi^k) - (\xi^k)^TH^kQ^{-1}A^TBAQ^{-1}H^k\xi^k \\ \geq & -(\xi^k)^TH^kQ^{-1}H^k\xi^k. \end{array}$$

Since H^k is positive semi-definite and $Q = H^k + (X^k)^{-1}S^k$,

$$(\xi^k)^TH^k\xi^k - (\xi^k)^TH^kQ^{-1}H^k\xi^k \geq 0,$$

which gives the desired result.

□

It should be observed that h^k needs to be computed only once in each iteration, even if (6.31) and (6.32) are solved for different right-hand-side vectors. Thus, the homogeneous algorithm needs exact one more solve per iteration than the "conventional" primal-dual algorithm with an interior feasible starting point.

The main work in computing the search direction lies in computing h^k and p^k. Both h^k and p^k are solutions to a linear system of the form

$$K(u_1; u_2) = \bar{K}(u_1; -u_2) = (v_1; v_2) \tag{6.35}$$

where

$$\bar{K} = \begin{bmatrix} H^k + (X^k)^{-1}S^k & A^T \\ A & 0 \end{bmatrix}.$$

$\bar{K}$ is a symmetric indefinite matrix and its inverse is given by

$$\bar{K}^{-1} = \begin{bmatrix} Q^{-1} - Q^{-1}A^T(AQ^{-1}A^T)AQ^{-1} & Q^{-1}A^T(AQ^{-1}A^T)^{-1} \\ (AQ^{-1}A^T)^{-1}AQ^{-1} & -(AQ^{-1}A^T)^{-1} \end{bmatrix}$$

where $Q = H^k + (X^k)^{-1}S^k$. Clearly Q is positive definite matrix, because c is convex.

If Cholesky factorizations of the positive definite matrices Q and $(AQ^{-1}A^T)$ are computed and known, then it is easy to solve system (6.35). Especially, if Q is a diagonal matrix, then the Cholesky factorization method might be desired. Indeed most interior-point codes for LP use this approach.

Another approach to solve (6.35) is to compute the factorization

$$\bar{K} = LDL^T \tag{6.36}$$

using the Bunch-Parlett strategy, where L is a lower triangular matrix and D is a non-singular block diagonal matrix. This approach has been pioneered for LP by Fourer and Mehrotra [51] and Turner [139]. The main advantage of this approach is that it has good numerical properties and it is not hampered by a few dense columns in A.

Vanderbei [141] proposes an alternative factorization based on the observation that the matrix $\bar{K}$ is quasi-definite. It follows that there exists a factorization of $\bar{K}$ such that D is a diagonal matrix. The main advantage of Vanderbei's approach is a sparsity preserving ordering can be chosen once and for all. This leads to a very efficient implementation. However, a factorization based on the quasi-definite property might not be as numerically stable as the more general Bunch-Parlett factorization.

In our preliminary implementation we use a factorization based on Vanderbei's ideas. The ordering is obtained with the minimum degree heuristic. In the future we plan to do some experiments with an iterative solver and the Bunch-Parlett factorization.

6.5.2 Choice of parameters and high-order

In each iteration we must choose the parameters η and γ related by $\gamma = 1 - \eta$. In Theorem 6.4.1 we have shown that the choice $\gamma = 1 - O(1/\sqrt{n})$ leads to polynomial complexity. Unfortunately this choice also leads to a very slow linear convergence. Therefore in our implementation we use a more aggressive approach. It is based on a heuristic originally proposed by Mehrotra [106] for solving LP problems. The idea is first to compute the affine scaling direction which is a solution to (6.31) and (6.32) for $\gamma = 0$ and $\eta = 1$. Let $(d_x^a; d_\tau^a; d_y^a; d_s^a)$ be the affine scaling direction then compute

$$\begin{array}{rl} \bar{\alpha}^a = \text{minimize} & ((x;\tau) + \alpha(d_x^a; d_\tau^a))^T((s^a, \kappa^a)^T + \alpha(d_s^a, d_\kappa^a)) \\ \text{subject to} & (x;\tau;s;\kappa) + \alpha(d_x^a; d_\tau^a, d_s, d_\kappa^a) \geq 0, \\ & \alpha \geq 0, \end{array} \tag{6.37}$$

and

$$r = \frac{((x;\tau) + \bar{\alpha}^a(d_x; d_\tau))^T((s,\kappa)^T + \bar{\alpha}^a(d_s, d_\kappa))}{(x;\tau)^T(s,\kappa)}. \tag{6.38}$$

Next we let

$$\gamma = \max(\min(r^2, r/10.0), 1.0e-6). \tag{6.39}$$

The idea is to reduce the barrier parameter γ fast whenever good progress can be made along the affine scaling direction.

Afterwards we recompute the search direction using this choice of γ and $\eta = 1 - \gamma$. The new point is given by

$$x^+ = x + \alpha d_x, \quad \tau^+ = \tau + \alpha d_\tau$$

and

$$y^+ = y + \alpha d_y, \quad s^+ = s + \alpha d_s \quad \kappa^+ = \kappa + \alpha d_\kappa.$$

Note that in our preliminary implementation we do not take different step size for the primal and dual variables. Moreover we do not use the non-linear update (6.26) of dual variables due to complication of finding a feasible step size. However we choose the step size α such that the new point satisfies

$$\min(x_j^+ s_j^+, \tau^+\kappa^+) \geq \beta_l \frac{(x^+;\tau^+)^T(s^+;\kappa^+)}{n+1} \tag{6.40}$$

where β_l is a constant. This prevents the iterates from prematurely to converge to the boundary (e.g., see Güler and Ye [70]).

It is common to choose the step size α such that the iterates remain in a certain neighborhood or a merit function is decreased to secure global convergence, see Kojima et.al. [84] and Anstreicher and Vial [18]. We have not used this safeguard yet.

It is well-known that high-order corrections can be used to improve the practical efficiency of the primal-dual barrier algorithm for LP significantly. Especially ideas along the lines of those proposed by Mehrotra [106] seem to work well. Also Hung and Ye [76] recently obtained good theoretical results for a high-order version of the homogeneous algorithm. In our preliminary implementation we have used a high-order algorithm similar to Mehrotra's. In our preliminary implementation we only do the correction on the complementarity equation.

