Computational Improvements of Interior Point Methods for Linear Programming.

(Draft Copy)

Marco Colombo

Doctor of Philosophy
University of Edinburgh
2007

For my family and my friends.
Se’n foi me de chesta brögna ché?

Declaration

Abstract

This research studies two implementable techniques that improve the practical performance of existing computational implementations of interior point methods for linear programming. Also, it provides the analysis of the concept of symmetric neighbourhood as the driving tool for the analysis and understanding of the good performance of some practical algorithms.

The use of the symmetric neighbourhood and the recent theoretical understanding of the behaviour of Mehrotra's corrector direction motivate the introduction of a weighting mechanism that can be applied to any corrector direction, whether originating from Mehrotra's predictor–corrector algorithm or as part of the multiple centrality correctors technique. Such modification on the way a corrector direction is applied concentrates on ensuring that such a direction can positively contribute to a successful iteration by increasing the overall stepsize. Also, it tries to use the information from a corrector direction even when otherwise it would be rejected because it produces a short stepsize. The usefulness of the weighting strategy is documented through a complete numerical experience on various sets of publicly available test problems.

The second advance is the development of an efficient way of constructing a starting point for structured large-scale stochastic linear programs. We generate a computationally viable warmstart point by solving to low accuracy a stochastic problem of much smaller dimension. The solution to the reduced problem is then expanded to the size of the problem instance, and used to initialise the interior point algorithm. The performance of this warm-start strategy is verified through a series of tests on problem instances coming from various stochastic programming sources.

Acknowledgements

I would like to thank Prof. Jacek Gondzio for proposing such an interesting and challenging project for me. This allowed me to focus simultaneously on three of my favourite subjects, Interior Point Methods, Stochastic Programming and Software Development. Moreover, his dedicated support has been crucial to my successful completion of this project.

Dr. Andreas Grothey's contribution to the success of this research cannot be underestimated. Besides having spent the last few years carrying forward the development of a fantastic software, he had the patience of introducing me to it and helping me out in the frequent occasions of panic.

Many friends deserve to be mentioned here as well, for the time spent together, the conversations, and the help that I received in many forms from them. So thanks to Aghlab, Gaurav, Iria, and to those that have left from Edinburgh like Paulo, Eleni and Giorgos. Cathy, David, Elingborg, Rosemary and Tomas have been great officemates and friends too!

Among my friends in Italy, how can I not mention Gigi and Betti, Devit and Ciarli? And plenty of others too!

I got lots of support from my family, in particular in the last few months. It's been really nice feeling cheered and spurred by them!

I am particularly indebted to Andrea, who has been close to me through most of this time, and who would have surely proofread several drafts of this document.

Table of Contents

Chapter 1

Background and introduction.

In this chapter we present the background and motivations for this research. In particular we introduce the field of linear programming, discuss its relevance, and compare some solution methods, in particular with regard to their computational complexity. Further, we motivate our research and provide an outline of the chapters of this thesis.

1.1 Linear programming

Linear programming is a relatively new discipline in the mathematical spectrum. Linear programming was developed as models were being used for economic and military planning in the years immediately following the end of World War II. The realisation of its usefulness came together with the development of a solution method, the simplex method; the introduction of the first computer calculators was crucial to the blossoming and increase of this newly born area of study.

Historical accounts on the birth and development of linear programming can be found in many sources, such as [15, ch. 2] and [61]. Dantzig's personal recollection are also in [16].

A definition of Linear programming has been given by Dantzig [16]:

> Linear programming can be viewed as part of the great revolutionary development which has given mankind the ability to state general goals and to lay out a path of detailed decisions to take in order to "best" achieve its goals when faced with practical situations of great complexity.

Further, Dantzig [16] mentions the essential components of linear programming:

> Our tools for doing this are ways to formulate real-world problems in detailed mathematical terms (models), techniques for solving the models (algorithms), and engines for executing the steps of algorithms (computers and software).

An *optimization problem* can be described in terms of a set of variables, some constraints, and an objective function. The investigation of optimization problems stems from the natural desire to solve a problem in the "best possible way". It is interesting to note that while the need of an objective function is obvious to us, it was not when the first problems were modelled: the set of feasible solution used to be investigated with some ad-hoc criteria, instead of being guided by the optimization of some quantity as objective [16].

A *linear programming problem* is an optimization problem for which the objective function and the constraints are linear. Linear programming problems arise directly (for example in economics, networks, scheduling or other applications), or as approximations to more complicated formulations, as most real-life relationships are nonlinear. Another important source of linear programs is the continuous relaxation of integer programming problems.

Among the class of convex optimization problems, linear programming has a peculiar feature which is described by the following Theorem (see, for example, [18]).

Theorem 1.1 (Fundamental theorem of linear programming)**.** *For a consistent linear programming problem with a feasible domain $\mathcal{P}$, the optimal objective value is either unbounded or is achievable at least at one extreme point of $\mathcal{P}$.*

The set of linear constraints defines a *polyhedron* that constitutes the feasible region. According to Theorem 1.1, in looking for a solution we can restrict our attention to the vertices of this polyhedron. The polyhedron corresponding to a linear system of m constraints in n variables ($m < n$) has a number of vertices equal to

$$\binom{n}{m} = \frac{n!}{m!(n-m)!}. \tag{1.1}$$

This number is an overestimate, as not all of these choices correspond to feasible points.

The fact that the number of vertices is finite guarantees termination of any algorithm that explores all vertices. However this number is exponential, as can be seen by further manipulating (1.1) [18, ch. 5.2]:

$$\frac{n!}{m!(n-m)!} \geq \left(\frac{n}{m}\right)^m \geq 2^m \quad \text{for } n \geq 2m.$$

This consideration gives rise to the need of defining an algorithm that discovers an optimal vertex among the multitude of nonoptimal ones.

Many algorithms have been proposed to solve a variety of optimization problems. However, despite their diversity, they are based on the same general framework summarised in the following scheme:

```
Given an initial iterate;
Repeat for $k = 0, 1, 2, \dots$
    ∘ Determine a search direction.
    ∘ Determine how far to move along it.
    ∘ Move to the next point.
Until some termination criteria are met.
```

Each element of this simplified algorithm (starting point, search direction, stepsize, termination criteria) has to be specialised to the specific algorithm we are talking about.

1.1.1 The simplex method

The simplex method was introduced by George Dantzig in 1947. We should note that the introduction of the simplex method happened simultaneously with the realisation of linear programming as an efficient modelling tool for practical decision making.

The simplex method exploits the insight provided by the fundamental theorem of linear programming (1.1). The simplex method reaches a solution by visiting a sequence of vertices of the polyhedron, moving from one vertex to an adjacent one characterised by a better objective function value (in the non-degenerate case).

Since the number of vertices is finite, then termination is guaranteed. While it is possible that all of them get visited, in real-life problems, this situation never arises, and only a fraction of the vertices are actually traversed (this discussion will be continued in Section 1.1.2).

Moreover, given the monotonic way of choosing the next vertex, in the non-degenerate case the set of possible vertices decreases after each iteration. Degeneracy happens when a vertex in $\mathbb{R}^m$ is traversed by $p > m$ constraints. In this case, there are $\binom{p}{m}$ different bases for the same vertex. In such a case, the step length produced by the ratio test is zero. Therefore, the simplex method may try different changes of basis without actually moving away from the vertex and thus not obtaining any improvement in the objective function value.

For the practical efficiency, the simplex algorithm has been considered for a long time the undiscussed method for solving linear programming problems.

1.1.2 Complexity considerations

The *computational complexity* of an algorithm can be used as a measure of the growth in the computational effort required by the algorithm as a function of the size of the problem. Therefore, it provides a worst-case measure. The concept of computational complexity has been introduced in the 70s, as the availability of computing machines required a deeper insight on the computational performance of different algorithms. An exhaustive presentation of this topic exceeds the aims of this thesis: we refer the reader to available introductions on the area, [61, ch. 2] among others.

Complexity proofs rely on two assumptions that are necessary simplifications:

1. Computations are performed in exact arithmetic;
2. The numerical data (A, b, c) of a problem instance is rational.

Computational complexity is measured by the number of elementary operations required to perform the algorithmic steps until termination. It often depends on the size of the binary representation of the input (L).

The simplex method has exponential complexity: it is possible that all the vertices of the feasible polyhedron have to be visited before reaching an optimal solution. Klee and Minty [42] were the first to provide an example of bad behaviour of the simplex method (when Dantzig's pivoting rule is used). In this example of n variables and $2n$ inequalities, the simplex method visits each of the 2^n vertices. The same example does not cause this pathological behaviour if different pivoting rules are implemented (see [11, ch. 4]).

However, no cases of exponential number of iterations have been encountered in real-life situations. Moreover, in most cases the simplex algorithm shows to have a polynomial behaviour, being linear in m and sublinear in n [18, p.94]. A survey on the efficiency of the simplex method is done by Shamir [62], where also a probabilistic analysis (as opposed to worst-case analysis) is presented.

The gap between the observed and the theoretical worst-case performance of the simplex method is still unexplained. Given this theoretical drawback, there have been efforts to find an algorithm for linear programming characterised by a polynomial-time bound.

In 1979 Khachiyan showed how to adapt the ellipsoid method for convex programming to the linear programming case. In Khachiyan's ellipsoid method, the feasible polyhedron is inscribed in a sequence of ellipsoids of decreasing size. The first ellipsoid has to be large enough to include a solution to the system of inequalities $Ax < b$; the volume of the successive ellipsoids shrinks geometrically. Therefore it generates improving iterates in the sense that the region in which the solution lies is reduced at each iteration in a monotonic fashion. The algorithm either finds a solution, as the centres of the ellipsoids converge to the optimal point, or states that no solution exists. More details on the ellipsoid method can be found for example in [61, ch. 13] and [56, ch. I.6].

The exciting property of the ellipsoid method is that it finds a solution in $\mathcal{O}(n^2L)$ iterations, thus has polynomial complexity. Khachiyan's contribution settled the question on the computational complexity of linear programming. However, since this worst-case bound is generally attained, its practical performance is not competitive with other solution methods. Besides, it displays other drawbacks related to large round-off errors and the need of dense matrix computation. Nevertheless, the ellipsoid method is often used in the context of combinatorial optimization as an analytic tool to prove complexity results for algorithms [56].

1.1.3 Interior point methods

Interior point methods were being developed in the 60s and the beginning of the 70s as methods to solve nonlinear programming problems with inequality constraints. However, they fell from favour and attracted less and less attention because of their inefficiency and the presence of strong competitors such as sequential quadratic programming.

Since their reintroduction, this time to solve linear programs, following Karmarkar's groundbreaking paper [41], interior point methods (IPMs for short) have attracted the interest of a growing number of researchers. Also this algorithm was proved to have polynomial complexity: indeed, it converges in $\mathcal{O}(nL)$ iterations. As opposed to Khachiyan's ellipsoid method, Karmarkar's algorithm would actually perform much better than what the bound states.

The main idea behind interior point methods is fundamentally different to the one that inspires the simplex algorithm: the optimal vertex is approached by moving through the interior of the feasible region. This is done by creating a family of parametrised approximate solutions that asymptotically converge to the exact solution. Therefore, by embedding the linear problem in a nonlinear context, an interior point method escapes the "curse of dimensionality" characteristic of the dealing with the combinatorial features of the linear programming problem.

Karmarkar [41] explained the advantage of an interior point approach as follows:

> In the simplex method, the current solution is modified by introducing a nonzero coefficient for one of the columns in the constraint matrix. Our

method allows the current solution to be modified by introducing several columns at once.

Karmarkar announced that his method was extremely successful in practice, claiming to beat the simplex method by a large margin (50 times, as reported in [66]). A variant of Karmarkar's original algorithm was then proposed and implemented by Adler-Karmarkar-Resende-Veiga (Math. Prog. 44, 1989). Since then the theoretical understanding has considerably improved, and many algorithmic variants have been proposed, and several of them have shown to be a computationally viable alternative to the simplex method. For details on Karmarkar's algorithm, we refer to [18, ch.6].

Over the last 20 years, an impressive wealth of theoretical research has been published, and computational developments have brought life to a field, that of Linear Programming, that seemed not to attract much attention anymore. Among the positive consequences brought by the renewed interest in linear programming, let us remind the improvements in the implementations of simplex-based solvers [6, 7].

There are classes of problems that are best solved with the simplex method, and others for which an interior point method is preferred. Size, structure and sparsity play a major role as to which algorithm should be chosen for computations. As a rule of thumb, with the increase of the problem dimension, the more effective interior point methods become. However, this does not hold in the hyper-sparse case, where the simplex method is virtually unbeatable [7, 30], and for network problems, where the specialised network simplex method can exploit the structure in an extremely efficient manner.

Interior point methods are well-suited to solving very large scale optimization problems. Their theory is well understood [67] and the techniques used in their implementation are documented in extensive literature (see, for example, [1, 25] and the references therein). They can be applied to a wider range of situations with no need of major changes. In particular, they have been successfully applied to complementarity problems, quadratic programming and convex nonlinear programming.

1.2 Motivation of this thesis

Optimization algorithms are extremely important in real-life applications. Theoretical advances are necessary for the understanding of the current state and the opening of new avenues of research.

However, theory per se has rarely a direct impact on the lives of those who use optimization as a tool to solve their problems. It is therefore necessary that the understanding gathered from theoretical studies is then transformed into effective practical tools. This usually requires the implementation of computer programs with the aims of accuracy, speed and reliability.

The process of creating computationally efficient methods from theoretical studies is not as direct as it might sound, but is somewhat of an art in itself. It often involves relaxing many of the theoretical assumptions, while ensuring other properties and conditions. Therefore we will put great effort in accompanying theoretical results with the corresponding computational considerations. While in a few cases these can be treated simultaneously, generally that will not be possible. There are a few reasons for this:

- Theoretical assumptions may not be realistic: this is the case when a condition stated in a theorem is not realistically satisfiable in practice (for example, bounds on some quantities).

- Theoretical assumptions may be too restrictive: this happens when the theory predicts a certain behaviour under some conditions but actually in practice it happens anyway, or provides a worst-case result which may be far from the average one.
- Theoretical requirements may be computationally expensive: this happens when the satisfaction of a certain condition is not computationally viable, in which case workarounds are usually employed.

On the other hand, the practical implementation of an optimization algorithm happens in a context that is not amenable to theoretical analysis for the following reasons:

- Finite precision of floating-point arithmetic;
- Heuristic choices;
- Dependence on the numerical inputs;
-

1.2.1 Scope of this thesis

We introduce and formalise the concept of symmetric neighbourhood as the driving tool for the analysis and understanding of the good behaviour of some practical algorithms. The use of the symmetric neighbourhood lets us simplify the presentation of the multiple centrality correctors technique [20], and further motivate their use in a new and more adaptive way, thanks to the introduction of a weighting mechanism. The practical value of the symmetric neighbourhood is appreciated also in the context of generation of a warm-start iterate for stochastic linear programs.

The original results presented in this thesis are mainly based on two papers that have been submitted for publication.

The first [12] is a joint work with Jacek Gondzio. The main objective of this paper is to analyze the efficiency of corrector directions in the light of the theoretical studies of Cartis [9, 10]. It concentrates on ensuring that a corrector direction computed at the current iterate is not rejected because it produces a short stepsize. Such a behaviour usually is manifested when the point is badly centered or highly infeasible.

The second [13] is a joint work with Jacek Gondzio and Andreas Grothey. It aims at developing an efficient way of constructing a starting point for structured large-scale stochastic linear programs. It shows that it is possible to obtain a computationally viable warmstart point by solving a stochastic problem of much smaller dimension.

1.2.2 Outline of this thesis

In Chapter 2 we introduce and derive the primal–dual path-following algorithm for linear programming. This will be the moment of introducing the main theoretical results that are at the base of most interior-point methods, as well as concentrate on those that are more used in practical implementations.

In Chapter 3 we shift our focus towards the main techniques adopted in computer implementations of interior point methods. We will discuss two important strategies used in generating effective search directions, paying particular attention to the details for which there is a divergence between theoretical analysis and practical implementation.

Chapter 4 is dedicated to studying the use of weighted correctors in the generation of search directions. This will be compared to the subspace searches approach, which tries something similar but considering a given set of directions. The advantages and drawbacks of both strategies will be discussed together with a rich computational study.

In Chapter 5 we present a warmstart technique that exploits the inherent structure of a stochastic linear programs. Alongside some theoretical results, we will show the computational experience on some standard test problems and larger instances coming from the telecommunication industry.

Finally, in Chapter 6 we present our conclusions and directions for future work.

Chapter 2

Interior point methods for linear programming.

Interior point methods are well-suited to solving very large scale optimization problems. Their theory is well understood and a number of survey papers and academic books are available [27, 66, 67].

This chapter is devoted to the derivation and analysis of primal–dual path-following interior point methods. We present the elements that are at the base of this successful class of algorithms, concentrating on their theoretical properties and attractive features.

2.1 Primal–dual path-following methods

Consider the following primal–dual pair of linear programming problems in standard form

$$(P)\quad \begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b, \\ & x \geq 0; \end{array} \qquad\qquad (D)\quad \begin{array}{rl} \max & b^T y \\ \text{s.t.} & A^T y + s = c, \\ & y \text{ free},\ \ s \geq 0, \end{array} \tag{2.1}$$

where $A \in \mathbb{R}^{m\times n}$, $x, s, c \in \mathbb{R}^n$ and $y, b \in \mathbb{R}^m$, $m < n$. We assume, without loss of generality, that A has full row rank, as linearly dependent rows can be removed without changing the solution set. This implies that a feasible $s \geq 0$ determines in a unique way the value of y. In fact, the y variables can be eliminated thus producing the symmetric combined primal–dual form studied by Todd and Ye [64].