6.5.3 Simple bounds

Many practical problems contains a lot of simple bounds of the type

$$l \leq x \leq u$$

Therefore we solve the problem

$$\begin{array}{rl} \min & c(x' + l) \\ \text{s.t.} & Ax' = b - Al \\ & x' + z = u - l \\ & x' \geq 0, \ z \geq 0, \end{array}$$

where z is an additional slack variable. We only introduce z variable for each finite upper bound, and our linear algebra handles the upper bounds explicitly. The function $c(x' + l)$ is seen as a new composite function.

6.5.4 Starting point and stopping criteria

Our starting point is

$$(x^0, \tau^0, y^0, s^0, \kappa^0) = (e, 1, 0, e, 1).$$

Define

$$pfeas_1^k = \frac{\|Ax^k - \tau^k b\|_1}{\tau^k + \|x^k\|_1} \quad \text{and} \quad dfeas_1^k = \frac{\|\tau^k \nabla c(x^k/\tau^k) - A^T y^k - s^k\|_1}{\tau^k + \|s^k\|_1}.$$

and

$$rgap^k = \frac{\nabla c(x^k/\tau^k)x^k - b^T y^k}{\tau^k + |\tau^k c(x^k/\tau^k) - \nabla c(x^k/\tau^k)x^k + b^T y^k|}.$$

$rgap^k$ is equal to absolute duality gap divided by the dual objective value. We terminate the algorithm in the case we have

$$pfeas_1^k \le tolpfeas_1, \quad dfeas_1^k \le toldfeas_1 \quad \text{and} \quad rgap^k \le tolrgap,$$

where $tolpfeas_1, toldfeas_1$ and $tolrgap$ are user-specified tolerances. In this case we say the solution is feasible and we report the solution

$$(x', y', s') = (x^k/\tau^k, y^k/\tau^k, s^k/\tau^k).$$

We also terminate the algorithm if

$$\frac{\mu^k}{\mu^0} \le tolured.$$

If

$$\frac{\tau^k \kappa^0}{\kappa^k \tau^0} \le tolinf.$$

we report the problem is infeasible. $tolured$ and $tolinf$ are also user-specified tolerances. Our termination criteria are a slight generalization of those proposed by Xu et al. [148].

6.6 Computational results

In this section we will report our computational results to demonstrate effectiveness of the homogeneous algorithm for linearly constraint convex programming problems. All our code is written in ANSI C. The test is performed on a HP 715/75 UNIX workstation. All timing results are obtained using the C procedure "clock". We use the tolerances $tolpfeas_1 = toldfeas_1 = 1.0e-6$, $tolrgap = 1.0e-8$, $tolured = 1.0e-14$ and $tolinf = 1.0e-12$.

In the first part of the test we use quadratic problems as test problems. The test problem is formed by adding a quadratic term $.5x^T Qx$ to the objective function of some of the NETLIB problems. The Q matrix has the form

$$\begin{bmatrix} 2 & -1 & & \\ -1 & 2 & & \\ & & 2 & -1 \\ & & -1 & 2 \end{bmatrix}.$$

Q is clearly positive definite because it is strictly diagonal dominant. Moreover, the Q matrix is block diagonal. We select the feasible problems 25FV47, DEGEN3, GREENBEA, PILOT and WOOD1P and two (primal) infeasible problems KLEIN2 and VOL1 from the NETLIB LP problems.

In addition we use some box-constrained QP problems arising from the so-called obstacle and elastic-plastic torsion problems. These problems are taken from the paper Han et al. [71]. The name of these problems all start with the letter B. The B2 problems are slightly modified. The original problems contain some very large upper bounds. These bounds are always redundant and therefore we have removed them in B2, which leads to a reduction in the number of iterations by approximately 50%. Our results are presented in Table 6.1.

The columns in Table 6.1 shows the number constraints, the number variables (including slack variables), the solution status, the number of iterations, the solution time in seconds, $rgap$, $pfeas_1$ and $dfeas_1$.

TITLE	ROWS	COLS	STATUS	ITER	TIME	RGAP	PFEAS1	DFEAS1
25FV47	821	1876	OPTIMAL	18	14.8	4.9E-10	2.0E-13	1.0E-15
DEGEN3	1503	2604	OPTIMAL	14	86.1	8.7E-09	1.0E-09	1.0E-10
GREENBEA	2392	5495	OPTIMAL	27	49.0	6.1E-09	1.0E-09	7.0E-13
KLEIN2	477	531	INFEASIBLE	21	21.8	1.0E+00	2.0E-02	6.0E-15
B1M100T1	0	10000	OPTIMAL	10	19.4	2.4E-10	4.0E-16	2.0E-13
B1M200T1	0	40000	OPTIMAL	10	107.9	9.1E-10	4.0E-16	5.0E-13
B1M300T1	0	90000	OPTIMAL	11	346.4	3.3E-10	7.0E-16	2.0E-12
B2M100T1	0	10000	OPTIMAL	13	20.9	5.9E-09	0.0E+00	5.0E-14
B2M200T1	0	40000	OPTIMAL	12	113.5	8.6E-13	0.0E+00	6.0E-13
B2M300T1	0	90000	OPTIMAL	10	287.8	9.9E-10	0.0E+00	2.0E-14
B3M100T20	0	10000	OPTIMAL	11	21.9	7.7E-11	6.0E-16	1.0E-11
B3M200T20	0	40000	OPTIMAL	10	108.0	4.2E-09	2.0E-16	3.0E-09
B3M300T20	0	90000	OPTIMAL	14	440.1	2.6E-09	8.0E-17	5.0E-09
PILOT	1441	4693	OPTIMAL	33	254.9	9.3E-09	2.0E-08	4.0E-11
VOL1	323	459	INFEASIBLE	29	2.6	1.0E+00	3.0E-01	5.0E-12
WOOD1P	244	2595	OPTIMAL	22	58.0	2.9E-10	9.0E-10	2.0E-15

Tabel 6.1: Quadratic test problems.

We see that all of the feasible problems are solved to the required precision and two infeasible problems are correctly detected. As expected, the algorithm uses very few iterations to solve these problems. In summary our algorithm seems to work well for solving quadratic problems.