We define the sets of primal feasible points and of primal interior points

$$\mathcal{P} = \{x : Ax = b,\ x \geq 0\}, \quad \mathcal{P}^0 = \{x \in \mathcal{P} : x > 0\},$$

and, similarly, the sets of dual feasible points and of dual interior points

$$\mathcal{D} = \{(y, s) : A^T y + s = c,\ s \geq 0\}, \quad \mathcal{D}^0 = \{(y, s) \in \mathcal{D} : s > 0\}.$$

The set of feasible primal–dual points is therefore $\mathcal{F} = \mathcal{P} \times \mathcal{D}$, and the set of primal–dual interior points is

$$\mathcal{F}^0 = \{(x, y, s) \in \mathcal{F} : (x, s) > 0\}.$$

A point $(x, y, s) \in \mathcal{F}^0$ is said to be *strictly feasible* for the primal–dual pair (2.1).

Using this notation, the primal–dual pair (2.1) can be written as

$$\min\ c^T x,\ \ x \in \mathcal{P}; \qquad \max\ b^T y,\ \ (y,s) \in \mathcal{D}, \tag{2.2}$$

We recall here some well-known results on the relationship between problems (P) and (D). These can be found in plenty of sources, for example [11, 61].

Lemma 2.1 (Weak duality)**.** *Let $(x, y, s) \in \mathcal{F}$. Then $c^T x \geq b^T y$.*

The weak duality property states that the primal and dual objective values bound each other. The difference $c^T x - b^T y$ is called *duality gap*.

Problem (P) has a solution if and only if $\mathcal{P} \neq \emptyset$; if also $\mathcal{D} \neq \emptyset$, then both problems admit an optimal solution (x^*, y^*, s^*), and the objective function values of both problems at that point coincide. This can be formalised in the following lemma.

Lemma 2.2 (Strong duality)**.** *A point $x \in \mathcal{P}$ is an optimal solution if and only if there exists a pair $(x, s) \in \mathcal{D}$ such that $c^T x = b^T y$.*

If one of the sets $\mathcal{P}$ or $\mathcal{D}$ is empty, then the other is either unbounded or empty as well. In such a case, an optimal solution for problem (2.2) does not exist,

In what follows, we make the standard assumption for the development of interior point methods that $\mathcal{P}^0 \neq \emptyset$ and $\mathcal{D}^0 \neq \emptyset$. This is also referred to as the *interior point assumption*. The interior point assumption corresponds to assuming that the primal–dual optimal face is bounded (this is mentioned in [28] and also in [29, Lemma 2.2]). Cases when this assumption does not hold can be considered by allowing the algorithm to accept infeasible iterates (see Section 2.2.2).

Optimality conditions let us recognise that a solution has been found. They also provide insight on the development of algorithms for finding a solution.

The Karush-Kuhn-Tucker (KKT) conditions express first-order optimality conditions for the primal–dual pair (2.1). They can be written as

$$\begin{array}{rcl} Ax &=& b \\ A^T y + s &=& c \\ XSe &=& 0 \\ (x, s) &\geq& 0, \end{array} \tag{2.3}$$

where $X, S \in \mathbb{R}^{n \times n}$ are diagonal matrices with elements x_i and s_i respectively, and $e \in \mathbb{R}^n$ is a vector of ones. In other words, an optimal solution is characterised by primal feasibility, dual feasibility and complementarity.

Complementarity can be seen as a certificate for optimality in linear programming [35, 61]. For non-optimal feasible iterates, complementarity measures the distance of the iterate to optimality:

$$c^T x - b^T y = c^T x - x^T A^T y = x^T (c - A^T y) = x^T s. \tag{2.4}$$

The quantity $x^T s$ is called *complementarity gap*: when it is driven to zero, then a feasible solution is also optimal. It should be noted that the equality between duality gap and complementarity gap of equation (2.4) holds only for a feasible point.

2.1.1 The barrier problem

Many algorithms used in mathematical programming can be interpreted as path-following. Here we restrict our attention to the path described by the logarithmic barrier function in linear programming. Given the linear program in standard form (P), it is possible to write the corresponding *barrier problem*:

$$(P_\mu) \quad \begin{array}{ll} \min & c^T x - \mu \sum_i \ln x_i \\ \text{s.t.} & Ax = b, \\ & x > 0. \end{array}$$

Problem (P_μ) denotes a family of problems parametrised by the quantity $\mu > 0$ (typically small), called *barrier parameter* in the interior point literature. It is worth noting that such approach is viable only if it is actually possible to find a point that strictly satisfies the constraints, that is, if $\mathcal{P}^0 \neq \emptyset$.

The presence of the logarithmic barrier forces the iterates to stay in the interior of the feasible region, as this term heavily penalises the points that are too close to the boundary. However, the influence exerted by the logarithmic barrier can be controlled through the penalty parameter μ. The weight on the barrier regulates the distance from the iterates to the boundary: as μ tends to zero, problem (P_μ) resembles more and more problem (P).

The objective function of problem (P_μ) is a strictly convex function. Therefore, for a fixed μ, the problem has at most one global minimum. The minimizer, if it exists, is completely characterised by the associated KKT conditions:

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

By substituting $s = \mu X^{-1} e$, we obtain the standard (primal–dual) formulation of the so called *perturbed KKT conditions*:

$$\begin{array}{rcl} Ax & = & b \\ A^T y + s & = & c \\ XSe & = & \mu e \\ (x, s) & > & 0. \end{array} \tag{2.5}$$

$$\boxed{\text{If the feasible domain } \mathcal{P} \text{ is bounded, then both } (P) \text{ and } (P_\mu) \text{ have optimal solution.}}$$

The following lemma has been proved by Megiddo [48].

Lemma 2.3. *Problem (P_μ) is either unbounded for every $\mu > 0$ or has a unique optimal solution for every $\mu > 0$.*

If the perturbed KKT system (2.5) has a solution for any $\mu > 0$, then it determines a unique continuous smooth curve $(x(\mu), y(\mu), s(\mu))$ toward the optimal set as $\mu \to 0$. In interior-point terminology, this curve is called the *central path*. We postpone its presentation to Section 2.1.3.

Under the assumptions that for a particular $\mu > 0$ the point $(x(\mu), y(\mu), s(\mu))$ is primal and dual feasible, we can state a similar result to what is expressed by (2.4):

$$g(\mu) = c^T x(\mu) - b^T y(\mu) = x(\mu)^T s(\mu),$$

that is, the duality gap corresponds to the complementarity gap. Hence reducing either of them is identical. Also, as $XSe - \mu e = 0$ implies $x_i s_i = \mu$, $i = 1, \ldots, n$, we have

$$s(\mu)^T x(\mu) = n\mu, \tag{2.6}$$

and for $\mu \to 0$, also $g(\mu) \to 0$. Megiddo [48] shows that $c^T x(\mu) \to c^T x^*$ as $\mu \to 0$. Furthermore, he proves the following, stronger result.

Lemma 2.4. *Under the assumptions of primal feasibility, dual feasibility, and full row rank of matrix A, then*

$$x(\mu) \to x^*, \quad (y(\mu), s(\mu)) \to (y^*, s^*)$$

as $\mu \to 0$.

2.1.2 Solving the perturbed KKT conditions

Primal–dual path-following methods solve the perturbed KKT conditions (2.3) by asking the complementarity pairs to align to a specific barrier parameter $\mu > 0$,

$$XSe = \mu e, \tag{2.7}$$

while enforcing $(x, s) > 0$. However, up to now, we have not defined how to choose the barrier parameter μ and how to update it at each iteration.

Explain how we choose it at the beginning.

At each iteration, μ is monotonically decreased by the quantity $\sigma \in (0, 1)$, called *centering parameter* for reasons that will become clear later on. Hence, the perturbed KKT conditions (2.5) approximate better and better the system (2.3) of optimality conditions for the original problem. The choice of the centering parameter σ is algorithm-dependent. We provide some theoretical insights on some possible choices in Section 2.2.

Path-following interior point methods seek a solution to the system of equations (2.5)

$$F(x, y, s) = \begin{bmatrix} Ax - b \\ A^T y + s - c \\ XSe - \sigma\mu e \end{bmatrix} = 0,$$

which is nonlinear in the perturbed complementarity constraints. We use Newton's method to linearise the system around the current point according to

$$\nabla F(x, y, s) \Delta(x, y, s) = -F(x, y, s),$$

and obtain the so-called step equations

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ -XSe + \sigma\mu e \end{bmatrix} = \begin{bmatrix} \xi_b \\ \xi_c \\ \xi_\mu \end{bmatrix}, \tag{2.8}$$

which need to be solved for a search direction $\Delta w = (\Delta x, \Delta y, \Delta s)$, with $\mu = x^T s/n$, $\sigma \in (0,1)$. For a feasible algorithm $\xi_b = \xi_c = 0$.

The solution of system (2.8) is the computationally dominant step in each iteration of an interior point algorithm. Throughout this thesis, we will restrict our attention to using a direct approach in solving these equations. We should note, however, that a wealth of research explored the use of iterative methods in the computation of the search direction [34, 57].

System (2.8) is usually reduced to two other formulations by exploiting the block structure of its matrix. The *augmented system* formulation is obtained by using the last row of (2.8) to eliminate $\Delta s = X^{-1}(\xi_\mu - S\Delta x)$. This produces

$$\begin{bmatrix} -X^{-1}S & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} \xi_c - X^{-1}\xi_\mu \\ \xi_b \end{bmatrix}, \tag{2.9}$$

which is a symmetric but indefinite system. By further eliminating Δx, we reduce system (2.9) to the set of *normal equations*

$$AD^2A^T\Delta y = AD^2(\xi_c - X^{-1}\xi_\mu) + \xi_b, \tag{2.10}$$

where we introduced the notation $D^2 = S^{-1}X$. Under the assumption of full row rank for A, matrix AD^2A^T is positive definite, since $D_i^2 = x_i/s_i > 0$ for all $i = 1, \dots, n$.

Besides the issue of definiteness, the two formulations differ in terms of sparsity and conditioning, the normal equations usually being denser and worse conditioned. The choice between augmented system and normal equations depends also on the relative density of AD^2A^T with respect to A. Normal equations are to be avoided when there are dense columns in A, as they generate dense blocks in AD^2A^T.

The augmented system formulation requires particular attention in the development of linear algebra routines for the fact that it presents an indefinite matrix. This raises problems of numerical stability, and an accurate choice of pivoting strategies is fundamental. Maros and Mészáros [47] presented an in-depth study of the properties of the augmented system formulation.

2.1.3 The central path

The study of the primal–dual properties of central path was pioneered by Megiddo [48] and Bayer and Lagarias [3].

We have seen in Lemma 2.4 that as $\mu \to 0$ the solution to (P_μ) converges to the optimal solutions of (P). More formally, the central path leads to a solution which is characterised by strict complementarity.

Theorem 2.5 (Strict complementarity)**.** *If (P) and (D) are feasible, then there exist a point $x^* \in \mathcal{P}$ and a pair $(y^*, s^*) \in \mathcal{D}$ such that*

$$(x^*)^T s^* = 0 \quad \text{and} \quad x_i^* + s_i^* > 0, \;\; i = 1, \dots, n.$$

A solution (x^*, s^*) that satisfies the above theorem is called *strictly complementary.* On the grounds of a strictly complementary solution we can define the concept of *optimal partition.* Following Jansen [35], we define the support set of a vector $v \in \mathbb{R}^n$ as

$$\sigma(v) = \{i : v_i > 0, \; i = 1, \dots, n\},$$

and partition the set of indices $\{1, \dots, n\}$ as

$$\mathcal{B} = \sigma(x^*), \quad \mathcal{M} = \sigma(s^*).$$

This partition is well-defined in the sense that a strictly complementary solution satisfies both $\mathcal{B} \cap \mathcal{M} = \emptyset$ and $\mathcal{B} \cup \mathcal{M} = \{1, \dots, n\}$. A proof of the latter result, also known as Goldman-Tucker theorem, can be found in [67]. The concepts of strict complementarity and optimal partition are recurrent motifs in the analysis of interior point methods.

Vavasis and Ye [65] studied the properties of the curvature of the central path, discovering that the central path is characterised by $\mathcal{O}(n^2)$ curves of high degree and segments where it is relatively straight. Such curves appear in correspondence with changes in the optimal partition. Close to the end, when the optimal partition has been identified, the central path becomes a straight line [48]. In this region, the algorithm displays the quadratic convergence property typical of Newton's method.

We now consider the limit of (P_μ) for $\mu \to \infty$, and therefore find the point from which the central path departs. This corresponds to finding the point that minimizes the barrier function:

$$\hat{x} = \arg\min_{x \in \mathcal{P}^0} \left(-\sum_i \ln x_i \right).$$

The point $\hat{x}$ is the *analytic center* of the feasible polytope, and was first studied by Sonnevend. Given the strict convexity of the barrier function, the concept of analytic center is well defined. As the analytic center minimizes the barrier, it is the point farthest away from the boundary.

However, there is a problem with defining the central path in terms of analytic center: the central path is affected by the presence of redundant constraints. This happens because it is an exclusively analytic concept, which does not exploit geometric considerations. Other type of centers (center of gravity, center of the ellipsoid of maximum volume that can be inscribed in $\mathcal{P}$, volumetric center) can be defined, but they usually are too demanding to compute [27].

2.1.4 Neighbourhoods of the central path

As we have seen, following the central path is the recommended way of traversing the interior of the feasible region towards the optimal solution. Nevertheless, it should be clear that keeping the iterates *exactly on* the central path is an unachievable aim. Finding a point that solves the perturbed complementarity conditions (2.7) for a specific μ is as difficult as solving the optimization problem itself. Therefore, we never insist on this extremely restrictive requirement, but we rather allow the iterates to be somewhere around the central path. This leads to the introduction of the important concept of *neighbourhood* of the central path. We can define several neighbourhoods, characterised by different properties. Two neighbourhoods are often used in theoretical developments.

The first is based on the Euclidean norm, and it is often referred to as the *tight neighbourhood*:

$$\mathcal{N}_2(\theta) = \{(x, y, s) \in \mathcal{F}^0 : \|XSe - \mu e\|_2 \leq \theta\mu\}.$$

This neighbourhood defines points which lie very close to the central path: search directions generated from points in this neighbourhood can be followed with full step, and the barrier parameter can be decreased by a small amount at each iteration (giving rise to the name of *short-step algorithms* to the algorithms that are based on this neighbourhood). The closeness to the central path that the tight neighbourhood imposes and maintains allows to produce the best convergence result for linear programming: short-step algorithms converge in $\mathcal{O}(\sqrt{n}L)$ iterations. However, since the reduction in the barrier parameter at each iteration is very small, the practical value of short-step algorithms is small.

The other commonly used neighbourhood is instead based on the infinity norm, and it is often called *wide neighbourhood*:

$$\mathcal{N}_{-\infty}(\gamma) = \{(x, y, s) \in \mathcal{F}^0 : x_i s_i \geq \gamma\mu, \ \forall i\}.$$

Algorithms based on such a neighbourhood are allowed to generate iterates that follow more loosely the central path. The iterates have more freedom of movement as they can approach the boundary of the feasible set. Hence, algorithms based on the wide neighbourhood (usually denoted as *long-step algorithms*) are less conservative and can decrease the barrier parameter more rapidly. However, the Newton direction computed from points in the wide neighbourhood has weaker properties, and a linesearch procedure is needed to ensure that the positivity of the (x, s) iterates is preserved.

In Section (2.3) we will study a variation of the $\mathcal{N}_{-\infty}$ neighbourhood which better describes the centrality requirements needed for a practical algorithm.

2.2 Theoretical results

Theoretical developments aim at lowering the upper bound on the number of steps needed for convergence. The results provided by such worst-case complexity analysis are informative but exceedingly pessimistic.

Theoretical proofs of complexity generally follow a common scheme. First they rely on a computable measure of the closeness to the central path: this is accomplished by the concept of neighbourhood. Second, they show that the direction computed by solving the Newton system (2.8) can be followed with a strictly positive step (and therefore guarantees some progress) and generates an iterate that retains the property of being in some neighbourhood of the central path (possibly larger than the one before). Finally, they require a decrease in the barrier parameter that allows to derive a polynomial upper bound on the number of iterations.

The theoretical analysis for the logarithmic barrier method was initiated by Megiddo [48]. On this basis, Kojima, Mizuno and Yoshise (1989) proposed a polynomial-time algorithm with the property of convergence in $\mathcal{O}(nL)$ iterations. This was later refined by the same group [44] and by Monteiro and Adler [55]. Both papers presented a primal–dual algorithm for linear programming based on the tight $\mathcal{N}_2$ neighbourhood with the property of convergence in $\mathcal{O}(\sqrt{n}L)$ iterations.

2.2.1 Feasible methods

A feasible algorithm is characterised by the requirement that all primal and dual iterates always lie within the interior of the feasible region. For this reason, these algorithms need to start from a strictly feasible point.