In the second part of the test we solve some entropy problems. These problems is formed by adding an entropy term, $\sum x_j \ln(x_j)$, to the objective function of some of the NETLIB LP problems: 25fV47, BNL1, FIT1D, KLEIN2, MONDOU2, SCSD8, SHI12L and WOOD1P. Except for problems KLEIN2 and MONDOU2, these problems are feasible, but many of them have no interior similar to the following example

$$\begin{array}{ll} \text{minimize} & x\ln(x) \\ \text{subject to} & x = 0, \\ & x \geq 0. \end{array}$$

Its dual problem is

$$\begin{array}{ll} \text{maximize} & -x \\ \text{subject to} & 1 + \ln(x) - y - s = 0, \\ & x, s \geq 0. \end{array}$$

In any feasible primal-dual pair the dual solution y must be equal to $-\infty$. Therefore if the entropy problem has an optimal solution with some of the primal variables equal to zero, unbounded dual solutions and numerical difficulties are expected.

In solving these entropy minimization problems we have used the non-linear update

$$s^+ = \nabla f(x + \alpha d_x) - A^T(y + \alpha d_y) + (1 - \alpha\eta) r_D. \tag{6.41}$$

of the dual slacks proposed in Section 6.4, see (6.26). If this update is used the primal and dual infeasibility is reduced at the same rate. We have implemented the update as follows. First the maximum step-size $\bar{\alpha}$ is determined subject to $(x + \bar{\alpha} d_x, \tau + \bar{\alpha} d_\tau, s + \bar{\alpha} d_s, \kappa + \bar{\alpha} d_\kappa) \geq 0$. Next the first t in $\{1, 2, 4, 16, \ldots\}$ is chosen with $\alpha = 0.9^t \bar{\alpha}$ such that s^+ in (6.41) is positive and (6.40) is satisfied. We also use a less aggressive choice of γ, see (6.39). γ is computed using $\gamma = \min(0.9, \max(r^2, 1.0e{-}6))$ where r is defined by (6.38). The step-size $\bar{\alpha}^a$ used in (6.38) is a approximation for maximum step-size to boundary using the non-linear update of the dual slacks. In Table 6.2 we have reported the results for solving these problems. We see from Table 6.2 that a good accuracy is achieved and all the problems are solved satisfactorily. As expected the iterations count is worse especially for the problem BNL1. The iteration count is increased because for these nonlinear problems we must follow the central path closely. We expect that a more sophisticated line search in the dual update will make the implementation more efficient.

TITLE	ROWS	COLS	STATUS	ITER	TIME	RGAP	PFEAS1	DFEAS1
25FV47	821	1876	OPTIMAL	32	15.0	4.2E-09	5.0E-09	7.0E-10
BNL1	643	1586	OPTIMAL	98	22.5	3.4E-09	4.0E-10	1.0E-13
FIT1D	24	1049	OPTIMAL	21	4.0	3.2E-09	4.0E-09	3.0E-10
KLEIN2	477	531	INFEASIBLE	28	25.6	1.0E+00	3.0E-02	1.0E-16
MONDOU2	312	604	INFEASIBLE	13	0.9	1.0E+00	7.0E+05	6.0E-15
SCSD8	397	2750	OPTIMAL	14	4.5	1.5E-11	1.0E-15	9.0E-15
SHIP12L	1151	5533	OPTIMAL	39	28.5	6.0E-09	9.0E-08	2.0E-09
WOOD1P	244	2595	OPTIMAL	43	49.7	2.9E-08	2.0E-07	1.0E-17

Tabel 6.2: Results for the entropy problems.

6.7 Conclusion

In this paper we have presented a generalization of the homogeneous model for LP to solving the monotone complementarity problem. The advantages of the model are: 1) it can detect the infeasible status by generating a certificate; 2) it can start from any point in the positive orthant; 3) if the problem is polynomially solvable with an interior feasible starting point, then the problem is polynomially solvable, either find a solution or prove infeasibility, without using or knowing such information at all.

We have specialized the proposed algorithm to solving the linearly constraint convex program, and implemented the proposed algorithm on solving some test convex programming problems. Our preliminary computational results are promising, although some issues, such as to handle unbounded asymptotical optimal solutions, need some further research work.

Kapitel 7

Conclusion

In this chapter, we first summarize the dissertation and its contributions. Next, we propose some subjects for further research.

7.1 Summary

Chapter 2 serves as an introduction to the dissertation. It reviews the development in the area interior-point methods since Karmarkar's 1984 paper.

The chapters 3 and 4 present some methods for automatic size reductions of LP problems. The main contribution of Chapter 3 is a comprehensive survey and unification of existing methods. Moreover, for the first time in the public literature, we give a comprehensive discussion of the postsolve phase. Our ideas are an improvement of the ideas outlined in the paper by Tomlin and Welch [135]. Finally, our computational results document that it is worthwhile to use automatic reduction methods in combination with an interior-point method.

In Chapter 4, we discuss how to remove all linearly dependent rows from large-scale LP problems. Our main contribution is the proposal of some simple heuristics which reduce the computational costs of finding all the linearly dependent rows significantly.

Chapter 5 discusses how to combine interior-point and pivoting algorithms. The issue of this chapter is how to obtain an optimal solution to an LP problem which is also a basic solution. This subject is very important for the usefulness of interior-point methods in many practical applications. Megiddo [104] has shown that if an exact primal and dual optimal solution is known, then a basic solution can be generated in strongly polynomial time. In practice, interior-point algorithms do not generate exact optimal solutions, but only a sequence converging towards the optimal solution. Therefore, our contribution is to show that, by exploiting properties of the interior-point methods, an exact optimal solution is not necessary. Moreover, we present an alternative interpretation of Megiddo's method leading to a computational convenient method. Finally, our numerical results show that a basic solution can be obtained in less than 15% of the total solution time and for most problems much faster.

Finally, in Chapter 6, we present a generalization of the homogeneous and self-dual algorithm for LP problems to the monotone complementarity problems. This class of problems contain all convex programming problems. The algorithm does not need to use any "big-M" parameter or two-phase method, and it generates either a solution converging towards feasibility and complementarity simultaneously or a certificate proving infeasibility. Moreover, if the problem is polynomially solvable with an interior feasible starting point, then it can be polynomially solved without using or knowing such information at all. No other algorithm has the same properties.

7.2 Future research areas

Several possibilities for extension of the work presented in this dissertation exist.

The automatic reduction methods, discussed in Chapter 3 and Chapter 4, are aimed at LP problems. Most of these methods can be generalized to general nonlinear problems. Indeed, the computational payoff from reducing a nonlinear problem might be significantly larger than reducing an LP problem.