A short-step feasible method based on the $\mathcal{N}_2$ neighbourhood reduced the barrier parameter by

$$\sigma = 1 - \delta/\sqrt{n}$$

at each iteration, for some positive constant δ.

In complexity proof, the barrier parameter is reduced slightly at each iteration in order to guarantee that one iteration of Newton's method can keep the point in the tight neighbourhood of the central path. However, this is a worst-case analysis, and in practice the same would happen even for a bigger update of the barrier parameter. This brings to study ways of allowing a more substantial reduction of the barrier parameter, at least in some iterations.

One important result is this direction was obtained by Mizuno, Todd and Ye [54], who analysed the short-step predictor–corrector method. Their strategy uses two nested neighbourhoods $\mathcal{N}_2(\theta^2)$ and $\mathcal{N}_2(\theta)$, $\theta \in (0,1)$, and exploits the quadratic convergence property of Newton's method in such a tight neighbourhood of the central path. Their algorithm alternates between two search directions characterised by different properties. First, by choosing $\sigma = 0$ in (2.8), the predictor direction gains optimality, possibly at the expense of worsening the centrality, keeping the iterate in a larger neighbourhood $\mathcal{N}_2(\theta)$ of the central path. Then, a pure re-centering step is performed, by setting $\sigma = 1$, leaving the duality gap unchanged but moves the iterate back into a tighter $\mathcal{N}_2(\theta^2)$ neighbourhood. Hence, every second step the algorithm produces a point in $\mathcal{N}_2(\theta^2)$.

An important contribution of this technique, is the idea of targeting optimality and centrality independently. However, the use of the very restrictive $\mathcal{N}_2$ neighbourhood makes it unattractive for practical applications. While this algorithm is not effective in practical terms, it provides a scheme upon which more computationally attractive methods can be constructed, as we will discuss in Chapter 3.

Thanks to the optimizing predictor direction which is identical to a short-step feasible method, also the predictor–corrector algorithm converges in $\mathcal{O}(\sqrt{n}L)$ iterations, with the only difference that the value of the barrier parameter is reduced over two iterations.

2.2.2 Infeasible methods

Finding a strictly feasible starting point is, in general, a nontrivial task. Solving the feasibility problem is an optimization problem in its own right, and is as difficult as solving the original problem to optimality. Moreover, the feasible region may have empty interior, in which case the theory developed above does not apply. For these reasons, a need exists for practical implementations to dispense from this requirement.

A way to recover a strictly feasible starting point involves solving an artificial Phase I subproblem by using the big-M method. However, the performance is dependent on the choice of the values given to the weights, and the use of very large values, while theoretically satisfying, causes numerical instabilities [45]. This is worsened by the presence of dense columns that compromises the computational efficiency.

A very different approach is based on the homogeneous self-dual formulation [67, ch. 9]. The self-dual formulation wraps the optimization problem into one of larger dimension, but for which a feasible solution is known from the start. Also, the homogeneous self-dual formulation has the very appealing property of being able to detect infeasibility with accuracy.

The use of a self-dual formulation is not attractive from a computational viewpoint, particularly because of the need of one extra backsolve at each iteration. Stability issues? Find a reference for that (Terlaky-Ye?).

It is possible to develop an algorithm which only requires the x and s components to be strictly positive. This was initiated by Lustig [45], who proposed some new feasibility restoration directions.

However, was the work of Kojima, Megiddo and Mizuno [43] that introduced a complete infeasible algorithm, with full theoretical analysis of convergence. Sure? Wright (page 109) says it was Lustig Marsten and Shanno.

In such an algorithm, all iterates are infeasible, but the limit points are feasible and optimal. This is obtained by using a neighbourhood that admits infeasible points:

$$\mathcal{N}_{-\infty}(\gamma,\beta) = \{(x,y,s) |\ \|(r_b,r_c)\| \leq \beta\mu\frac{\|(r_b^0,r_c^0)\|}{\mu_0}, (x,s) > 0, x_i s_i \geq \gamma\mu\},$$

where $\gamma \in (0,1)$ and $\beta \geq 1$ are parameters, and we denoted the primal and dual residuals, respectively, by $r_b = Ax - b$ and $r_c = A^T y + s - c$.

Therefore, there is no strict feasibility requirement for the iterates; however, the residuals at each iteration must be bounded above by a multiple of the duality measure μ. By reducing μ we can force the primal and dual residuals r_b and r_c to zero, thus approaching complementarity and feasibility at the same speed.

Kojima, Megiddo and Mizuno [43] proposed a stepsize rule that guarantees global convergence of an infeasible interior-point algorithm. An algorithm is globally convergent if it is possible to choose a strictly positive stepsize such that the complementarity gap is reduced at each iteration. This property is very important, at it guarantees the good behaviour of the algorithm for any given starting point. However, in order to prove it, it is required that some safeguards are implemented:

- reduction in infeasibility should be faster than in complementarity
- $x_i s_i \geq \beta x^T s$ for a chosen β, and all iterates must stay in this neighbourhood.

Let $(x', y', s') = (x^0, y^0, s^0) + \alpha(\Delta x, \Delta y, \Delta s)$, then:

- $\xi'_P = (1-\alpha)\xi_P^0$
- $\xi'_D = (1-\alpha)\xi_D^0$
- $x'^T s' = (1-\alpha(1-\sigma))(x^0)^T s^0 + \alpha^2 \Delta x^T \Delta s$

so the reduction in infeasibilities is linear, while complementarity reduces for a small α (but if the point is feasible, then the reduction is linear as well, since $\Delta x^T \Delta s = 0$.)

Order of convergence: $\mathcal{O}(n^2 L)$ in Wright.

2.3 Symmetric neighbourhood

The quality of centrality (understood in a simplified way as complementarity) is *not* well characterised by either of two neighbourhoods $\mathcal{N}_2$ or $\mathcal{N}_{-\infty}$ commonly used in theoretical developments of interior point methods. Instead, we propose to use a symmetric neighbourhood $\mathcal{N}_s(\gamma)$, in which complementarity pairs have to satisfy $\gamma\mu \leq x_i s_i \leq \gamma^{-1}\mu$, where $\gamma \in (0,1)$, for a strictly feasible iterate (x, y, s).

Jansen [36] (or actually [35]?) proposed to use the ratio between the smallest and the largest complementarity pair as an indication of the quality of a point. It mentions that was Atkinson and Vaidya [2] that noticed that as the ratio increases, the region in which Newton converges becomes smaller.

Practical experience with the primal–dual algorithm in HOPDM [19] suggests that one of the features responsible for its efficiency is the way in which the quality of centrality is assessed. By "centrality" we understand here the spread of complementarity products $x_i s_i$, $i = 1, \dots, n$. Large discrepancies within the complementarity pairs, and therefore bad centering, create problems for the search directions: an unsuccessful iteration is caused not only by small complementarity products, but also by very large ones. This can be explained by the fact that Newton's direction tries to compensate for very large products, as they provide the largest gain in complementarity gap when a full step is taken. However, the direction thus generated may not properly consider the presence of very small products, which then become blocking components when the stepsizes are computed.

The notion of spread in complementarity products is not well characterised by either of the two neighbourhoods $\mathcal{N}_2$ or $\mathcal{N}_{-\infty}$ commonly used in theoretical developments of IPMs. To overcome this disadvantage, here we formalise a variation on the usual $\mathcal{N}_{-\infty}(\gamma)$ neighbourhood, in which we introduce an upper bound on the complementary pairs. This neighbourhood was implicitly used in Section 3.3 to define an achievable target for multiple centrality correctors. We define the symmetric neighbourhood to be the set

$$\mathcal{N}_s(\gamma) = \{(x, y, s) \in \mathcal{F}^0 : \gamma\mu \le x_i s_i \le \frac{1}{\gamma}\mu,\ i = 1, \dots, n\},$$

where $\mathcal{F}^0$ is the set of strictly feasible primal–dual points, $\mu = x^T s/n$, and $\gamma \in (0, 1)$.

While the $\mathcal{N}_{-\infty}$ neighbourhood ensures that some products do not approach zero too early, it does not prevent products from becoming too large with respect to the average. In other words, it does not provide a complete picture of the centrality of the iterate. The symmetric neighbourhood $\mathcal{N}_s$, on the other hand, promotes the decrease of complementarity pairs which are too large, thus taking better care of centrality.

Actually also the $\mathcal{N}_{-\infty}$ neighbourhood expresses an upper bound, but it's much higher.

The analysis is done for the long-step feasible path-following algorithm, where the search direction $(\Delta x, \Delta y, \Delta s)$ is found by solving system (2.8) with $r = (0,\ 0, -XSe + \sigma\mu e)^T$, $\sigma \in (0, 1)$, $\mu = x^T s/n$. The exposition follows closely the presentation of [67, Chapter 5].

First we need a technical result, the proof of which can be found in [67, Lemma 5.10] and is unchanged by the use of $\mathcal{N}_s$ rather than $\mathcal{N}_{-\infty}$.

Lemma 2.6. *If $(x, y, s) \in \mathcal{N}_s(\gamma)$, then* $\|\Delta X \Delta S e\| \le 2^{-3/2}\left(1 + \frac{1}{\gamma}\right) n\mu.$

Our main result is presented in Theorem 2.7. We prove that it is possible to find a strictly positive stepsize α such that the new iterate $(x(\alpha), y(\alpha), s(\alpha)) = (x, y, s) + \alpha(\Delta x, \Delta y, \Delta s)$ will not leave the symmetric neighbourhood, and thus this neighbourhood is well defined. This result extends Theorem 5.11 in [67].

Theorem 2.7. *If $(x, y, s) \in \mathcal{N}_s(\gamma)$, then $(x(\alpha), y(\alpha), s(\alpha)) \in \mathcal{N}_s(\gamma)$ for all*

$$\alpha \in \left[0, 2^{3/2}\gamma \frac{1-\gamma}{1+\gamma}\frac{\sigma}{n}\right].$$

Proof. Let us express the complementarity product in terms of the stepsize α along the direction $(\Delta x, \Delta y, \Delta s)$:

$$\begin{aligned} x_i(\alpha)s_i(\alpha) &= (x_i + \alpha\Delta x_i)(s_i + \alpha\Delta s_i) \\ &= x_i s_i + \alpha(x_i\Delta s_i + s_i\Delta x_i) + \alpha^2\Delta x_i\Delta s_i \\ &= (1-\alpha)x_i s_i + \alpha\sigma\mu + \alpha^2\Delta x_i\Delta s_i. \end{aligned} \tag{2.11}$$

We need to study what happens to this complementarity product with respect to both bounds of the symmetric neighbourhood. Let us first consider the bound $x_i s_i \le \frac{1}{\gamma}\mu$. By Lemma 2.6, equation (2.11) implies

$$x_i(\alpha)s_i(\alpha) \le (1-\alpha)\frac{1}{\gamma}\mu + \alpha\sigma\mu + \alpha^2 2^{-3/2}\left(1+\frac{1}{\gamma}\right)n\mu.$$

At the new point $(x(\alpha), y(\alpha), s(\alpha))$, the new duality gap is $x(\alpha)^T s(\alpha) = n\mu(\alpha)$. The relation $x_i(\alpha)s_i(\alpha) \le \frac{1}{\gamma}\mu(\alpha)$ holds provided that

$$(1-\alpha)\frac{1}{\gamma}\mu + \alpha\sigma\mu + \alpha^2 2^{-3/2}\left(1+\frac{1}{\gamma}\right)n\mu \le \frac{1}{\gamma}(1-\alpha+\alpha\sigma)\mu,$$

from which we derive a first bound on the stepsize:

$$\alpha \le 2^{3/2}\frac{1-\gamma}{1+\gamma}\frac{\sigma}{n} = \bar{\alpha}_1.$$

Considering now the bound $x_i s_i \ge \gamma\mu$ and proceeding as before, we derive a second bound on the stepsize:

$$\alpha \le 2^{3/2}\gamma\frac{1-\gamma}{1+\gamma}\frac{\sigma}{n} = \bar{\alpha}_2.$$

Therefore, we satisfy both bounds and guarantee that $(x(\alpha), y(\alpha), s(\alpha)) \in \mathcal{N}_s(\gamma)$ if

$$\alpha \in [0, \min(\bar{\alpha}_1, \bar{\alpha}_2)],$$

which proves the claim, as $\gamma \in (0, 1)$. □

It is interesting to note that the introduction of the upper bound on the complementarity pairs does not change the polynomial complexity result proved for the $\mathcal{N}_{-\infty}(\gamma)$ neighbourhood [67, Theorem 5.12]. Therefore, the symmetric neighbourhood provides a better practical environment without any theoretical loss. This understanding provides some additional insight into the desired characteristics of a well-behaved iterate.

Chapter 3

Practical implementations of interior point methods.

In this chapter we turn our attention to the computational side of interior point methods. We concentrate on the main strategies which are at the basis of effective implementations of interior point methods for linear programming, and present other issues that are peculiar to practical algorithms. These are documented in extensive literature (see, for example, [1, 25] and the references therein).

3.1 A practical algorithm

Interior point methods require the computation of the Newton direction (2.8) for the associated barrier problem and make a step along this direction, thus usually reducing infeasibilities and complementarity gap. The computation of the Cholesky factorization dominates the cost of each iteration. As this is usually a major computational task, the efforts in the theory and practice of interior point methods concentrate on reducing the number of times the Newton system matrix (2.8) has to be factorised.

In Section 2.2 we presented the theoretical results on the order of convergence of some interior point algorithms. In practice, convergence is much faster than stated by those results: optimality is usually reached in a number of iterations proportional to the logarithm of the problem dimension. As it happens with the analysis of the simplex method, we see quite a gap between the predicted and the observed performance that is still to be fully understood.

Find a citation for the logarithm thing!

Practical algorithms are very different from the ones used for theoretical purposes, and usually they implement some variation of infeasible interior point algorithms. In particular they show differences in the search directions, the evaluation of the stepsize, the use of the neighbourhood concept, and the update of the barrier parameter. This is further complicated by issues of computational efficiency and numerical stability, which often suggest the use of amended techniques or heuristic approaches, which make extremely difficult the analysis of the algorithms implemented in practice.

The computation of different stepsizes in the primal and dual spaces is almost the rule for linear programming. This has the advantage of speeding up the restoration of feasibility. According to [25], the use of different stepsizes contributes to a reduction in number of iterations of 10% on the Netlib set of tests.

In order to ensure that (x, s) remain positive after moving along the Δw direction, we need to employ a linesearch procedure and find the maximum feasible stepsizes α_P and α_D such that

$$x + \alpha_P \Delta x > 0, \qquad s + \alpha_D \Delta s > 0.$$

The achievable stepsizes for a given search direction Δw, in the primal and dual space respectively, are computed as:

$$\alpha_P = \min \left\{ -\frac{x_i}{\Delta x_i} : \Delta x_i < 0 \right\}, \qquad \alpha_D = \min \left\{ -\frac{s_i}{\Delta s_i} : \Delta s_i < 0 \right\}. \tag{3.1}$$

We remark that in the computation of the stepsizes, we maintain the strict positivity of the iterates, without restricting the point inside a neighbourhood. While this is done on the grounds of computational efficiency, we may lose the property of global convergence ensured by keeping the iterates in a neighbourhood of the central path.

Upon discussing the choice for the centering term in his algorithm, Mehrotra [50] makes this comment:

> It is not clear if the central path (with equal weights) is the best path to follow, particularly since it is affected by the presence of redundant constraints. Furthermore, the points on (or near) the central path are only intermediate to solving the linear programming problem. It is only the limit point on this path that is of interest to us.

3.1.1 Correcting techniques

Two techniques have proved particularly successful in reducing the number of iterations within practical algorithms: Mehrotra's predictor–corrector algorithm [50] and multiple centrality correctors [20]. These techniques have been implemented in most of commercial and academic interior point solvers for linear and quadratic programming such as BPMPD, Cplex, HOPDM, Mosek, OOPS, OOQP, PCx and Xpress.

Both correcting techniques originate from the observation that (when direct methods of linear algebra are used) the computation of the Newton direction requires two computationally intensive operations:

1. Factorization: computation of the Cholesky factors of a sparse symmetric matrix;

2. Backsolve: use of the factors just computed to solve the system.

However, the cost of computing the factors is usually significantly larger than that of backsolving: in some cases the ratio between these two computational efforts may even exceed 1000.

Present a small table that shows the cost of factoring and of backsolving for HOPDM (and PC-x?).

Consequently, it is worth adding more (cheap) backsolves if this reduces the number of (expensive) factorizations. Mehrotra's predictor–corrector technique [50] uses two backsolves per factorization; the multiple centrality correctors technique [20] allows recursive corrections: a larger number of backsolves per iteration is allowed, leading to a further reduction in the overall number of factorizations.

Since these two methods were developed, there have been a number of attempts to investigate their behaviour rigorously and thus provide further insight on their success. Such objectives are difficult to achieve because correctors use heuristics that are effective in practice but hard to analyse theoretically. Besides, both correcting techniques are applied to long-step and infeasible algorithms which have very little in common with the short-step and feasible algorithms that display the best known theoretical complexity.

These two strategies will be the focus of the next sections.

3.2 Mehrotra's predictor–corrector algorithm

Practical implementations usually follow Mehrotra's predictor–corrector algorithm [50]. In such a framework, we first generate a predictor direction to make progress towards optimality, and then we compute a corrector to remedy for some of the error made by the predictor and move the point closer to the central path.

A number of advantages can be obtained by splitting the computation of the Newton direction into two steps, corresponding to solving the linear system (2.8) independently for the two right-hand sides

$$r_1 = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ -XSe \end{bmatrix} \quad \text{and} \quad r_2 = \begin{bmatrix} 0 \\ 0 \\ \sigma\mu e \end{bmatrix}. \tag{3.2}$$

First, we can postpone the choice of the centering parameter σ and base it on the assessment of the quality of the pure Newton direction computed with right-hand side r_1; second, the error made by this direction may be taken into account and corrected. Mehrotra's predictor–corrector technique [50] translates these observations into a powerful computational method.

Mehrotra's predictor–corrector algorithm [50, 46] is extremely efficient in practice. Since its introduction, it has been considered the method of choice for practical implementations because it is usually very fast and reliable. Moreover, it has a convincing interpretation in terms of second order approximations.

3.2.1 Affine-scaling predictor direction

The *predictor direction* $\Delta_a w = (\Delta_a x, \Delta_a y, \Delta_a s)$ is obtained by solving system (2.8) with right-hand side r_1 defined in (3.2). This corresponds to computing the pure Newton direction for the original KKT system (2.3), and for this reason is often called *affine-scaling* direction. This direction is strongly optimizing, as it targets a point for which all complementarity products are zero. The achievable stepsizes for the predictor direction, in the primal and dual space respectively, are computed according to (3.1).

As it targets a point for which $XSe = 0$, the affine-scaling direction may be distracted by points that have small complementarity products but are not optimal. In particular, it may well point towards the boundary of the positive orthant or approach an infeasible vertex, generating a very small stepsize. This issue is usually worsened if the current iterate is badly centered, and therefore only a very small step is acceptable in order to maintain positivity or to keep the iterate in the neighbourhood of the central path.

Since it completely ignores the central path, the affine-scaling direction is not enough in a practical implementation of interior point methods, but it has to be complemented by other techniques. The role of the centering term is to remedy this situation by affecting the search direction to move closer to the central path, therefore allowing a longer stepsize.

Tapia et al. [63] noted that the choice of the centering parameter can be crucial both in theory and in practice, and suggest that it is a function of the Newton step. This calls for two separate backsolves at each iteration.

3.2.2 Second order corrector direction

Mehrotra's algorithm exploits a centrality corrector in order to remedy to badly centered points. The purpose of this direction is to move closer to the central path, and therefore reduce the spread in complementarity products, without aiming for more optimality. This is a somewhat conservative direction, which it is hoped will provide more room for movement at the next iteration.

One tool introduced by Mehrotra [50] is a dynamic evaluation of the centering parameter σ. It is based on a simple heuristic that evaluates the quality of the predictor direction in order to judge the amount of centering term needed. The length of the stepsizes α_P and α_D are used to predict the complementarity gap after such a step:

$$g_a = (x + \alpha_P \Delta_a x)^T (s + \alpha_D \Delta_a s). \tag{3.3}$$

The ratio $g_a / x^T s \in (0, 1)$ measures the quality of the predictor direction. A small ratio indicates a successful reduction of the complementarity gap. On the other hand, if the ratio is close to one, then very little progress is achievable in direction Δ_a, and a strong recentering is recommended.

In [50] the following choice of the new barrier parameter is suggested

$$\mu = \left(\frac{g_a}{x^T s} \right)^2 \frac{g_a}{n} = \left(\frac{g_a}{x^T s} \right)^3 \frac{x^T s}{n}, \tag{3.4}$$

corresponding to the choice of $\sigma = (g_a / x^T s)^3$ for the centering parameter.

Mehrotra's heuristic was actually more elaborate.

The centering parameter, more generally could be chosen as

$$\sigma = \left(\frac{g_a}{x^T s} \right)^p,$$

for various choices of the exponent. Mehrotra [50] studied the effect of different values $p = 1, 2, 3, 4$ for the exponent on a subset of Netlib problems, and concluded that for p between 2 and 4 there was not much difference. Also Lustig et al. [46] commented on the weak dependence of the computational performance on the choice of the exponent.

If the predictor provides a good improvement, a small σ is chosen, and therefore very little centering will be used. When, on the other hand, affine scaling produces very small stepsizes and so very little improvement can be achieved, σ will be close to one, and so a stronger recentering will occur.

A further important contribution by Mehrotra consists in the introduction of a second-order direction. As said above, the affine-scaling direction corresponds to a linear approximation to the the trajectory from the current point to the optimal set, where no information about higher-order terms is taken into account. This linearisation, however, produces an error which can be determined analytically. Assuming that a full step in the affine-scaling direction is made, the new complementarity products are equal to

$$(X + \Delta_a X)(S + \Delta_a S)e \;=\; XSe + (S\Delta_a x + X\Delta_a s) + \Delta_a X \Delta_a S e \;=\; \Delta_a X \Delta_a S e,$$

as the third equation in the Newton system satisfies $S\Delta_a x + X\Delta_a s = -XSe$. The term $\Delta_a X \Delta_a S e$ corresponds to the error introduced by Newton's method in linearising the complementarity conditions of (2.3).

Ideally, we would like the next iterate to be perfectly centered:

$$(X + \Delta X)(S + \Delta S)e = \sigma\mu e,$$

which is equivalent to solving the nonlinear system

$$S\Delta x + X\Delta s = -XSe + \sigma\mu e - \Delta X \Delta S e. \tag{3.5}$$

By comparing (3.5) and (2.8), we see that the linearisation error made by the affine-scaling direction is exactly the $\Delta X \Delta S e$ term that is missing in the right-hand side of the last equation of (2.8).

Mehrotra introduced a second-order correction in which the linearisation error is taken into account. Therefore, Mehrotra's corrector term is obtained by solving the Newton system (2.8) with right-hand side

$$r = \begin{bmatrix} 0 \\ 0 \\ -\Delta_a X \Delta_a S e + \sigma\mu e \end{bmatrix}, \tag{3.6}$$

for the direction $\Delta_c w = (\Delta_c x, \Delta_c y, \Delta_c s)$. Such corrector direction combines the centrality term $\sigma\mu e$ and the second-order term $\Delta_a X \Delta_a S e$.

Once the predictor and corrector terms are computed, they are added to produce the composite predictor–corrector direction

$$\Delta w = \Delta_a w + \Delta_c w. \tag{3.7}$$

Note that the affine-scaling predictor and Mehrotra's corrector direction contribute with equal weights to the final search direction. This argument will be considered again in Chapter 4, where we study the use of a weighting strategy for the corrector directions.

The next iterate is given by

$$w^{k+1} = w^k + (\alpha_P \Delta x^k, \alpha_D \Delta y^k, \alpha_D \Delta s^k)$$

where α_P and α_D are chosen according to (3.1).

For reasons of computational efficiency, in the computation of the corrector, Mehrotra's algorithm exploits the same Jacobian matrix employed in finding the affine-scaling direction: hence, the same Cholesky factors are reused. The cost of a single iteration in the predictor–corrector method is only slightly larger than that of the standard method because two backsolves per iteration have to be executed, one for the predictor and one for the corrector.

The practical advantage of Mehrotra's predictor–corrector technique is that it often produces longer stepsizes before violating the non-negativity constraints. This usually translates into significant savings in the number of iterations: Mehrotra [50] reports on savings of the order of 35%-50% compared to other strategies. For problems for which the factorisation cost is relevant, this leads into significant CPU time savings [46, 50]. Indeed, Mehrotra's predictor–corrector technique is advantageous in all interior point implementations for linear programming which use direct methods to compute the Newton direction.

As mentioned above, Mehrotra's way of assessing the value of σ and the computation of the second order term is an heuristic procedure: thus there are no global convergence results or polynomial complexity results. Tapia et al. [63] interpreted the Newton step produced by Mehrotra's predictor–corrector algorithm as a perturbed composite Newton method and gave results on the order of convergence. They proved that a level-1 composite Newton method, when applied to the perturbed Karush-Kuhn-Tucker system, produces the same sequence of iterates as Mehrotra's predictor–corrector algorithm. While, in general, a level-m composite Newton method has a Q-convergence rate of $m + 2$ [58], the same result does not hold if the stepsize has to be damped to keep non-negativity of the iterates, as is necessary in an interior-point setting. However, under the additional assumptions of strict complementarity and nondegeneracy of the solution and feasibility of the starting point, Mehrotra's predictor–corrector method can be shown to have Q-cubic convergence [63].

3.3 Multiple centrality correctors

Mehrotra's predictor–corrector, as it is implemented in optimization solvers [46, 50], is a very aggressive technique. It is based on the assumption, rarely satisfied, that a *full* step in the corrected direction will be achievable. Moreover, an attempt to correct all complementarity products to the same value μ is also very demanding and occasionally counterproductive. Besides, practitioners noticed that this technique may sometimes behave erratically, especially when used for a predictor direction applied from highly infeasible and not well centered points.

Is there a reference for that?
Consider how Salahi et al. [60] modify the centrality term in the corrector.

Finally, Mehrotra's corrector does not provide CPU time savings when used recursively [8].

Trying to provide a remedy to the above considerations, Gondzio [20] introduced the multiple centrality corrector technique as an additional tool to complement those presented by Mehrotra. The idea behind this technique is to "force" an increase in the length of the stepsizes by correcting the centrality of Mehrotra's iterate. Instead of attempting to correct for the whole second-order error, multiple centrality correctors concentrate on improving the complementarity pairs which really seem to hinder the progress of the algorithm, ie. the complementarity products that are far from the average.

We assume that a long-step path-following algorithm is used, and we work with the symmetric neighbourhood $\mathcal{N}_s(\gamma)$ of the central path as defined in Section 2.3.

In this context, Mehrotra's predictor–corrector direction (3.7) is considered to be a new predictor direction Δ_p to which one or more centrality correctors can be applied. In the framework of multiple centrality correctors, we look for a centrality corrector Δ_m for which larger steps are allowed in the composite direction $\Delta = \Delta_p + \Delta_m$.

Assume that a predictor direction Δ_p is given and the corresponding feasible stepsizes α_P and α_D in the primal and dual spaces are determined. We want to enlarge the stepsizes to

$$\tilde{\alpha}_P = \min(\alpha_P + \delta,\, 1) \quad \text{and} \quad \tilde{\alpha}_D = \min(\alpha_D + \delta,\, 1),$$

for some fixed aspiration level $\delta \in (0, 1)$. We compute a trial point

$$\tilde{x} = x + \tilde{\alpha}_P \Delta_p x, \quad \tilde{s} = s + \tilde{\alpha}_D \Delta_p s,$$

and the corresponding complementarity products $\tilde{v} = \tilde{X}\tilde{S}e \in \mathbb{R}^n$. It is worth noting that this trial point is necessarily infeasible: this is not a drawback, as the trial point is used exclusively in determining a target for the centrality corrector.

The products $\tilde{v}$ are very unlikely to align to the same value μ. Some of them are significantly smaller than μ, including cases of negative components in $\tilde{v}$, and some exceed μ. Instead of trying to correct them all to the value of μ, we correct only those that lie outwith the symmetric neighbourhood. Namely, we try to move small products ($\tilde{x}_i\tilde{s}_i \le \gamma\mu$) to $\gamma\mu$ and move large products ($\tilde{x}_i\tilde{s}_i \ge \gamma^{-1}\mu$) to $\gamma^{-1}\mu$, where $\gamma \in (0, 1)$; complementarity products which satisfy $\gamma\mu \le x_i s_i \le \gamma^{-1}\mu$ are already reasonably close to their target values, and do not need to be changed. In other words, we attempt to move the iterate inside the symmetric neighbourhood of the central path.

The corrector term Δ_m is computed by solving the usual system of equations (2.8) for a special right-hand side $(0,\, 0,\, t)^T$, where the target t is defined as follows:

$$t_i = \begin{cases} \gamma\mu - \tilde{x}_i\tilde{s}_i & \text{if} \quad \tilde{x}_i\tilde{s}_i \le \gamma\mu \\ \gamma^{-1}\mu - \tilde{x}_i\tilde{s}_i & \text{if} \quad \tilde{x}_i\tilde{s}_i \ge \gamma^{-1}\mu \\ 0 & \text{otherwise.} \end{cases} \tag{3.8}$$

The target point is not on the central path, but in the symmetric neighbourhood.

One important feature of multiple centrality correctors technique is that it can be applied recursively on the direction $\Delta_p := \Delta_p + \Delta_m$. The maximum number of centrality correctors allowed is problem dependent. Such number is determined heuristically, trying to balance the cost of additional backsolves to the savings in iteration count. Each corrector computed is accepted as long as the stepsizes increase at least by a fraction of the aspiration level δ.

The computational experience presented in [20] showed that this strategy is effective, as the stepsizes in the primal and dual spaces computed for the composite direction are larger than those corresponding to the predictor direction. This leads to reductions in the number of iterations, which in turn translate into bigger CPU time savings the higher the factorization cost.

Virtually all existing interior point codes implement such technique [67, Appendix B]. Their use depends on the quality of the linear algebra codes.

3.4 Other considerations for implementations

In this section we mention a few more topics that are relevant to any practical implementation of interior point methods.

They concern the choice of the starting point, the termination criteria. This selection has been made according to the relevance of this thesis. Therefore, we will not discuss other topics of extreme importance in practical algorithms (presolve techniques, detection of infeasibilities, linear algebra issues, use of iterative methods). We refer the reader to the references in [1, 25].

3.4.1 Mehrotra's starting point heuristic

The choice of an initial iterate for interior point methods is a critical one. It challenges both the feasible and infeasible algorithms, and the solutions proposed in the two contexts are completely different.

Turning our attention to infeasible algorithms, the major hurdle of finding a feasible starting point is removed. However, the practical performance is very sensitive to the initial iterate, so the use of arbitrary points is not recommended. In particular, two requirements become important: the centrality of the point and the magnitude of the corresponding infeasibilities.

Expand!

Mehrotra [50] introduced a tool to find a starting point that attempts to fulfil the above requirements. In this heuristic, we solve two least squares problems which aim to satisfy the primal and dual constraints:

$$\begin{aligned} &\min_{x} \; x^T x \qquad \text{s.t.} \;\; Ax = b, \\ &\min_{y,s} \; s^T s \qquad \text{s.t.} \;\; A^T y + s = c. \end{aligned}$$

The corresponding solution $(\tilde{x}, \tilde{y}, \tilde{s})$ is further shifted inside the positive orthant, and the starting point is

$$(x_0, y_0, s_0) = (\tilde{x} + \delta_x e, \; \tilde{y}, \; \tilde{s} + \delta_s e),$$

where δ_x and δ_s are positive quantities. Their values depend on the distance of $\tilde{x}$ and $\tilde{s}$ to non-negativity, and an additional correction term to ensure strict positivity.

The following variation is described in [25]:

$$\begin{aligned} &\min_{x} \; c^T x + \rho x^T x \qquad \text{s.t.} \;\; Ax = b, \\ &\min_{y,s} \; b^T y + \rho s^T s \qquad \text{s.t.} \;\; A^T y + s = c, \end{aligned}$$

where the parameter ρ is fixed to a predetermined value, in order to compensate for the contribution of the primal and dual objectives.

A problem with Mehrotra's strategy is that it is scale dependent, it is affected by the presence of redundant constraints, and it does not guarantee to produce a well-centered iterate.

The procedure of finding an appropriate starting point will be discussed again in Chapter 5, where a specialised technique for the case of stochastic linear programming problems will be analysed.

3.4.2 Termination criteria

Contrary to active set algorithms, interior point methods reach a solution only asymptotically. Because of the presence of the barrier term that keeps the iterates away from the boundary, they can never produce an exact solution. When working in the context of limited precision of the floating point representation typical of computers, feasibility and complementarity can be attained only within a certain level of accuracy.

For these reasons, some criteria have to be established according to which a decision in made about the termination of the algorithm. The common termination criteria used in practice are the following [25]:

$$\frac{\|Ax-b\|}{1+\|x\|_\infty} \le 10^{-p}, \qquad \frac{\|A^Ty+s-c\|}{1+\|s\|_\infty} \le 10^{-p}, \qquad \frac{|c^Tx-b^Ty|}{1+|b^Ty|} \le 10^{-p}. \tag{3.9}$$

The value of p required depends on the specific application. In the literature, it is common to use the value $p=8$.

The third condition is usually the most important, as it is commonly attained only after the feasibility requirements are satisfied.

Dependence on scaling.

Once a solution has been found within a prescribed optimality tolerance, such a point can be projected onto a face of the polyhedron in an efficient way. This can be done thanks to a strongly polynomial algorithm due to Megiddo [49]. This procedure goes under the name of *basis crossover*, as given a complementary primal–dual solution it generates a basis that is both primal and dual feasible.

However, we should discuss what we mean by "exact" solution. In most cases we do not need the additional precision of being on a vertex (in this case, integer programming is the notable exception). Also, in case of multiple solutions, interior point methods terminate at the analytic center of the optimal face rather than at a vertex; but note the arbitrariness of the simplex in the choice of the solution vertex. This difference has important consequences on the use of the solution for sensitivity analysis.

Find the correct citations! Have a look at Jansen's and Yildirim's phds.

Chapter 4

Weighted corrector directions.

In this chapter we illustrate the theoretical understanding of the cases of failure of Mehrotra's corrector direction, drawing particularly from [9, 10]. We discuss subspace searches as a way to determine good search directions, and present a computationally competitive way to improve the performance of multiple centrality correctors. The original content of this chapter has already appeared in [12], co-authored with Jacek Gondzio.