The purification methods, presented in Chapter 5 cannot be improved a lot from a theoretical point view. However, the practical efficiency of the methods can be improved for classes of LP problems. Indeed, in a forthcoming paper we discuss this possibility.

Chapter 6 presents a generalization of the homogeneous and self-dual algorithm for LP to the monotone complementarity problem. This algorithm has nice theoretical properties and the presented numerical results are good. However, the developed theory assumes that all variables are bounded from below. This is not always the case in practice, and one possible extension is to develop a variant which handles free variables directly. Moreover, much more practical experience with the algorithm is needed to find the most efficient variant. For example, we propose two updates of the dual variables. One question is, which one of them should be used. Another possibility is to use a combination of the two updates.

I believe that this method (and interior-point methods) can be used to solve significantly larger convex optimization problems than is possible with today's active set methods (such as MINOS). In practice, this implies, when solving problems in operations research, economics, and finance that better decisions can be made.

Kapitel 8

Danish summary

I dette afsnit opsummerer jeg på dansk indholdet af den foreliggende afhandling. Selve afhandlingen består af fem tildels uafhængige artikler, der har emnet indre-punkts metoder til fælles. Disse artikler er indeholdt i hver deres kapitel.

Det første artikel (Kapitel 2) er en gennemgang af de sidste 10 års udvikling inden for området indre-punkts metoder. Formålet med artiklen er at give læseren en ramme til forståelse af de efterfølgende fire artikler.

Det andent artikel (Kapitel 3) diskuterer automatiske reduktionsmetoder til LP problemer. Det vil sige, at der foreslås nogle metoder, som kan bruges til at reducere størrelsen på et LP problem, førend det løses med en optimeringsalgoritme. Artiklen præsenterer først en gennemgang af de eksisterende metoder. Dernæst diskuteres det, hvordan den optimale løsning til det originale problem rekonstrueres ud fra løsningen til det reducerede problem. Endelig præsenteres numeriske resultater til at dokumentere fordelagtigheden af de foreslåede metoder.

Det tredje artikel (Kapitel 4) diskuterer, hvordan alle linært afhængige rækker fjernes fra et LP problem. Dette er et vigtigt, fordi i indre-punkts metoder skal der i hver iteration løses et linært ligningssystem, som ikke har en unik løsning i tilfælde af linær afhængige rækker. I artiklen foreslås nogle simple heuristiker, som gør det meget hurtigt at finde de linear afhængige rækker i praksis. Sidstnævnte påstand er dokumenteret med numeriske resultater.

Det fjerde artikel (Kapitel 5) præsenterer en metode til at kombinere en pivoterings og en indre-punkts algoritme. Dette er et vigtigt emne, fordi i mange praktiske applikationer af LP er det vigtigt, at den optimale løsning også er en basis løsning. En indre-punkts algoritme genererer normalt ikke en løsning, som også er en basisløsning med mindre den optimale løsning er unik. Megiddo [104] har vist, at hvis man har en eksakt primal og dual optimal løsning, så kan en optimal basis løsning findes i stærk polynomiel tid. I artiklen vises det, at en eksakt optimal løsning ikke er nødvendig. Det resultat opnås ved at udnytte konvergensegenskaber ved indre-punkts algoritmerne. Desuden gives en alternativ fortolkning af Megiddos algoritme, som letter implementeringen af den. Endelig præsenteres numeriske resultater, som viser, at den foreslåede algoritme er meget effektiv.

I det femte artikel (Kapitel 6) generaliseres den selv-duale og "homogeneous" LP algoritme til det monotone komplementaritets problem. Dette problem indeholder klassen af alle konvekse optimeringsproblemer. Den foreslåede algoritme løser problemet i en fase uden brug af nogen "big-M" parameter. Desuden kan algoritmen opdage, at problemet ingen løsning har, hvis det er tilfældet. Endelig er algoritmen polynomiel, hvis en "smoothness" antagelse er opfyldt. Så vidt vides har ingen anden algoritme tilsvarende egenskaber.

Bibliografi

[1] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga. An implementation of Karmarkar's algorithm for linear programming. *Math. Programming*, 44:297–235, 1989.

[2] I. Adler, N. Karmarkar, M. G. C. Resende, and G. Veiga. Data structures and programming techniques for the implementation of Karmarkar's algorithm. *ORSA J. on Comput.*, 1(2):84–106, 1989.

[3] I. Adler and R.D.C. Monteiro. A geometric view of parametric linear programming, 1992.

[4] F. Alizadeh. Optimization over the positive definite cone: Interior-point methods and combinatorial applications. In P. M. Pardalos, editor, *Advances in optimization and parallel computing*, pages 1–25. North Holland, 1992.

[5] R. Anbil, R. Tanga, and E.L. Johnson. A global approach to crew-pairing optimization. *IBM Systems J.*, 31(1):71–78, 1992.

[6] E. D. Andersen. Finding all linearly dependent rows in large-scale linear programming. *Optimization Methods and Software*, 6:219–227, 1995.

[7] E. D. Andersen and K. D. Andersen. Presolving in linear programming. *Math. Programming*, 71(2):221–245, 1995.

[8] E. D. Andersen, K. D. Andersen, and X. Xu. Improved Cholesky decompositions for work-stations and vector computers. Technical report, 1994. In preparation.

[9] E. D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior-point methods of mathematical programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[10] E. D. Andersen and Y. Ye. On a homogeneous algorithm for the monotone complementarity problem. Technical report, Department of Management Sciences, The University of Iowa, 1995. Revised November 1996, to appear in Mathematical Programming.

[11] E. D. Andersen and Y. Ye. Combining interior-point and pivoting algorithms. *Management Sci.*, 42(12):1719–1731, December 1996.

[12] K. D. Andersen. A large scale implementation of a dual affine scaling algorithm for linear programming. Master's thesis, Dept. of Math. and Computer Sci., Odense University, 1991.

[13] K. D. Andersen. An efficient Newton barrier method for Minimizing a Sum of Euclidean Norms. Preprint 42, Dept. of Math. and Computer Sci., Odense University, 1993. submitted to SIAM J. on Optim.

[14] K. D. Andersen. An infeasible dual affine scaling method for linear programming. *COAL Bulletin*, 22:19–27, 1993.