4.1 Weaknesses in the existing approaches

Newton's method applied to the primal–dual path-following algorithm provides a first-order approximation of the central path, in which the nonlinear KKT system (2.5) corresponding to the barrier problem is linearised around the current point (x^k, y^k, s^k). Consistently with the standard analysis of Newton's method, this linear approximation is valid locally, in a small neighbourhood of the point where it is computed. Depending on the specific characteristics of the point, such an approximation may not be a good direction at all if used outside this neighbourhood.

Mehrotra's algorithm, as already discussed in Section 3.2, adds a second-order correction to the search direction in order to construct a quadratic approximation of the central path. This technique is at the basis of all implementations of interior point methods for linear programming, and the practical superiority of a second-order algorithm over a first-order one is broadly recognised. However, the central path is a highly nonlinear curve that, according to Vavasis and Ye [65], is composed by $O(n^2)$ turns of a high degree and segments in which it is approximately straight. Given the complexity of this curve, it is unrealistic to be able to approximate it everywhere with a second-order curve.

Failures of Mehrotra's corrector have been known by practitioners since its introduction. In practical implementations, it was noticed that Mehrotra's corrector would sporadically produce a shorter stepsize than the one obtained in the predictor direction. In such situations, it is common for a solver to reject the corrector, then try to use some multiple centrality correctors or move along the predictor direction alone.

Run Netlib problems with HOPDM, print Mehrotra rejected, and count them.

This issue has recently been analysed by Cartis [9], who provided an example in which the second-order corrector does not behave well. Cartis's analysis is based on an algorithm that combines a standard primal–dual path-following method with a second-order correction. Despite not being exactly Mehrotra's predictor–corrector algorithm, both are very close in spirit. Cartis's example shows that for certain starting points the corrector is always orders of magnitude larger than the predictor, in both primal and dual spaces. Whilst the predictor points towards the optimum, the second-order corrector points away from it. As the final direction is given by

$$\Delta = \Delta_a + \Delta_c,$$

the combined direction is influenced almost exclusively by the corrector, hence it is not accurate.

Where does this misbehaviour come from?
Coralia's starting point is badly centered. Maybe we could try other points and see if a better centred point fixes the issue of the large magnitude of the corrector.

The solution outlined by Cartis in [9], and then further developed in [10], is to reduce the influence exerted by the corrector by weighting it by the square of the stepsize.

Besides ensuring that the corrector is really considered a second-order term as in a rigorous Taylor expansion, this proposal allows for better convergence results.

Present some results.

In a similar way, Salahi et al. [60] propose to find the corrector by weighting the term $\Delta_a X \Delta_a S e$ in (3.6) by the allowed stepsize for the affine-scaling direction.

A better understanding of these issues and a solution to these problems will be the major focus of the next sections.

4.2 Subspace searches

Subspace searches are a different strategy of generating search directions. They are usually derived differently than from the Newton system, and are built according to some criteria that are believed to be essential for a good search direction.

4.2.1 Jarre and Wechs's directions

Jarre and Wechs [37] looked for an implementable technique for generating efficient higher-order search directions in a primal–dual interior-point framework. In the Newton system, while it is clear what to consider as right-hand side for primal and dual feasibility constraints (the residual at the current point), the complementarity component leaves more freedom in choosing a target t. They argue that there exists an optimal choice for which the corresponding Newton system would produce immediately the optimizer; however, it is not obvious how to find it. Therefore, they propose to search a subspace spanned by k different directions $\Delta w_1, \Delta w_2, \ldots, \Delta w_k$ generated from some affinely independent targets $t_1, t_2, \ldots, t_k$. As the quality of a search direction can be measured by the length of the stepsize and the reduction in complementarity gap, they aim to find the combination

$$\Delta w = \Delta w(\rho) = \rho_1 \Delta w_1 + \rho_2 \Delta w_2 + \ldots + \rho_k \Delta w_k$$

that maximizes these measures. This can be formulated as a small linear subproblem which can be solved approximately to produce a search direction Δw that is generally better than Mehrotra's predictor–corrector direction.

Present analysis from Jarre-Wechs [37].

The way they construct the search direction is the following. Given the system

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} b - Ax \\ c - A^T y - s \\ t \end{bmatrix}, \tag{4.1}$$

they ask what choice of right-hand side t would produce a good search direction.

Let $w = (x, y, s)$ be the current point and $w^* = (x^*, y^*, s^*)$ be an optimal primal–dual solution to the system (4.1). Therefore $\Delta w^* = w^* - w$ satisfies the system when $t = X\Delta s^* + S\Delta x^* = t^*$. Therefore, this particular choice of t^* proves to be the best overall, as it leads to the optimizer.

Obviously, when the system is set up, we have no knowledge of t^*. However, a lower bound can be determined. Knowing that $s^* + \Delta s^* \geq 0$ and $x^* + \Delta x^* \geq 0$, we have $\Delta s^* \geq -s^*$ and $\Delta x^* \geq -x^*$, from which:

$$t^* = X\Delta s^* + S\Delta x^* \geq -2XSe.$$

It would be interesting to see if in practice this bound is useful. We could check whether at any iteration it's violated, and if so just put the bound...

An upper bound is derived from a result by Vavasis and Ye (Lemma 16 in [65]):

"Let (x, y, s) and (x', y', s') be two points on the central path such that $0 < \mu' < \mu$. Then, for any i: $s'_i \leq ns_i$, and $x'_i \leq nx_i$."

From this, immediately we obtain that if (x, s) is on the central path, then

$$(\Delta x^*, \Delta s^*) \leq (n-1)(x, s).$$

Considering the final step of the proof in the cited Lemma:

$$\frac{x'_i}{x_i} + \frac{s'_i}{s_i} \leq n,$$

we can obtain the equivalent expression $s_i x'_i + x_i s'_i \leq n x_i s_i$, and so

$$SX'e + XS'e \leq nXSe.$$

Therefore, we can produce the following upper bound:

$$t^* = X\Delta s^* + S\Delta x^* = X(s^* - s) + S(x^* - x) = \underbrace{XS^*e + SX^*e}_{\leq nXSe} - 2XSe \leq (n-2)XSe.$$

It is possible to write t^* as a different expression. Given that $X^*S^*e = 0$, we have:

$$0 = (X + \Delta X^*)(S + \Delta S^*)e = XSe + \underbrace{X\Delta s^* + S\Delta x^*}_{=r^*} + \Delta X^* \Delta S^* e,$$

from which $t^* = -XSe - \Delta X^* \Delta S^* e$.

They consider how to use Mehrotra's corrector in an iterative fashion, and notice that a straightforward recursion of the type

$$\begin{aligned} A\Delta x^j &= b - Ax \\ A^T\Delta y^j + \Delta s^j &= c - A^T y - s \\ X\Delta s^j + S\Delta x^j &= \mu e - XSe - \Delta X^{j-1}\Delta S^{j-1}e \end{aligned}$$

is usually divergent for general positive starting points.

They noticed that following targets generated by iterating Mehrotra's corrector in a straightforward recursion of the type

$$t^{k+1} = \mu e - XSe - \Delta X^k \Delta S^k e$$

usually shows a divergent behaviour for general positive starting points. Therefore, they introduce a rescaling of Mehrotra's corrector and other strategies to guarantee good convergence properties.

4.2.2 Krylov subspace searches

While the multiple centrality correctors approach presented in Section 3.3 generates a series of correctors that are evaluated and applied recursively, Mehrotra and Li [51] propose a scheme in which a collection of linearly independent directions is combined through a small linear subproblem.

Following the approach explored by Jarre and Wechs [37], they express the requirements for a good search direction as a linear program. In particular, they impose conditions aimed at ensuring global convergence of the algorithm when using generic search directions. The directions considered in the subspace search can include all sorts of linearly independent directions: affine-scaling direction, Mehrotra's corrector, multiple centrality correctors, Jarre–Wechs directions. In their approach, Mehrotra and Li [51] generate directions using a Krylov subspace mechanism.

At the k-th iteration of interior point method we have to solve the Newton system $H_k\Delta_k = \xi_k$, where

$$\xi_k = \begin{bmatrix} b - Ax^k \\ c - A^T y^k - s^k \\ \mu e - X^k S^k e \end{bmatrix}$$

is the right-hand side evaluated at the current iterate and H_k is the corresponding Jacobian matrix. The direction Δ_k is used to compute a trial point:

$$\tilde{x} = x^k + \alpha_P \Delta_k x, \quad \tilde{y} = y^k + \alpha_D \Delta_k y, \quad \tilde{s} = s^k + \alpha_D \Delta_k s.$$

At the trial point $(\tilde{x}, \tilde{y}, \tilde{s})$, a usual interior point method would have to solve the system $\tilde{H}\tilde{\Delta} = \tilde{\xi}$ in order to find the next search direction. Instead, Mehrotra and Li [51] generate a Krylov subspace for $\tilde{H}\tilde{\Delta} = \tilde{\xi}$. The Krylov subspace of dimension j is defined as

$$K_j(H_k, \tilde{H}, \tilde{\xi}) = \text{span}\{\xi_H, G\xi_H, G^2\xi_H, \dots, G^{j-1}\xi_H\},$$

where $\xi_H = H_k^{-1}\tilde{\xi}$, and $G = I - H_k^{-1}\tilde{H}$. Note that for stability reasons $\tilde{H}$ is preconditioned with H_k, the factors of which have already been computed. The subspace thus generated contains j linearly independent directions.

In the algorithm of [51], the affine-scaling direction Δ_a, Mehrotra's corrector Δ_0, the first j directions $\Delta_1, \Delta_2, \ldots, \Delta_j$ from $K_j(H_k, \tilde{H}, \tilde{\xi})$ and, but only under some circumstances, a pure recentering direction Δ_{cen} are combined with appropriate weights ρ:

$$\Delta(\rho) = \rho_a \Delta_a + \sum_{i=0}^{j} \rho_i \Delta_i + \rho_{cen} \Delta_{cen}.$$

The choice of the best set of weights ρ in the combined search direction is obtained by solving an auxiliary linear programming subproblem. The subproblem maximizes the rate of decrease in duality gap whilst satisfying a series of requirements: non-negativity of the new iterate, upper bounds on the magnitude of the search direction, upper bounds on infeasibilities, decrease in the average complementarity gap, and closeness to the central path.

4.3 A practical implementation

The theoretical findings outlined above give rise to the following generalisation

$$\Delta = \Delta_a + \omega \Delta_c,$$

where we weight the corrector by a parameter $\omega \in (0, 1]$ independent of α.

In our implementation the weight is chosen independently at each iteration such that the stepsize in the composite direction is maximized. This gives us the freedom to find the optimal weight $\hat{\omega}$ in the interval $(0, 1]$. This generalisation allows for the possibility of using Mehrotra's corrector with a small weight, if that helps in producing a better stepsize; on the other hand, the choice $\hat{\omega} = 1$ yields Mehrotra's corrector again.

Discuss the difficulties related to the choice of ω.

4.3.1 Extension to multiple centrality correctors

We have applied the weighting strategy to multiple centrality correctors as well. The justification in this case comes from the following argument. In Section 3.3 we have seen that the target point in the multiple centrality correctors technique depends on the aspiration level δ which measures the greediness of the centrality corrector.

In the previous implementations, this parameter was fixed at coding time to a value determined after tuning to a series of representative test problems. However, for a specific problem such a value may be too conservative or too aggressive; moreover, the same value may not be optimal throughout the solution of the same problem. Hence, it makes sense to provide a mechanism that changes these correctors adaptively in order to increase their effectiveness.

Maybe present a table where for a subset of problems we show the the value of the stepsizes for a fixed δ and for an adaptive $\delta\omega$.

In our implementation we generate a sequence of multiple centrality correctors, and for each of them we choose the optimal weight $\hat{\omega}$ which maximizes the stepsizes in primal and dual spaces for a combined direction of the form

$$\Delta = \Delta_p + \omega \Delta_c.$$

The composite direction $\Delta = \Delta_p + \hat{\omega}\Delta_c$ becomes a predictor for the next centrality corrector, hence the correcting process is recursive, and can be interrupted at any stage.

Below we formalise the weighted correctors algorithm.

```
Given an initial iterate $(x^0, y^0, s^0)$ such that $(x^0, s^0) > 0$;

Repeat for $k = 0, 1, 2, \dots$ until some convergence criteria are met:

    ∘ Solve system (2.8) with right-hand side $r_1$ (3.2) for a predictor direction $\Delta_a$.
    ∘ Set $\mu$ according to (3.4) and find Mehrotra's corrector direction $\Delta_c$ by solving
      system (2.8) with right-hand side (3.6).
    ∘ Do a linesearch to find the optimal $\hat{\omega}$ that maximizes the stepsize $\alpha$ in $\Delta^\omega =$
      $\Delta_a + \omega\Delta_c$.
    ∘ Set $\Delta_p = \Delta_a + \hat{\omega}\Delta_c$.
      Do   – Solve system (2.8) with right-hand side $(0, 0, t)$, $t$ given by (3.8) for a
             centrality corrector direction $\Delta_m$.
           – Perform a linesearch to find the optimal $\hat{\omega}$ that maximizes the stepsize
             $\alpha$ in $\Delta^\omega = \Delta_p + \omega\Delta_m$.
           – Set $\Delta_p := \Delta_p + \hat{\omega}\Delta_m$.
      While the stepsize increases by at least a fraction of the aspiration level $\delta$;
    ∘ Update the iterate $(x^{k+1}, y^{k+1}, s^{k+1}) = (x^k, y^k, s^k) + \alpha_k\Delta_p(x^k, y^k, s^k)$.

End
```

4.3.2 Finding an appropriate weight

Concerning the choice of ω, we implemented a linesearch in the interval $[\omega_{\min}, \omega_{\max}] = [\alpha_P\alpha_D, 1]$. There are two reasons for using $\omega_{\min} = \alpha_P\alpha_D$. First, using the stepsizes α_P and α_D for the predictor direction gives

$$(X + \alpha_P\Delta X)(S + \alpha_D\Delta S)e \;=\; XSe + \alpha_P S\Delta Xe + \alpha_D X\Delta Se + \alpha_P\alpha_D\Delta X\Delta Se,$$

and the term $\alpha_P\alpha_D$ appears with the second-order error. Secondly, the study of Cartis [9] suggests squaring the stepsize for the corrector. Our computational experience indicates that the straight use of $\omega = \omega_{\min} = \alpha_P\alpha_D$ is too conservative. Still, such $\omega_{\min}$ is a reliable lower bound for attractive weights ω.

Show a table or a figure for this.

The ultimate objective in choosing ω is to increase the stepsizes α_P and α_D. These stepsizes depend on ω in a complex way. Examples corresponding to a common behaviour are given in Figure 4.1, where we show how the product $\alpha_P\alpha_D$ varies depending on the choice of ω for Mehrotra's corrector at two different iterations of problem `capri` of the Netlib collection. On the left, $\omega \in [0.4, 1]$ and $\hat{\omega} = 0.475$ gives a product $\alpha_P\alpha_D = 0.583$, better than a value of 0.477 that would have been obtained by using a full weight on Mehrotra's corrector. On the right, $\omega \in [0.178, 1]$ and the choice of $\omega \in (0.6, 0.7)$ leads to the best product $\alpha_P\alpha_D$ of about 0.375.

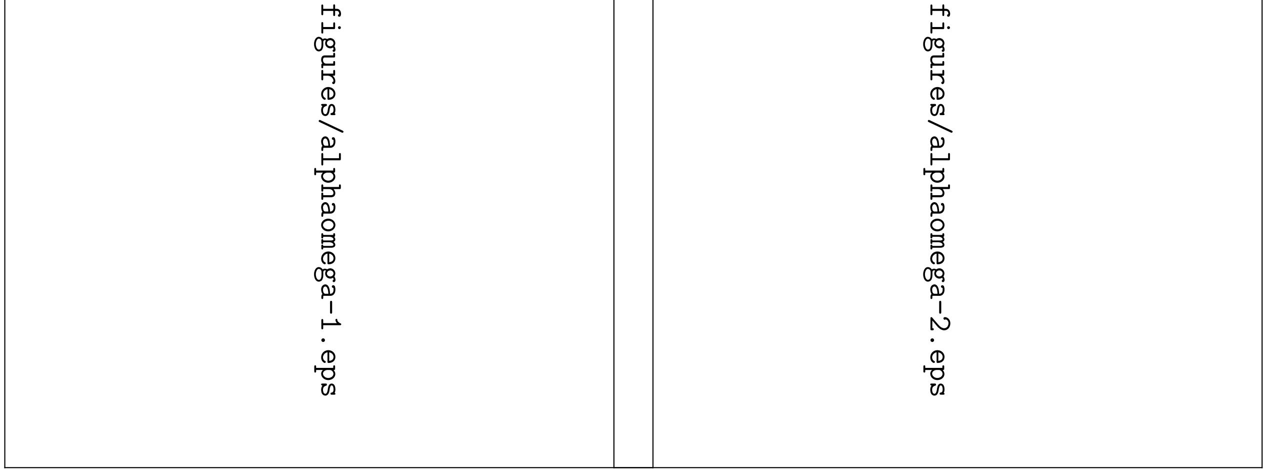

Figure 4.1: Relationship between ω and $\alpha_P\alpha_D$ in two iterations of problem `capri`.