[15] K. D. Andersen. A modified Schur complement method for handling dense columns in interior point methods for linear programming. Technical report, Dept. of Math. and Computer Sci., Odense University, 1994. Submitted to ACM Transaction on Mathematical Software.

[16] K. D. Andersen and E. Christiansen. A symmetric primal-dual Newton method for minimizing a sum of norms. Technical report, Dept. of Math. and Computer Sci., Odense University, 1995. Submitted to Math. Programming.

[17] K. D. Andersen, E. Christiansen, and M.L. Overton. Computing limit loads by minimizing a sum of norms. Preprint 30, Dept. of Math. and Computer Sci., Odense University, 1994. Submitted to SIAM Journal on Scientific Computing.

[18] K. Anstreicher and J. -P. Vial. On the convergence of an infeasible primal-dual interior-point method for convex programming. Technical report, 1993. to appear in Optimization Methods and Software.

[19] E.R. Barnes. A variation on Karmarkar's algorithm for solving linear programming problems. *Math. Programming*, 36:174–182, 1986.

[20] R. E. Bixby. Implementing the simplex method: The initial basis. *ORSA J. on Comput.*, 4(3):267–284, 1992.

[21] R. E. Bixby. Progress in linear programming. *ORSA J. on Comput.*, 6(1):15–22, 1994.

[22] R. E. Bixby, J. W. Gregory, I. J. Lustig, R. E. Marsten, and D. F. Shanno. Very large-scale linear programming: A case study in combining interior point and simplex methods. *Oper. Res.*, 40(5):885–897, 1992.

[23] R. E. Bixby and M. J. Saltzman. Recovering an optimal basis from an interior point solution. *Oper. Res. Lett.*, 15(4):169–178, 1993.

[24] P. T. Boggs, P. D. Domich, J. R. Donaldson, and C. Witzgall. Algorithmic enhancements to the method of centers for linear programming problems. *ORSA J. on Comput.*, 1(3), 1989.

[25] G. H. Bradley, G. G. Brown, and G. W. Graves. Structurel redundancy in large-scale optimization models. In M. H. Karwan et al., editor, *Redundancy in mathematical programming*, pages 145–169. Springer Verlag, Berlin, 1983.

[26] A. L. Brearley, G. Mitra, and H. P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Math. Programming*, 15:54–83, 1975.

[27] M. G. Breitfeld and D. F. Shanno. Computational experience with modified log-barrier functions for large-scale nonlinear programming. Technical Report RRR 17-93, RUTCOR, Rutgers Center for Operations Research, P.O. Box 5062, New Brunswick, New Jersey, October 1993.

[28] M. G. Breitfeld and D. F. Shanno. Preliminary computational experience with modified log-barrier functions for large-scale nonlinear programming. In W. W. Hager, D. W. Hearn, and P. M. Pardalos, editors, *Large scale optimization: Sate of the art*, pages 45–66. Kluwer Academic Publishers, London, 1994.

[29] W.J. Carolan, J.E. Hill, J.L. Kennington, S. Niemi, and S.J. Wichman. An empirical evaluation of the KORBX algorithms for military airlift applications. *Oper. Res.*, 38(2):240–248, 1990.

[30] T. J. Carpenter, I. J. Lustig, J. M. Mulvey, and D. F. Shanno. Higher order predictor-corrector interior point methods with application to quadratic objectives. *SIAM J. on Optim.*, 3:696–725, 1993.

[31] T. J. Carpenter, I. J. Lustig, J. M. Mulvey, and D. F. Shanno. Separable quadratic programming via a primal-dual interior point method and its use in a sequential procedure. *ORSA J. on Comput.*, 5:182–191, 1993.

[32] S. F. Chang and S. T. McCormick. A hierachical algorithm for making sparse matrices sparse. *Math. Programming*, 56:1–30, 1992.

[33] A. Charnes and K. O. Kortanek. An opposite sign algorithm for purification to an extreme point solution, 1963. Memorandum No. 89. Office of Naval Research.

[34] E. Christiansen and K. O. Kortanek. Computation of the collapse state in limit analysis using the LP primal affine scaling algorithm. *J. Comput. Appl. Math.*, 34:47–63, 1991.

[35] V. Chvátal. *Linear programming*. W.H. Freeman and Company, 1983.

[36] R. W. Cottle, J. -S. Pang, and R. E. Stone. *The Linear Complementarity Problem*. Computer Science and Scientific Computing, Academic Press, San Diego, 1992.

[37] G. B. Dantzig. Reminiscenes about the origins of linear programming. *Oper. Res. Lett.*, 1(2):42–48, 1992.

[38] D. den Hertog. *Interior point approach to linear, quadratic and convex programming: Algorithms and complexity*. PhD thesis, Faculty of Mathematics and Informatics, Delft University of Technology, 1992.

[39] D. den Hertog, F. Jarre, C. Roos, and T. Terlaky. A sufficient conditions for self-concordance, with application to some classes of structured convex programming problems. *Math. Programming*, 69(1):75–88, 1995.

[40] D. den Hertog, J. Kaliski, C. Roos, and T. Terlaky. A logarithmic barrier cutting plane method for convex programming. Technical Report DUT-TWI-93-43, Delft University of Technology, Delft, The Netherlands, 1993.

[41] I. I. Dikin. Iterative solution of problems of linear and quadratic programming. *Soviet Math. Dokl.*, 8:674–675, 1967.

[42] I. I. Dikin. About the convergence of an iterative process. *Upravlayemye Sistemi*, 12:54–60, 1974.

[43] P. D. Domich, P. T. Boggs, J. E. Rogers, and C. Witzgall. Optimizing over three dimensional subspaces in an interior-point method for linear programming. *Linear Algebra Appl.*, 152:315–342, 1991.

[44] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. Oxford University Press, New York, 1989.

[45] I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner. The factorization of sparse symmetric indefinite matrices. *IMA J. Numer. Anal.*, 11:181–204, 1991.

[46] A. S. El-Bakry, R. A. Tapia, T. Tsuchiya, and Y. Zhang. On the formulation and theory of the primal-dual Newton interior-point method for nonlinear programming. Technical Report TR92-40, Department of Computational & Applied Mathematics, Rice University, December 1992. Revised October 1993.

[47] A. S. El-Bakry, R. A. Tapia, and Y. Zhang. A study of indicators for identifying zero variables in interior-point methods. *SIAM Rev.*, 36(1):45–72, 1994.