In our crude linesearch procedure we choose 9 points uniformly distributed in the interval $[\alpha_P\alpha_D, 1]$ and evaluate, for each of these points, the stepsizes in both spaces. When a larger stepsize α_P or α_D is obtained, the corresponding ω is stored as ω_P or ω_D respectively. Hence, we allow two different weightings for directions in the primal and dual spaces.

4.4 Numerical results

We have implemented our proposal within the HOPDM interior point solver [19]. We tested our implementation in a series of computational experiments, using test problems from different collections. As a comparison, we present the results obtained by PCx [14], a reference implementation of interior point methods. Since the two implementations, PCx and HOPDM, have different termination criteria in their default configurations, for the purpose of consistency we decided to implement in HOPDM the criteria used in the study performed by Mehrotra and Li [51]. Therefore, optimal termination occurs when the following conditions are met:

$$\frac{\mu}{1+|c^Tx|} \leq 10^{-10}, \qquad \frac{\|b-Ax\|}{1+\|b\|} \leq 10^{-8}, \qquad \frac{\|c-A^Ty-s\|}{1+\|c\|} \leq 10^{-8}. \tag{4.2}$$

Criticize these criteria.

We use $\gamma = 0.1$ in the definition of symmetric neighbourhood and define aspiration levels for the stepsizes using the rule

$$\tilde{\alpha}_P = \min(1.5\alpha_P{+}0.3,\ 1) \quad \text{and} \quad \tilde{\alpha}_D = \min(1.5\alpha_D{+}0.3,\ 1).$$

The values suggested in [20] were more conservative: $\tilde{\alpha} = \min(\alpha + 0.1,\ 1)$. Thanks to the weighting mechanism we can control the contribution of the corrector in an adaptive way, and thus be more demanding in the definition of the aspired stepsizes.

Centrality correctors are accepted in the primal and/or dual space if $\alpha_P^{new} \geq 1.01\alpha_P$ and/or $\alpha_D^{new} \geq 1.01\alpha_D$, respectively.

Note that this is different from what described in [20]: there, a corrector is accepted if $\alpha^{new} \geq \alpha + \gamma\delta_\alpha$, where $\gamma = \delta_\alpha = 0.1$. What did HOPDM use to do?

We present our results in terms of number of iterations and number of backsolve operations. The rationale behind this decision is that the multiple centrality correctors technique determines the number of allowed correctors on the basis of the ratio between factorization cost and backsolve cost. This ratio can be very different across implementations, and is mainly influenced by the linear algebra routines used. HOPDM comes with an in-house linear algebra implementation, while PCx relies on the more sophisticated sparse Cholesky solver of Ng and Peyton. Therefore, the PCx code tends to use less correctors per iteration.

4.4.1 Mehrotra-Li test collection

First we considered the test set used in [51]: it contains 101 problems from both Netlib and Kennington collections. In Table 4.1 we present the computational comparison outlining the total number of iterations and the total number of backsolves necessary to solve the problems in Mehrotra and Li's test set. Column HO displays the results obtained by the previous implementation, while column dHO reports the results obtained by the current implementation of weighted correctors with a default choice of the number of centrality correctors. The last column presents the relative change between the two versions of HOPDM tested. As a reference, we also report in this table the overall statistics of PCx (release 1.1) on these problems. Also for PCx we adopted the termination criteria (4.2). We found the number of backsolves by counting the number of calls to the functions `IRSOLV()` and `EnhanceSolve()`, for HOPDM and PCx respectively.

	PCx	HO	dHO	Change
Iterations	2114	1871	1445	-22.77%
Backsolves	4849	6043	5717	-5.39%
Backsolves/iter.	2.29	3.23	3.95	+22.29%

Table 4.1: Overall results obtained on Mehrotra and Li's test collection.

From Table 4.1 we first observe the very small number of backsolves per iteration needed by PCx. This is due to the fact that PCx allows the use of Gondzio's multiple centrality correctors only in 4 problems: `dfl001`, `maros-r7`, `pds-10` and `pds-20`. Also we notice that when we allow an adaptive weighting of the correctors there is a tendency to use more correctors per iteration than previously. This happens because the weighting mechanism makes it more likely to accept some correctors that otherwise would have been rejected as too aggressive. While this usually leads to a decrease in iteration count, it also makes each iteration more expensive.

In Table 4.2 we detail the problems for which we obtained savings in computational time. Given the small dimension of most of the problems in the Netlib collection, we did not expect major savings. However, as the problem sizes increase, we can see that the proposed way of evaluating and weighting the correctors pays off. This led us to investigate further the performance of the proposed implementation, which we will discuss in Section 4.4.2.

We were particularly interested in comparing the results produced by our weighted correctors approach with those published in [51]. The computation of Krylov subspace directions in Mehrotra and Li's approach does involve considerable computational cost, as the computation of each Krylov direction requires a backsolve operation. This can be

Problem	HO	dHO
bnl1	0.36	0.25
dfl001	150.63	114.80
maros-r7	7.76	7.52
pilot	5.23	4.35
pilot87	12.62	11.88
pds-06	24.59	21.31
pds-10	96.57	79.29
pds-20	923.71	633.64

Table 4.2: Problems that showed time savings (times are in seconds).

seen from the definition of the power basis matrix

$$G = I - H_k^{-1}\tilde{H},$$

which involves an inverse matrix. In fact, calling u the starting vector in the Krylov sequence, the computation of the vector $H_k^{-1}\tilde{H}u$ requires first to compute $v = \tilde{H}u$ (matrix–vector multiplication) and then to determine $t = H_k^{-1}v$ (backsolve on the Cholesky factors).

In the tables of results presented in [51], the best performance in terms of iteration count is obtained by PCx4, which uses 4 Krylov subspace vectors. These directions are combined with an affine-scaling predictor direction and Mehrotra's second-order correction, leading to 6 backsolves per iteration. This number increases when the linear subproblem produces an optimal objective value smaller than a specified threshold or the new iterate fails to satisfy some neighbourhood condition: in these cases the pure centering direction Δ_{cen} also needs to be computed, and a seventh backsolve is performed.

As we understand the paper [51], PCx0 uses exactly 2 backsolves per iteration: one to compute the affine-scaling direction, another to compute Mehrotra's corrector; PCx2 and PCx4 compute two and four additional Krylov vectors, hence they use 4 and 6 backsolves per iteration, respectively. In columns HO-0, HO-2 and HO-4, we present the results obtained by HOPDM when forcing the use of 0, 2 and 4 multiple centrality correctors. In the column called HO-∞ we report the results obtained when an unlimited number of correctors is allowed (in practice we allow no more than 20 correctors). The last column, labelled dHO, presents the results obtained by the default way of choosing the number of correctors allowed.

Consequently, up to 2, 4 and 6 backsolves per iteration are allowed in PCx0, PCx2 and PCx4 and in HO-0, HO-2 and HO-4 runs, respectively. The number of backsolves reported for HOPDM includes two needed by the initialisation procedure: the number of backsolves should not exceed $2 \times \text{Its} + 2$, $4 \times \text{Its} + 2$ and $6 \times \text{Its} + 2$ respectively for HO-0, H0-2 and H0-4. The observed number of backsolves is often much smaller than these bounds because the correcting mechanism switches off when the stepsizes are equal to 1 or when the corrector does not improve the stepsize. Problem `afiro` solved by HO-4 needs 24 backsolves, 22 of which compute different components of directions, hence the average number of backsolves per iteration is only 22/6 and is much smaller than 6. Occasionally, as a consequence of numerical errors, certain components of direction are rejected on the grounds of insufficient accuracy: in such case the number of backsolves may exceed the stated upper bounds. The reader may observe for example that `pilot4` is solved by HO-4 in 16 iterations, but the number of backsolves is equal to 100 and exceeds $6 \times 16 + 2 = 98$.

The version HO-∞ requires 1139 iterations to solve the collection of 101 problems, an average of just above 11 iterations per problem. This version has only an academic interest, yet it reveals a spectacular efficiency of interior point methods which can solve difficult linear programs of medium sizes (reaching a couple of hundred thousand variables) in just a few iterations. In particular, it suggests that if we had a cheap way of generating search directions, then it would be beneficial to have as many as possible.

4.4.2 Beyond Netlib

We have applied our algorithm to examples from other test collections besides Netlib. These include other medium to large linear programming problems, stochastic problems and quadratic programming problems.

We used a collection of medium to large problems taken from different sources: problems `CH` through `CO9` are MARKAL (Market Allocation) models; `mod2` through `worldl` are agricultural models used earlier in [20]; problems `route` through `rlfdual` can be retrieved from

`http://www.sztaki.hu/~meszaros/public_ftp/lptestset/misc/`,

problems `neos1` through `fome13` can be retrieved from

`ftp://plato.asu.edu/pub/lptestset/`.

In Table 4.3 we detail the sizes of these problems and provide a time comparison between our previous implementation (shown in column HO), and the current one (in column dHO). This test collection contains problems large enough to show a consistent improvement in CPU time: in only 4 problems (`mod2`, `dbc1`, `watson-1`, `sgpf5y6`) we recorded a deterioration of the performance by more than 1 second. The improvements are significant on problems with a large number of nonzero elements. In these cases, dHO produces savings from about 10% to 30%, with the remarkable results in `rail2586` and `rail4284`, for which the relative savings reach 45% and 65%, respectively.

More numerical results obtained on some collections of stochastic programming problems and quadratic problems can be found in [12].

Problem	Rows	Cols	Nonzeros	HO	dHO	Change
CH	3852	5062	42910	1.03	1.23	19.4%
GE	10339	11098	53763	5.72	5.46	-4.5%
NL	7195	9718	102570	4.37	3.95	-9.6%
BL	6325	8018	58607	2.15	2.14	-0.5%
BL2	6325	8018	58607	2.35	2.31	-1.7%
UK	10131	14212	128341	2.48	3.21	29.4%
CQ5	5149	7530	83564	2.54	2.60	2.4%
CQ9	9451	13778	157598	9.67	8.84	-8.6%
CO5	5878	7993	92788	3.16	3.59	13.6%
CO9	10965	14851	176346	21.10	15.35	-27.3%
mod2	35664	31728	220116	20.59	21.68	5.3%
world	35510	32734	220748	26.35	23.41	-11.2%
world3	36266	33675	224959	31.13	27.49	-11.7%
world4	52996	37557	324502	73.21	56.14	-23.3%
world6	47038	32670	279024	39.33	32.79	-16.6%
world7	54807	37582	333509	43.14	36.02	-16.5%
worldl	49108	33345	291942	43.95	36.82	-16.2%
route	20894	23923	210025	40.92	33.78	-17.4%

Table 4.3: Time comparison on other large problems (times are in seconds).

Problem	Rows	Cols	Nonzeros	HO	dHO	Change
ulevi	6590	44605	162207	9.04	9.55	5.6%
ulevimin	6590	44605	162207	16.52	16.46	-0.4%
dbir1	18804	27355	1067815	162.18	146.51	-9.7%
dbir2	18906	27355	1148847	208.93	156.11	-25.3%
dbic1	43200	183235	1217046	72.96	77.31	5.9%
pcb1000	1565	2428	22499	0.26	0.33	26.9%
pcb3000	3960	6810	63367	1.13	1.16	2.7%
rlfprim	58866	8052	265975	15.63	15.08	-3.5%
rlfdual	8052	66918	328891	71.17	49.79	-30.0%
neos1	131581	1892	468094	169.11	141.89	-16.1%
neos2	132568	1560	552596	113.86	86.13	-24.4%
neos3	132568	1560	552596	132.02	120.59	-8.7%
neos	479119	36786	1084461	1785.80	1386.58	-22.4%
watson-1	201155	383927	1053564	138.60	166.21	19.9%
sgpf5y6	246077	308634	902275	49.58	64.45	30.0%
stormG2-1000	528185	1259121	4228817	1661.54	1623.19	-2.3%
rail507	507	63009	472358	9.77	10.10	3.4%
rail516	516	47311	362207	7.59	5.89	-22.4%
rail582	582	55515	457223	9.67	9.60	-0.7%
rail2586	2586	920683	8929459	1029.36	566.82	-44.9%
rail4284	4284	1092610	12372358	2779.63	978.48	-64.8%
fome11	12142	24460	83746	407.20	265.21	-34.9%
fome12	24284	48920	167492	766.96	508.61	-33.7%
fome13	48568	97840	334984	1545.05	990.62	-35.9%

Table 4.3: Time comparison on other large problems (times are in seconds).

4.4.3 Conclusions

In this paper we have revisited the technique of multiple centrality correctors [20] and added a new degree of freedom to it. Instead of computing the corrected direction from $\Delta = \Delta_p + \Delta_c$ where Δ_p and Δ_c are the predictor and corrector terms, we allow a choice of weight $\omega \in (0, 1]$ for the corrector term and compute $\Delta = \Delta_p + \omega\Delta_c$. We combined this modification with the use of a symmetric neighbourhood of the central path. We have shown that the use of this neighbourhood does not cause any loss in the worst-case complexity of the algorithm.

The computational results presented for different classes of problems demonstrate the potential of the new scheme. We have compared our algorithm against the recently introduced Krylov subspace scheme [51]. The two approaches have similarities: they look for a set of attractive independent terms from which the final direction is constructed. Mehrotra and Li's approach uses the first few elements from the basis of the Krylov space; our approach generates direction terms using centrality correctors of [20]. Mehrotra and Li's approach solves an auxiliary linear program to find an optimal combination of all available direction terms; our approach repeatedly chooses the best weight for each newly constructed corrector term (and switches off if the use of the corrector does not offer sufficient improvement). Eventually, after adding k corrector terms, the directions used in our approach have form

$$\Delta = \Delta_a + \omega_1\Delta_1 + \ldots + \omega_k\Delta_k,$$

and the affine-scaling term Δ_a contributes to it without any reduction. Hence, the larger the stepsize, the more progress we make towards the optimizer.

The comparison presented in Section 4.4.1 shows a clear advantage of our scheme over that of [51]. Indeed, with the same number of direction terms allowed, our scheme outperforms Krylov subspace correctors by a wide margin. Multiple centrality correctors show consistent excellent performance on other classes of problems including medium to large scale linear programs beyond the Netlib collection and medium scale quadratic programs.

Monteiro, Adler and Resende (citation 188 in Jansen [35]) talk about corrector steps.

It would be good to try and implement using the search direction from the previous iteration as another (free) direction to span the subspace.

Remark 4.1. From the study of subspace searches it is clear that the more directions we consider, the better the final search direction we get. Therefore, if we had a cheap way of generating search directions (rather than from solving a system of linear equations), then these should be employed.

Remark 4.2. Mehrotra and Li [51] mention employing previous search directions alongside the usual ones. The use of these incurs an increased memory usage, but no additional computational cost. However, it does not seem that they were actually employed in the runs. This opens some questions on what constitutes a valid previous direction (only affine scaling, the final composite direction or something else).

Chapter 5

Warm-start strategies for stochastic linear programs.

Stochastic programming [5, 40] is a technique to inform decision-making in many areas of applied mathematics, engineering, economics and finance where some parameters are unknown. Stochastic programming models uncertainty through the analysis of possible future outcomes (scenarios). We hope that the more detailed the description is, the more robust are the decisions taken.

Taking into account a large number of possible scenarios leads to the generation of large-scale structured optimization problems. With the growing industrial acknowledgement of the benefits of considering uncertainty for planning purposes, it is expected that the need for solving very large instances will grow as well. As the dimensions of the problems increase, the computational advantages of relying on interior point solvers become more and more evident.

In this chapter we develop an efficient way of constructing a starting point for structured large-scale stochastic linear programs.

This chapter is designed as follows. We first present the relevant concepts of stochastic programming and introduce a measure of the distance between scenarios. Then we review some studies on warm-start techniques for interior point methods. Then we introduce the proposed method of generating a good starting point for stochastic programming that exploits the inherent structure of the problem. Finally, we present an analysis of the warm-start strategy.

The results presented in this chapter have been the subject of a joint work with Jacek Gondzio and Andreas Grothey [13].

5.1 Stochastic programming

By stochastic programming, we mean decision and control models in which data evolves over time, and are subject to significant uncertainty. Uncertainty in the data is a commonly observed phenomenon in optimization problems coming from applications. Uncertainty affects problems that aim to plan future actions based on forecasted prices or costs. In can be argued that nearly all practical optimization problems display uncertainty in the data, even if this is not made explicit in the chosen solution method.

A crude approach to the problem of optimization under uncertainty has been to replace the uncertain data by their expected value and solve an *average case* problem. However, this approach is not suitable when some sort of provision of hedging against risk has to be taken into account. Another popular approach is *Robust optimization*, or the optimization of the *worst case* scenario [40].

When the uncertainty cannot be conveniently forecasted, the use of deterministic models is considered inadequate for decision making. In these situations, being able to describe and model the uncertain parameters becomes a requirement for robust decision making. Stochastic programming [5, 40] is the discipline that studies the methods and provides the tools for modelling uncertainty.

Stochastic programming aims to take all possible future scenarios into account, weighting them with their respective probabilities. Its strength lies in the adaptability which allows to express preferences such as restricting the exposure to risk. Unlike alternative approaches, it allows to model situations in which possible future events are correlated or follow a time-structure, in that realisations become known in stages and it is possible to react to the observed events.

The stages do not necessarily refer to time periods, they correspond to steps in the decision process. The main interest lies in the first-stage decisions which consist of all decisions that have to be made before the information is revealed. Later-stage decisions are allowed to adapt to the information gathered up to that point.

Its perceived weaknesses are the need for reliable forecasts of the probabilities of the future events under consideration (which may not be available), and the fact that stochastic programming (especially when applied to multi-stage models) tends to lead to problems with very large dimensions, thus making their solution challenging.

5.1.1 Stochastic programming concepts

A stochastic programming problem incorporates the uncertain parameters in the model, as can be illustrated by the problem

$$\min_x \mathbb{E}_\xi(f(x,\xi)) \text{ subject to } g(x,\xi) \le 0, \tag{5.1}$$

where ξ is a random variable and $\mathbb{E}_\xi$ is the expectation function. This can be interpreted as an optimization problem in which some parameters or coefficients are unknown. Formulation (5.1) is however an unsatisfactory model, since it is unclear how to interpret the *probabilistic* constraint $g(x,\xi) \le 0$. A better formulation is the stochastic program with recourse:

$$\begin{aligned} &\min_x \mathbb{E}_\xi[f(x,\xi) + Q(x,\xi)] \\ &\qquad Q(x,\xi) = \min_y \{\, q(y) : u_i(y) \ge g_i^+(x,\xi),\ \forall i \,\}, \\ &\qquad g_i^+(x,\xi) = \max\{\, g_i(x,\xi),\ 0 \,\}. \end{aligned} \tag{5.2}$$

Explain what the recourse is.

In stochastic programming, the uncertain environment is described through a stochastic process which is assumed to be known or can be either estimated from historical data or conjectured according to some prescribed properties. The continuous process ξ is usually further approximated by a discrete distribution, $\xi \in \{\xi_1, \ldots, \xi_n\}$, $p(\xi = \xi_i) = p_i$, in order to obtain a computationally amenable description. In such a case, the most common technique generates a finite, but usually very large, number of scenarios that represent an approximate description of the possible outcomes.

The model can be generalised to a multi-stage model in which the evolution of uncertainties can be described as an alternating sequence of decisions and random realisations that occur at different points in time (stages). The discrete stochastic process can be represented as an event tree, where each node denotes a stage when a realisation of the random process becomes known and a subsequent decision is taken. A path from the root to a leaf node of the event tree represents a scenario. A very simple event tree is shown in Figure 5.1.

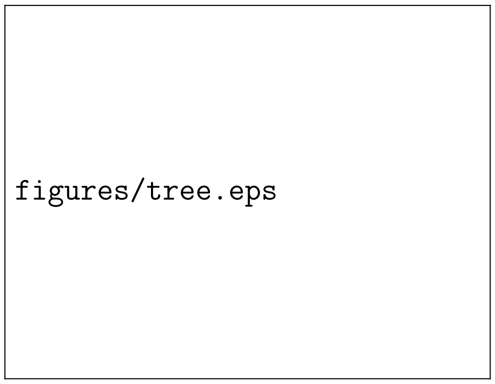

Figure 5.1: Event tree.

To each node of the event tree we associate a set of constraints, an objective function, and the conditional probability of visiting the node from its parent node in the previous stage. The probability of each scenario is the product of the conditional probabilities of visiting each node on the path.

The question of how to generate an appropriate scenario tree is not trivial, and has received extensive attention in the literature [17, 32, 33, 59].

It should be noted that usually a very large number of scenarios is needed to adequately capture the characteristics of the underlying continuous distribution, particularly in the multi-stage setting. In this case, the number of scenarios grows exponentially with the number of decisions considered at each stage.
The size of the scenario tree determines the size of the optimization program.

5.1.2 Deterministic equivalent formulation for stochastic programs

A natural formulation of a stochastic programming problem relies on recursion to describe the dynamics of the modelled process. The term *recourse* means that, at each stage, the decision variables adapt to the different outcomes of the random parameters.

A multistage stochastic program with recourse is a multi-period mathematical program where parameters are assumed to be uncertain along the time path.

For ease of presentation, we consider the linear version of the general recourse problem (5.2):

$$\begin{array}{ll} \min & c^T x + I\!\!E_\xi[\min q(y)^T y(\xi)] \\ \text{s.t.} & Ax \quad = \quad b, \\ & T(\xi)x + W(\xi)y(\xi) \quad = \quad h(\xi), \\ & x \geq 0,\ y(\xi) \geq 0, \end{array} \tag{5.3}$$

where $y(\xi)$ denotes the recourse action which depends on the outcome of the random process ξ. After discretizing ξ according to $P(x_i = \xi_i) = p_i$, and using the notation $T_i = T(\xi_i)$ (and analogously for h_i, W_i, y_i and q_i), problem (5.3) can be written in the *deterministic equivalent formulation*:

$$\begin{array}{llccl} \min & c^T x + \sum_i p_i q_i^T y_i & & & \\ \text{s.t.} & Ax & & & = b, \\ & T_1 x + W_1 y_1 & & & = h_1 \\ & \vdots & \ddots & & \vdots \\ & T_n x & & + W_n y_n & = h_n \\ & x \geq 0,\ y_i \geq 0. & & & \end{array} \tag{5.4}$$

Problem (5.4) displays dual-block angular structure in the constraint matrix.

Explain why deterministic.

To formulate the deterministic equivalent of the multi-stage stochastic programming problem we first need to enumerate all nodes of the event tree. We use a breadth-first search order, i.e., we start from a root node corresponding to the initial stage and end with leaf nodes corresponding to the last stage. In this case, the constrint matrix displays a nested dual-block angular structure.

Let $t = 1, 2, \ldots, T$ denote the stage and l_t be the index of a node at stage t. Let L_t denote the last node at stage t. Hence the nodes that belong to stage $t > 1$ have indices $l_t = L_{t-1}+1, L_{t-1}+2, \ldots, L_t$. With $a(l_t)$ we denote the direct ancestor of node l_t (which is a node that belongs to stage $t-1$).

For any node in the tree at stage t there is only one immediate ancestor at stage $t-1$ and a finite number of descendants at $t+1$.

All decision variables x are superscripted with the node number l_t.

The main constraint that describes the dynamics of the system has the form

$$T^{l_t} x^{a(l_t)} + W^{l_t} x^{l_t} = h^{l_t}, \qquad l_t = L_{t-1}+1, \ldots, L_t,$$

where T^{l_t} is the technology matrix that varies with the node in the event tree, and W^{l_t} is the recourse matrix that, in general, depends on realisations within the same stage, but often varies only with time.

The deterministic equivalent formulation of the multi-stage problem has the following general form:

$$
\begin{array}{llllll}
\min & (q^{l_1})^T x^{l_1} & + \sum\limits_{l_2=L_1+1}^{L_2} p^{l_2}(q^{l_2})^T x^{l_2} & + \ldots + & \sum\limits_{l_T=L_{T-1}+1}^{L_T} p^{l_T}(q^{l_T})^T x^{l_T} & \\
\text{s.t.} & & W^{l_1} x^{l_1} = h^{l_1}, & & & \\
& T^{l_2} x^1 & + \; W^{l_2} x^{l_2} = h^{l_2}, & & l_2 = L_1+1, \ldots, L_2, & \\
& \vdots & \vdots & & & \\
& T^{l_T} x^{a(l_T)} & + \; W^{l_T} x^{l_T} = h^{l_T}, & & l_T = L_{T-1}+1, \ldots, L_T, & \\
& & x^{l_t} \geq 0, & & l_t = 1, \ldots, L_T. &
\end{array}
\tag{5.5}
$$

Note that the probabilities in the objective function of problem (5.5) are the unconditional path probabilities: p^n is the probability that a path goes through node n, which equals the product of the conditional probabilities along the path from the root to the node.

If the event tree is traversed with depth-first-search ordering of the nodes during the generation of the mathematical program, the corresponding constraint matrix displays a nested dual block-angular structure. While the different ordering of blocks whithin the matrix is not relevant for general-purpose solvers, the structure-exploiting software OOPS [24] can take fully advantage of the nested structure.

Kall and Mayer [39] provided a comparison of different solution algorithms for stochastic linear programming problems.

5.2 Warm-start with interior point methods

Consider again the linear programming problem in standard form

$$\min \; c^T x \quad \text{s.t.} \;\; Ax = b, \;\; x \geq 0,$$

where $A \in \mathbb{R}^{m \times n}$ is full rank, $x, c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. As discussed in Section 3.4.1, the problem of finding a starting point is usually solved by using Mehrotra's starting point heuristic [50], which is considered to be computationally effective for generic problems.

However, many practical applications rely on solving a sequence of closely related problems, where the instances differ by some perturbation. This happens within algorithms that are sequential in their nature, such as sequential linear programming or sequential quadratic programming. It is also very common in (mixed) integer programming, when the problems are solved by relaxing the integrality constraints and introducing some branching strategy, such as in branch-and-bound, branch-and-cut, branch-and-price. Similarly, sequential problems appear in the context of solution methods based on cutting planes.

In these situations, we expect the solution of an instance to be close to the solution of the next one, in the sense that some of the information gathered in the solution process may still be valid. Therefore, (warm) starting the optimization of one problem from the solution of the previous one should reduce the computational effort of solving the perturbed instance.

Warm-start techniques are very successful when implemented with a simplex solver (see, for example, [7]).

More on the simplex.

However, in the context of interior point methods, it is much more difficult to implement successfully, for the reasons we outline below.

5.2.1 Difficulties

An optimal solution of a linear programming problem found with interior point methods is very close to the optimal vertex of the feasible polytope. With changes in the polytope (e.g. because of the addition of cutting planes or other changes in the problem data), the optimal solution changes as well. In such a case the previous solution is very close to a non optimal vertex or, in other words, very far from the central path.

The effectiveness of an interior point algorithm degrades when an iterate gets too close to a boundary before optimality is reached, and when the iterate gets far from the central path.

> As said in Chapter 2, interior point methods approach the solution to the KKT system of optimality conditions by relaxing the complementarity requirements. This is equivalent to postponing the choice of the optimal partitioning.
> Therefore, moving toward a vertex can be interpreted as making a decision on the optimal partitioning. If the vertex is not an optimal one, then the central path will not approach it.
> Therefore the algorithm may spend many iterations in recovering centrality, during which small steps are usually generated and thus very slow progress is achieved.

Hipolito [31] considered the issue of robustness of search directions in interior point methods in these situations. In his analysis, Hipolito showed that if the iterate is close to a boundary, the affine-scaling direction may be parallel to the nearby constraints. In such cases, the corrector direction may also display the same behaviour, and short stepsizes are obtained for the resulting search direction.

The required features for a good warm-start candidate for an interior point algorithm are somewhat contradictory. The point should not be too close to the boundary of the feasible region in order to be able to absorb larger perturbations in the problem data. However, it should be sufficiently advanced to provide computational savings over a cold-start iterate. A common way of accommodating these requirements is to store an approximate μ-center well before reaching optimality [21, 22, 26, 68].

The theory and practice of warm-start techniques for interior point methods is a relatively new and still open field of study. In the remainder of this section we present a review of some of the warm-start approaches proposed in the interior point literature.

5.2.2 Changes in the problem size

Mitchell [52] and Mitchell and Todd [53] analyse the potential reduction interior point method within a cutting plane algorithm. They exploit the fact that the primal feasible point can be constructed after a set of new columns is added to the problem. They use this strategy with success in column generation scheme and more generally in the solution of combinatorial optimization problems.

Gondzio [21] presents a warm-start procedure for primal–dual interior point methods in the context of a cutting plane method. The interior point method is used to solve a sequence of restricted master problems, which differ by one or more cutting planes. The idea proposed in [21] is to store a nearly optimal point (3–4 digits of accuracy) to be employed as a warm-start point. As one requirement for a good iterate is centrality, it is of interest to perform a few centering steps on the stored iterate, the cost of which is marginal, as a factorisation is already available. The recentering steps proposed are based on centrality correctors [20]. An auxiliary feasibility recovery procedure may be needed as, due to the addition of cuts, large infeasibilities are often produced.

5.2.3 Perturbations that do not affect the problem size

Hipolito [31] studies an alternative centering direction in the context of the dual affine-scaling algorithm. Such a direction is designed to move the iterate away from the boundary, overcoming the risk of moving parallel to it that was mentioned in Section 5.2.1. By considering a weighted least squares formulation, Hipolito develops the dual affine-scaling and the corresponding Newton centering directions. This study has an immediate interest for warm-starting approaches, as the resulting direction points towards the interior of the feasible region, thus providing the algorithm with the necessary space to make fast progress. Unfortunately, it is developed only for the dual affine-scaling algorithm, and to the best of our knowledge there have been no studies on how to obtain a similar search direction in the primal–dual context.

The warm-start approach proposed in [21] is extended in [26] to the case of solving a sequence of problems with the same dimensions but changing problem data (the objective function or the right-hand side). Such situations arise in the context of decomposition approaches for large structured linear programs. In the case of Dantzig–Wolfe decomposition, successive subproblems differ only in the objective function, while in the case of Benders decomposition they differ only in the right-hand sides. Following [21], nearly optimal points are saved and used to warm start the solution of subsequent subproblems [26].

Yıldırım and Wright [68] consider again the case of solving a sequence of problems in fixed dimensions, and analyse the number of iterations required to converge to a solution of the perturbed instance from the warm-start point and obtain worst-case estimates. They show that these estimates depend on the size of the perturbation as well as on the conditioning of the problem instances. Thus they obtain conditions under which the complexity of the warm-start approach is better than for the cold start case.

The strategy proposed in [68] aims at correcting the primal and dual infeasibilities introduced by the perturbation in just one step. In this strategy, they backtrack to an iterate for which μ is large enough to allow a full step for the correction they produce. Hence, the amount of necessary backtracking depends on the magnitude of the perturbation. This is intuitively justified considering that a large perturbation will produce a large adjustment, and it is essential to guarantee that a full step will not compromise the non-negativity requirements of the iterate. To ensure the availability of an approximate μ-center from which the perturbation can be absorbed in one step, a subset of iterates for different values of μ is stored. When the size of the perturbation becomes known, the closest μ available is retrieved, and the corresponding iterate is used as a warm-start point for the next problem in the sequence. In [68], two different corrections for the perturbation are studied: one is based on least squares, the other on a Newton step correction. A detailed computational comparison of these strategies has been carried out by John and Yıldırım [38].

Gondzio and Grothey [22] propose a different reoptimization technique for interior point methods. As in [68], they aim at obtaining conditions for perturbations that can be absorbed in one Newton step. Hence, they introduce a relative measure of perturbations, in which the primal and the dual spaces are analysed independently.

Their reoptimization procedure is based on two phases: first, an attempt is made to absorb the infeasibilities caused by the perturbation with a full Newton step; second, the centrality of the iterate is improved. Another key feature of their approach is that the primal search direction is governed only by the primal perturbation, and similarly for the dual space. This corresponds to solving the Newton system (2.8) for the two right-hand sides independently

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta s \end{bmatrix} = \begin{bmatrix} \xi_b \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ \xi_c \\ 0 \end{bmatrix}. \tag{5.6}$$

They produce bounds on the magnitude of primal perturbation ξ_b and dual perturbation ξ_c that can be absorbed in a single Newton step. As opposed to the results of [68], the bounds of [22] are easy to compute and thus can be used in practice.

If the perturbation is bounded above by some quantity depending on n and θ, a full Newton step can be taken in the dual space. This happens for both the $\mathcal{N}_2$ and $\mathcal{N}_\infty$ neighbourhoods. Similarly, a full Newton step can be taken in the primal feasibility restoration direction. These conditions may not be satisfied in practice. In such case, a different iterate further away from optimality may be more likely to allow for a full Newton step.

A practical implementation is developed within an infeasible interior point method. In this context, a stronger emphasis may be put on reducing the infeasibilities. This is accomplished with additional centering steps.

An approximate μ-center is stored for a tolerance level that depends on the magnitude of the expected perturbation. This does not exclude the possibility of storing a sequence of iterates, as proposed in [68], thus postponing the choice of the one to use. As the problem size does not change, an approximate μ-center can immediately be used as an iterate for the next problem. If only one iterate is stored, then the absorption of infeasibilities may be spread across a few iterations whenever the stepsizes fall below a predefined level.

As this strategy does not make assumptions upon the centrality of the warm-start iterate, it can be initialised with any iterate. Gondzio and Grothey [22] apply this warm-start strategy successfully to structured problems for crash-start points that come from a cross-decomposition scheme, and thus may lack centrality.

A different approach has been studied by Benson and Shanno [4]. They investigate how to improve the efficiency of interior point methods in a reoptimization context by the use of a primal–dual penalty approach. While standard penalty techniques are effective only in one space, the introduction of penalty parameters in both the primal and the dual problems allows to capture perturbations in both spaces. The penalised problem allows the variables to become negative, with the immediate advantage of accepting larger stepsizes along the computed search direction, thus favouring a faster progress especially in the first few iterations.

Large or huge-scale problems, however, create more than one difficulty to general purpose solvers. Problems of these sizes can be solved by exploiting the structure present in the matrix. A further advantage comes from assigning the work to be done to more than one processing unit. This is where structure-exploiting parallel solvers such as OOPS [24] excel. Moreover, structure-exploiting interior point methods can be used not only for linear programming problems, but also for quadratic and nonlinear problems [23].