[48] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley and Sons, New York, 1968.

[49] J. J. H. Forrest and J. A. Tomlin. Implementing the simplex method for optimization subroutine library. *IBM Systems J.*, 31(1):11–25, 1992.

[50] R. Fourer and S. Mehrotra. Performance of an augumented system approach for solving least squares problems in an interior-point method for linear programming. *COAL Newsletter*, 19:26–31, 1991.

[51] R. Fourer and S. Mehrotra. Solving symmetric indefinite systems in an interior point method for linear programming. *Math. Programming*, 62:15–40, 1993.

[52] R. W. Freund and F. Jarre. A QMR-based interior-point algorithm for solving linear programs. Technical report, AT&T, Bell Laboratories, December 1994. Numerical Analysis Manuscript 94-19.

[53] R. W. Freund and F. Jarre. An interior-point method for fractional programs with convex constraints. *Math. Programming*, 67(3):407–440, 1994.

[54] K. R. Frisch. The logarithmic potential method of convex programming. Technical report, University Institute of Economics, Oslo, Norway, 1955.

[55] G. W. Graves and R. D. McBride and I. Gershkoff and D. Anderson and D. Mahidhara. Flight crew scheduling. *Management Sci.*, 39(6):736–745, 1993.

[56] D. M. Gay. Electronic mail distribution of linear programming test problems. *COAL Newsletter*, 13:10–12, 1985.

[57] I. Gershkoff. Optimizing flight crew schedules. *Interfaces*, 19(4):29–43, 1989.

[58] P. E. Gill, W. Murray, and M. A. Saunders. A single-phase dual barrier method for linear programming. Technical Report SOL 88-10, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Standford, California 94305-4022, USA, August 1988.

[59] P. E. Gill, W. Murray, M. A. Saunders, J. A. Tomlin, and M. H. Wright. On the projected Newton barrier methods for linear programming and an equivalence to Karmarkar's projective method. *Math. Programming*, 36:183–209, 1986.

[60] P.E. Gill, W. Murray, D.B. Ponceleón, and M.A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Mat. Anal. Appl.*, 13:292 – 311, 1992.

[61] D. Goldfarb and M. J. Todd. Linear programming. In G. L. Nemhauser, A. H. G. Rooney Kan, and M. J. Todd, editors, *Optimization*, pages 73–170. North Holland, New York and Amsterdam, 1989.

[62] A. J. Goldman and A. W. Tucker. Theory of linear programming. In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and related Systems*, pages 53–97, Princeton, New Jersey, 1956. Princeton University Press.

[63] J. Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. Technical Report 1994.20, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, November 1994. Revised May 1995, to appear in Computational Optimization and Applications.

[64] J. Gondzio. Presolve analysis of linear programs prior to applying the interior point method. Technical Report 1994.3, Logilab, HEC Geneva, Section of Management Studies, University of Geneva, 1994. Revised Dec. 1994, to appear in ORSA J. on Comp.

[65] J. Gondzio and T. Terlaky. A computational view of interior point methods for linear programming. In J. Beasley, editor, *Advances in linear and integer programming*. Oxford University Press, Oxford, England, 1995.

[66] C. C. Gonzaga. An algorithm for solving linear programming programs in $O(n^3L)$ operations. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 1–28. Springer Verlag, 1989.

[67] C. C. Gonzaga. Path-following methods for linear programming. *SIAM Rev.*, 34(2):167–224, June 1992.

[68] H. J. Greenberg. The use of the optimal partition in a linear programming solution for postoptimal analysis. *Oper. Res. Lett.*, 15(4):179–186, 1994.

[69] O. Güler. Existence of interior points and interior paths in nonlinear monotone complementarity problems. *Math. Oper. Res.*, 18:148–162, 1993.

[70] O. Güler and Y. Ye. Convergence behaviour of interior-point algorithms. *Math. Programming*, 60(2):215–228, 1993.

[71] C.-G. Han, P. M. Pardalos, and Y. Ye. Computational aspects of an interior point algorithm for quadratic programming problems with box constraints. In T. F. Coleman and Y. Li, editors, *Large-Scale Numerical Optimization*. SIAM, 1990.

[72] P.M.J. Harris. Pivot selection methods in the Devex LP code. *Math. Programming Study*, 4:30–57, 1975.

[73] Nicholas J. Highham. Analysis of the Cholesky decomposition of a semi-definit matrix. In M.G. Cox and S. Hammarling, editors, *Reliable Numerical Computation*, chapter 9, pages 161–185. Clarendon Press, Oxford, 1990.

[74] K. L. Hoffman and M. Padberg. Solving airline crew scheduling problems by branch-and-cut. *Management Sci.*, 39(6):657–682, 1993.

[75] P. Huard. Resolution of mathematical programming with nonlinear constraints by method of centers. In J. Abadie, editor, *Nonlinear programming*. North Holland, Amsterdam, 1967.

[76] P. -F. Hung and Y. Ye. An asymptotical $O(\sqrt{n}L)$-iteration path-following linear programming algorithm that uses wide neighborhoods. Technical report, Department of Mathematics, The University of Iowa, March 1994. To appear in SIAM J. on Optimization.

[77] B. Jansen, C. Roos, and T. Terlaky. An interior point approach to postoptimal and parametric analysis in linear programming. In *Interior point methods*. Eövös Loránd University, Department of Operations Research, H-1088 Budapest, Múzeum krt. 6-8., Hungary, 1992.

[78] B. Jansen, T. Terlaky, and C. Roos. The theory of linear programming: Skew symmetric self-dual problems and the central path. *Optimization*, 29:225–233, 1994.

[79] F. Jarre. Interior-point methods for convex programming. *Applied Mathematics and Optimization*, 26:287–311, 1992.

[80] F. Jarre and M. A. Saunders. A practical interior-point method for convex programming. *SIAM J. on Optim.*, 5(1):149–171, February 1995.

[81] H. -W. Jung, R. E. Marsten, and M. J. Saltzman. Numerical factorization methods for interior point algorithms. *ORSA J. on Comput.*, 6(1):94–105, 1994.

[82] N. Karmarkar. A polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[83] L. G. Khachian. A polynomial algorithm in linear programming. *Soviet Math. Dokl.*, 20:191–194, 1979.

[84] M. Kojima, N. Megiddo, and S. Mizuno. A primal-dual infeasible-interior-point algorithm for linear programming. *Math. Programming*, 61:263–280, 1993.

[85] M. Kojima, N. Megiddo, and S. Mizuno. Theoretical convergence of large-step primal-dual interior-point algorithms for linear programming. *Math. Programming*, 59(1), 1993.

[86] M. Kojima, S. Mizuno, and A. Yoshise. A polynomial-time algorithm for a class of linear complementarity problems. *Math. Programming*, 44:1–26, 1989.

[87] M. Kojima, S. Mizuno, and A. Yoshise. A primal-dual interior point algorithm for linear proramming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 29–47. Springer Verlag, Berlin, 1989.

[88] M. Kojima, S. Mizuno, and A. Yoshise. A convex property of monotone complementarity problems. Technical Report B-267, Department of Information Sciences, Tokyo Institute of Technology, Tokyo, Japan, 1993.

[89] K. O. Kortanek, X. Xu, and Y. Ye. An interior-point algorithm for solving primal and dual geometric programs. Technical report, Department of Management Sciences, The University of Iowa, 1994. to appear in Math. Programming.

[90] K. O. Kortanek and J. Zhu. New purification algorithms for linear programming. *Naval Res. Logist. Quart.*, 35:571–583, 1988.

[91] Scicon LTD. *Sciconic user guide v2.00*, 1990.

[92] I. J. Lustig. Feasibility issues in primal-dual interior-point methods for linear programming. *Math. Programming*, 49:145–162, 1990.

[93] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a primal-dual interior point method for linear programming. *Linear Algebra Appl.*, 20:191–222, 1991.

[94] I. J. Lustig, R. E. Marsten, and D. F. Shanno. The interaction of algorithms and architectures for interior point methods. In P. M. Pardalos, editor, *Advances in optimization and parallel computing*, pages 190–205. Elsevier Sciences Publishers B.V., 1992.

[95] I. J. Lustig, R. E. Marsten, and D. F. Shanno. On implementing Mehrotra's predictor-corrector interior-point method for linear programming. *SIAM J. on Optim.*, 2(3):435–449, 1992.

[96] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Computational experience with a globally convergent primal-dual predictor-corrector algorithm for linear programming. *Math. Programming*, 66(1):123–135, 1994.

[97] I. J. Lustig, R. E. Marsten, and D. F. Shanno. Interior point methods for linear programming: Computational state of the art. *ORSA J. on Comput.*, 6(1):1–15, 1994.

[98] R. Marsten, M. Saltzman, R. Subramanian, I. Lustig, and D. Shanno. Interior point methods for linear programming: Just call Newton , Lagrange and Fiacco and McCormick ! *Interfaces*, 20(4):105–116, 1990.

[99] R. E. Marsten, M. J. Saltzman, D. F. Shanno, G. S. Pierce, and J. F. Ballintijn. Implementation of a dual affine interior point algorithm for linear programming. *ORSA J. on Comput.*, 1(4):287–297, 1989.

[100] G. P. McCormick. Resolving the Shell dual with a nonlinear primal-dual algorithm. In P. M. Pardalos, editor, *Advances in optimization and parallel computing.* Elsevier Sciences Publishers B.V., 1992.

[101] K. A. McShane. Superlinearly convergent $O(\sqrt{n}L)$-iteration interior-point algorithms for linear programming and the monotone linear complementarity problem. *SIAM J. on Optim.*, 4(2), May 1994.

[102] K. A. McShane, C. L. Monma, and D. F. Shanno. An implementation of a primal-dual method for linear programming. *ORSA J. on Comput.*, 1(2):70–83, 1989.

[103] N. Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 131–158. Springer Verlag, 1989.

[104] N. Megiddo. On finding primal- and dual- optimal bases. *ORSA J. on Comput.*, 3(1):63–65, 1991.

[105] S. Mehrotra. High order methods and their performance. Technical Report 90-16R1, Department of Industrial Engineering and Managment Sciences, Northwestern University, Evanston, USA., 1991.

[106] S. Mehrotra. On the implementation of a primal-dual interior point method. *SIAM J. on Optim.*, 2(4):575–601, 1992.

[107] S. Mehrotra. Quadratic convergence in primal-dual methods. *Math. Oper. Res.*, 18(3):741–751, 1993.

[108] S. Mehrotra and Y. Ye. Finding an interior point in the optimal face of linear programs. *Math. Programming*, 62(3):497–515, 1993.

[109] S. Mizuno, F. Jarre, and J. Stoer. A unified approach to infeasible-interior-point algorithms via geometrical linear complementarity problems. Technical Report Preprint 213, Department of Mathematics, Wurzburg University, Wurzburg, Germany, 1994.

[110] S. Mizuno, M.J. Todd, and Y. Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Math. Oper. Res.*, 18:964–981, 1994.

[111] S. Mizuno, M.J. Todd, and Y. Ye. A surface of analytic centers and primal-dual infeasible-interior-point algorithms for linear programming. *Math. Oper. Res.*, 20(1):135–162, 1995.

[112] C. L. Monma and A. J. Morton. Computational experience with a dual affine variant of Karmarkar's method for linear programming. *Oper. Res. Lett.*, 6(6), 1987.

[113] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Math. Programming*, 44:27–41, 1989.

[114] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part II: Convex quadratic programming. *Math. Programming*, 44:43–66, 1989.

[115] R. D. C. Monteiro and S. Wright. Local convergence of interior-point algorithms for degenerate monotone LCP. Technical Report MCS-P357-0493, Mathematics and Computer Science Division, Argonne National Laboratory, Chicago, IL, 1993.

[116] J.L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.

[117] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. SIAM, Philadelphia, PE, 1 edition, 1994.

[118] Y. E. Nesterov and A. S. Nemirovskii. An interior-point method for generalized linear-fractional programming. *Math. Programming*, 69(1), 1995.

[119] F. A. Potra. A quadratically convergent predictor-corrector method for solving linear prgrams from infeasible starting points. *Math. Programming*, 67(3), 1994.

[120] F. A. Potra and Y. Ye. Interior point methods for nonlinear complementarity problem. Technical Report TR 15, Department of Mathematics, The University of Iowa, Iowa City, IA, 1991.

[121] M. J. D. Powell. On the number of iterations of Karmarkar's algorithm for linear programming. *Math. Programming*, 62(1):153–197, 1993.

[122] J. Renegar. A polynomial-time algorithm, based on Newton's method, for linear programming. *Math. Programming*, 40:59–73, 1988.

[123] M.G.C. Resende, K.G. Ramakrishnan, and Z. Drezner. Computing lower bounds for the quadratic assignment problem with an interior point algorithm for linear programming. Technical report, AT&T Bell Laboratores, Murray Hill, NJ, USA, 1994.

[124] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, New Jersey, 1970.

[125] C. Roos and J. P. Vial. A polynomial approximate centers for linear programming. *Math. Programming*, 54:295–305, 1992.

[126] M.A. Saunders. Major Cholesky would feel proud. *ORSA J. on Comput.*, 6(1):94–105, 1994.

[127] D. F. Shanno and A. Bagchi. A unified view of interior point methods for linear programming. *Annals of Operations Research*, 22, 1990.

[128] G. Sonnevend. *An analytical centre for polyhedrons and new classes of global algorithms for linear (smooth, convex) programming*, pages 866–876. Springer Verlag, New York, 1985.

[129] R. Subramanian, R. P. S. Scheff Jr., J. D. Qillinan, D. S. Wiper, and R. E. Marsten. Coldstart: Fleet assigment at Delta Air Lines . *Interfaces*, 24(1), 1994.

[130] U. H. Suhl. MPOS - Mathematical optimization system. *European J. Oper. Res.*, 72(2):312–322, 1994.

[131] U. H. Suhl and L. M. Suhl. Computing sparse LU factorizations for large-scale linear programming bases. *ORSA J. on Comput.*, 2(4):325–335, 1990.

[132] M. J. Todd and Y. Ye. A lower bound on the number of iterations of long-step primal-dual algorithm linear programming algorithms. Technical report, School of Operations Research and Industrial Engineering, Cornell University, January 1994. Revised july 1995.

[133] M. J. Todd and Y. Ye. Approximate Farkas lemmas and stopping rules for iterativ infeasible interior-point algorithms for linear programming. Technical report, School of Operations Research and Industrial Engineering, Cornell University, November 1994.

[134] J. A. Tomlin. A note on comparing simplex and interior point methods for linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point Algorithms and Related Methods*, pages 91–103. Springer Verlag, 1989.

[135] J.A. Tomlin and J.S. Welch. Formal optimization of some reduced linear programming problems. *Math. Programming*, 27(2):232–240, 1983.

[136] J.A. Tomlin and J.S. Welch. A pathological case in the reduction of linear programs. *Oper. Res. Lett.*, 2:53–57, 1983.

[137] J.A. Tomlin and J.S. Welch. Finding duplicate rows in a linear program. *Oper. Res. Lett.*, 5:7–11, 1986.

[138] P. Tseng. A simple complexity proof for polynomial-time linear programming algorithm. *Oper. Res. Lett.*, 8:155–159, 1989.

[139] K. Turner. Computing projections for Karmarkar algorithm. *Linear Algebra Appl.*, 152:141–154, 1991.

[140] L. Vandenberghe and S. Boyd. Positive definite programming. Technical report, Informations Systems Laboratory Eectrical Engineering Department, Stanford University, Stanford CA 94305, 1994. Available from: ftp://ils.stanford.edu.

[141] R. J. Vanderbei. Symmetric quasi-definite matrices. *SIAM J. on Optim.*, 5(1), February 1995.

[142] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior point methods. *Math. Programming*, 58:1–32, 1993.

[143] R. J. Vanderbei, B. A. Freeman, and M. S. Meketon. A modification of Karmarkar's algorithm. *Algorithmica*, 1:395–409, 1986.

[144] S. A. Vavasis and Y. Ye. A priml-dual accelerated interior point method whose running time depends only on A. Technical report, Department of Computer Science, Upson Hall, Cornell University, Ithaca, NY 14853, December 1994.

[145] J. -P. Vial. Computational experience with a primal-dual algorithm for smooth convex programming. Technical report, 1993. To appear in Optimization Methods and Software.

[146] H. P. Williams. A reduction procedure for linear and integer programming models. In M. H. Karwan et al., editor, *Redundancy in Mathematical programming*, pages 87–109. Springer Verlag, Berlin, 1983.

[147] M. H. Wright. Interior methods for constrained optimization. In A. Iserles, editor, *Acta Numerica 1992*, pages 341–407. Cambridge University Press, Cambridge, 1992.

[148] X. Xu, P. -F. Hung, and Y. Ye. A simplified homogeneous and self-dual linear programming algorithm and its implementation. Technical report, Department of Management Sciences, The University of Iowa, 1993.

[149] G. Xue and Y. Ye. An efficient algorithm for minimizing a sum of Euclidean norms with applications. Technical report, Department of Computer Science and Electrical Engineering, The University of Vermont, June 1995. University of Vermont Research Report, CSEE/955/01-01, to appear in SIAM J. on Optimization.

[150] Y. Ye. *Interior algorithms for linear, quadratic, and linearly constrained convex programming.* PhD thesis, Stanford University, Stanford, CA, 1987.

[151] Y. Ye. On the finite convergence of interior-point algorithms for linear programming. *Math. Programming*, 57:325–335, 1992.

[152] Y. Ye. On the von Neumann economic growth problem. Technical Report Working paper series NO 92-9, Department of Management Sciences, The University of Iowa, 1992.

[153] Y. Ye. On homogeneous and self-dual algorithm for LCP. Technical report, College of Business Administration, The University of Iowa, Iowa City, IA, 1994. To appear in Math Programming.

[154] Y. Ye. Interior-point algorithm: Theory and analysis. Department of Management Sciences, The University of Iowa, 1995.

[155] Y. Ye, O. Güler, R.A. Tapia, and Y. Zhang. A quadratically convergent $O(\sqrt{n}L)$-iteration algorithm for linear programming. *Math. Programming*, 59(2):151–162, 1993.

[156] Y. Ye, M. J. Todd, and S. Mizuno. An $O(\sqrt{n}L)$ - iteration homogeneous and self-dual linear programming algorithm. *Math. Oper. Res.*, 19:53–67, 1994.

[157] Y. Zhang and D. Zhang. On polynomility of the Mehrotra-type predictor-corrector interior-point algorithm. *Math. Programming*, 68(3), 1995.

[158] J. Zhu. A path following algorithm for a class of convex programming problems. *Zeitschrift für Operations Research*, 36:359–377, 1992.