In a large scenario tree there may be very little difference among scenarios, and so the large-scale problem can provide a fine-grained solution to a problem that could have been solved more coarsely by using a much smaller tree. This observation suggests a warm-start technique that can be applied in the context of interior point methods. A warm-start solution is obtained by solving the stochastic optimization problem for a reduced event tree, whose dimension is much smaller than that of the complete one. The solution to the reduced problem is used to construct an advanced iterate for the complete formulation. We provide evidence that this novel way of exploiting the problem structure to generate an initial point provides a better iterate (in terms of centrality, feasibility, and closeness to optimality) than the one produced by a generic strategy.

5.3 Other stuff

- The reduced tree approach to construct a warmstart iterate doesn't seem to be IPM specific, and could be exploited also by a simplex solver (in which case, how would such a solver obtain the complete basis?). However, the advantage of IPM are the possibility of solving huge scale problems, parallelism, and extension to quadratic problems.

- Provide a comparison between initial infeasibilities produced by Mehrotra's heuristic and the reduced-tree starting point. It would be great to have some sort of theoretical back-up about the relationship between magnitude of perturbation and initial infeasibilities.

- Point out that the way we generate our starting point guarantees better centrality (at least in the sense of complementary pairs). Possibly this could be compared again with the points generated by Mehrotra's starting point heuristic.

- Explain what we do with probabilities: how we adjust them in the reduced tree, and how they affect the dual iterates.

- Point out that if we knew about the underlying stochastic process, then we could exploit it directly in the generation of the reduced tree (althogh, in such a case how could we ensure the correspondence of nodes between the reduced and the complete tree?)

- Compare the efficiency of the strategy according to the number of variations in the stochastic file. Also, how many variations happen in our test problems?

- Clarify the issue of nonanticipativity constraints: since we have our own deterministic equivalent generator, we already avoid duplication without using those constraints.

  Nonanticipativity means that the decisions taken at any stage do not depend on future realizations of the random parameter or on future decisions, but only on the history.

- ∘ Mention that if the iterate produced by a reduced tree is not good enough (according to what criteria?), another one can be produced by generating a modified reduced tree (more bushy for example).

5.3.1 Theoretical analysis

Once we formulate the complete deterministic equivalent A, we need to satisfy the following system:

$$Ax = b,$$

where $A \in \mathbb{R}^{m \times n}$ is a block-structured matrix of large dimension. Since this is a challenging task, we consider only a fraction of the scenarios we are interested in. Therefore, instead of the complete event tree, we will focus on a subtree. This corresponds to partitioning the deterministic equivalent like this:

$$A = \begin{bmatrix} \tilde{A} & 0 \\ Q & T \end{bmatrix},$$

where $\tilde{A}$ contains only the scenarios corresponding to the subtree, T contains the remaining scenarios, and Q provides the linking blocks. For computational purposes, the dimension of $\tilde{A}$ is much smaller than that of A.

Therefore, to obtain a warm-starting solution we need to solve the reduced system

$$\tilde{A}\tilde{x} = \tilde{b},$$

where we adopted a similar partitioning for x and b.

In the context of primal–dual interior point methods, we are solving the system

$$\begin{bmatrix} \tilde{A} & 0 & 0 \\ 0 & \tilde{A}^T & I \\ \tilde{S} & 0 & \tilde{X} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta\tilde{y} \\ \Delta\tilde{s} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -\tilde{X}\tilde{S}e - \sigma\mu e \end{bmatrix}$$

Given an optimality tolerance ϵ, we need to solve the above system for some primal–dual solution $(\tilde{x}^*, \tilde{s}^*, \tilde{y}^*)$. This will be used as a warm-starting iterate for the complete system.

We face now the problem of how to reconcile the solution from the subtree to the solution for the complete tree.

Later stages are the more subject to variation because they depend on the realisations of all previous stages. Stages closer to the root display less variability when additional scenarios are considered.

5.3.2 Initialising the warm-start iterate

As observed in the literature review above, the perturbations studied were of two types:

- ∘ changes in the data that do not affect the problem size;
- ∘ changes that affect the problem size, by increasing the number of constraints (in either the primal or the dual problem).

In our approach, we have to face a different sort of perturbation, where the size of the problem changes in both the number of constraints and the number of variables in the primal and in the dual. Therefore, we are faced with the challenge of using the solution of the small instance to provide a warm-start iterate for the complete system. This entails initialising both the primal and the dual vectors, of which only a fraction of the elements are known. This objective can be achieved by exploiting the inherent structure of the problem.

We start by showing how to initialise the warm-start iterate in the simplest case possible. Let the primal problem be

$$\begin{array}{lllllll}
\min & c^T x_0 & +c_1^T x_1 & +c_2^T x_2 & +c_3^T x_3 & & \\
\text{s.t.} & F x_0 & + G x_1 & & & = & b_1 \\
 & F x_0 & & + G x_2 & & = & b_2 \\
 & F x_0 & & & + G x_3 & = & b_3 \\
 & x_i & \geq 0, & i = 0, 1, 2, 3. & & &
\end{array}$$

Note that the technology matrix F and the recourse matrix G are the same for all scenarios. This fact is fundamental in our approach, and is not overly simplistic: many real life situations are modelled this way.

Suppose we solve a reduced problem with only two scenarios. While such a reduction is not sensible in practice, it will make our approach evident.

5.3.3 Dual initialisation

After having solved the reduced problem, we want to obtain proper initial values for (x_3, y_3, s_3). To do this, let us consider the dual formulation (WHY?)

$$\begin{array}{llllllll}
\max & b_1^T y_1 + & b_2^T y_2 + & b_3^T y_3 & & & & \\
\text{s.t.} & F^T y_1 + & F^T y_2 + & F^T y_3 + s_0 & & & & = c_0 \\
 & G^T y_1 & & & + s_1 & & & = c_1 \\
 & & G^T y_2 & & & + s_2 & & = c_2 \\
 & & & G^T y_3 & & & + s_3 & = c_3 \\
 & s_i \geq 0, & i = 0, 1, 2, 3. & & & & &
\end{array}$$

We notice that the last set of equations depends only on the newly added variables. Hence, we may want to find a feasible solution with respect to these constraints. By neglecting the first set of constraints, we will likely violate them. We accept this as the unavoidable price to pay if we want to find a computationally viable starting point.

By linearity, if we find weights ω_1 and ω_2 such that

$$\omega_1 c_1 + \omega_2 c_2 = c_3,$$

then we would immediately retrieve y_3 and s_3:

$$y_3 = \omega_1 y_1 + \omega_2 y_2, \quad s_3 = \omega_1 s_1 + \omega_2 s_2.$$

However, without imposing conditions on ω_1 and ω_2, we have no guarantee to obtain $s_3 \geq 0$.

Let us first consider the unconstrained subproblem. As the system is overdetermined, we aim for a least squares solution. Hence we try to find weights such that

$$\min \; \|c_3 - C\omega\|_2^2,$$

where $C = [c_1\ c_2]$ and $\omega = [\omega_1\ \omega_2]^T$. This corresponds to

$$\min\ (c_3 - C\omega)^T(c_3 - C\omega),$$

whose solution satisfies

$$C^T C\omega = C^T c_3.$$

Let us now turn to the constrained problem

$$\begin{array}{ll} \min & (c_3 - C\omega)^T(c_3 - C\omega) \\ \text{s.t.} & Sw \geq 0, \end{array}$$

where $S = [s_1\ s_2]$. The solution to this problem satisfies

$$C^T C\omega = C^T c_3 + \frac{1}{2}S^T\lambda,$$

where λ is the vector of Lagrange multipliers and can be used to penalise solutions that violate the non-negativity constraints. We first set $\lambda = 0$ and thus solve the unconstrained problem. If this produces an iterate $s_3 \geq 0$, then we are done. Otherwise, we set $\lambda^{(j)} > 0$ for the components $s_3^{(j)} < 0$ and solve again. This may need to be repeated until we obtain weights ω for which $S\omega \geq 0$.

This ad-hoc mechanism may be too complicated to be computationally viable. Another possibility is to solve the constrained problem as a quadratic programming problem. However, this possibility may still require more effort than what we are willing to use.

For a practical implementation, we may resort to the simpler strategy of shifting any negative values of $S\omega$. We observe that a similar approach is used in finding the starting point for interior point methods [50].

5.3.4 Primal initialisation

How to initialise the primal variables:

- From dual such that $x_3 s_3 = \mu$;
- From $Gx_3 = b_3 - Fx_0$ (maybe plus additional conditions on centrality since it's an underdetermined system).

Chapter 6

Conclusions

The relative youngness of interior point methods means that there is still a lot to learn and try, particularly in practical implementations.

A big avenue of research, according to the author, is the development of specialised techniques to exploit the problem structure.

This means that the development and diffusion of structure-exploiting codes and of structure-aware modelling languages may become a necessary requirement for a new generation of interior point codes.

In this sense, also theoretical developments in these aspects are wanted and necessary.

Bibliography

[1] Erling D. Andersen, Jacek Gondzio, Csaba Mészáros, and Xiaojie Xu. Implementation of interior point methods for large scale linear programming. In T. Terlaky, editor, *Interior Point Methods in Mathematical Programming*, pages 189–252. Kluwer Academic Publishers, 1996.

[2] David S. Atkinson and Pravin M. Vaidya. A scaling technique for finding the weighted analytic center of a polytope. *Mathematical Programming*, 57:163–192, 1992.

[3] Dave A. Bayer and Jeffrey C. Lagarias. The nonlinear geometry of linear programming I. Affine and projective scaling trajectories. *Transactions of the American Mathematical Society*, 314(2):499–526, 1989.

[4] Hande Y. Benson and David F. Shanno. An exact primal-dual penalty method approach to warmstarting interior-point methods for linear programming. *Computational Optimization and Applications*, to appear, 2006.

[5] John R. Birge and François Louveaux. *Introduction to Stochastic Programming.* Springer Series in Operations Research, 1997.

[6] Robert E. Bixby. Progress in linear programming. *ORSA Journal on Computing*, 6(1):15–22, 1994.

[7] Robert E. Bixby. Solving real-world linear programs: A decade and more of progress. *Operations Research*, 50(1):3–15, 2002.

[8] Tamra J. Carpenter, Irvin J. Lustig, John M. Mulvey, and David F. Shanno. Higher-order predictor-corrector interior point methods with application to quadratic objectives. *SIAM Journal on Optimization*, 3(4):696–725, 1993.

[9] Coralia Cartis. Some disadvantages of a Mehrotra-type primal-dual corrector interior-point algorithm for linear programming. Technical Report 04/27, Numerical Analysis Group, Computing Laboratory, Oxford University, 2004.

[10] Coralia Cartis. On the convergence of a primal-dual second-order corrector interior point algorithm for linear programming. Technical Report 05/04, Numerical Analysis Group, Computing Laboratory, Oxford University, 2005.

[11] Vašek Chvátal. *Linear programming.* W. H. Freeman and Company, New York, 1983.

[12] Marco Colombo and Jacek Gondzio. Further development of multiple centrality correctors. Technical Report MS-05-001, School of Mathematics, The University of Edinburgh, October 2005. Accepted for publication in *Computational Optimization and Applications.*

[13] Marco Colombo, Jacek Gondzio, and Andreas Grothey. A warm-start approach for large-scale stochastic linear programs. Technical Report MS-06-004, School of Mathematics, The University of Edinburgh, August 2006.

[14] Joe Czyzyk, Sanjay Mehrotra, and Steve Wright. PCx user guide. Technical Report OTC 96/01, Optimization Technology Center, May 1996.

[15] George B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, 1963.

[16] George B. Dantzig. Linear programming. *Operations Research*, 50(1):42–47, 2002.

[17] Jitka Dupačová, Giorgio Consigli, and Stein W. Wallace. Scenarios for multistage stochastic programs. *Annals of Operations Research*, 100:25–53, 2000.

[18] Shu-Cherng Fang and Sarat Puthenpura. *Linear optimization and extensions: theory and algorithms.* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[19] Jacek Gondzio. HOPDM (version 2.12) – A fast LP solver based on a primal-dual interior point method. *European Journal of Operational Research*, 85:221–225, 1995.

[20] Jacek Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6:137–156, 1996.

[21] Jacek Gondzio. Warm start of the primal-dual method applied in the cutting-plane scheme. *Mathematical Programming*, 83:125–143, 1998.

[22] Jacek Gondzio and Andreas Grothey. Reoptimization with the primal-dual interior point method. *SIAM Journal on Optimization*, 13(3):842–864, 2003.

[23] Jacek Gondzio and Andreas Grothey. Solving nonlinear portfolio optimization problems with the primal-dual interior point method. Technical Report MS-04-001, School of Mathematics, The University of Edinburgh, October 2004. Accepted for publication in *European Journal of Operational Research.*

[24] Jacek Gondzio and Robert Sarkissian. Parallel interior-point solver for structured linear programs. *Mathematical Programming*, 96:561–584, 2003.

[25] Jacek Gondzio and Tamás Terlaky. A computational view of interior point methods for linear programming. In J. Beasley, editor, *Advances in Linear and Integer Programming*, chapter 3, pages 103–144. Oxford University Press, Oxford, England, 1996.

[26] Jacek Gondzio and Jean-Philippe Vial. Warm start and ε-subgradients in a cutting plane scheme for block-angular linear programs. *Computational Optimization and Applications*, 14:17–36, 1999.

[27] Clovis C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34(2):167–224, 1992.

[28] Clovis C. Gonzaga and Marli Cardia. Properties of the central points in linear programming problems. *Numerical Algorithms*, 35:185–204, 2004.

[29] Osman Güler, Cornelis Roos, Tamás Terlaky, and Jean-Philippe Vial. A survey of the implications of the behavior of the central path for the duality theory of linear programming. *Management Science*, 41(12):1922–1934, 1995.

[30] Julian A. J. Hall and Ken I. M. McKinnon. Hyper-sparsity in the revised simplex method and how to exploit it. *Computational Optimization and Applications*, 32(3):259–283, 2005.

[31] Alexander L. Hipolito. A weighted least squares study of robustness in interior point linear programming. *Computational Optimization and Applications*, 2:29–46, 1993.

[32] Kjetil Høyland, Michal Kaut, and Stein W. Wallace. A heuristic for moment-matching scenario generation. *Computational Optimization and Applications*, 24(2-3):169–185, 2003.

[33] Kjetil Høyland and Stein W. Wallace. Generating scenario trees for multistage decision problems. *Management Science*, 47(2):295–307, February 2001.

[34] Preconditioning indefinite systems in interior point methods for optimization. Luca bergamaschi and jacek gondzio and giovanni zilli. *Computational Optimization and Applications*, 28:149–171, 2004.

[35] Benjamin Jansen. *Interior point techniques in optimization. Complementarity, sensitivity and algorithms.* PhD thesis, Delft University of Technology, 1995.

[36] Benjamin Jansen, Cornelis Roos, and Tamás Terlaky. Primal-dual target-following algorithms for linear programming. *Annals of Operations Research*, 62:197–231, 1996.

[37] Florian Jarre and Martin Wechs. Extending Mehrotra's corrector for linear programs. *Advanced Modeling and Optimization*, 1:38–60, 1999.

[38] Elizabeth John and E. Alper Yıldırım. Implementation of warm-start strategies in interior-point methods for linear programming in fixed dimensions. Technical report, Department of Industrial Engineering, Bilkent University, Turkey, May 2006. Accepted for publication in *Computational Optimization and Applications*.

[39] Peter Kall and János Mayer. Some insights into the solution algorithms for SLP problems. *Annals of Operations Research*, 142:147–164, 2006.

[40] Peter Kall and Stein W. Wallace. *Stochastic programming.* John Wiley and Sons, New York, 1994.

[41] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[42] Victor Klee and George J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities – III*, pages 159–175. Academic Press, New York, 1972.

[43] Masakazu Kojima, Nimrod Megiddo, and Shinji Mizuno. A primal–dual infeasible-interior-point algorithm for linear programming. *Mathematical Programming*, 61:263–280, 1993.

[44] Masakazu Kojima, Shinji Mizuno, and Akiko Yoshise. A polynomial-time algorithm for a class of linear complementarity problems. *Mathematical Programming*, 44:1–26, 1989.

[45] Irvin J. Lustig. Feasibility issues in a primal–dual interior-point method for linear programming. *Mathematical Programming*, 49:145–162, 1991.

[46] Irvin J. Lustig, Roy E. Marsten, and David F. Shanno. On implementing Mehrotra's predictor–corrector interior-point method for linear programming. *SIAM Journal on Optimization*, 2(3):435–449, 1992.

[47] István Maros and Csaba Mészáros. The role of the augmented system in interior point methods. *European Journal of Operational Research*, 107:720–736, 1998.

[48] Nimrod Megiddo. Pathways to the optimal set in linear programming. In N. Megiddo, editor, *Progress in Mathematical Programming: Interior-Point and Related Methods*, chapter 8, pages 131–158. Springer-Verlag, New York, 1989.

[49] Nimrod Megiddo. On finding primal- and dual-optimal bases. *ORSA Journal on Computing*, 3(1):63–65, 1991.

[50] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[51] Sanjay Mehrotra and Zhifeng Li. Convergence conditions and Krylov subspace-based corrections for primal-dual interior-point method. *SIAM Journal on Optimization*, 15(3):635–653, 2005.

[52] John E. Mitchell. *Karmakar's Algorithm and Combinatorial Optimization Problems.* PhD thesis, Cornell University, 1988.

[53] John E. Mitchell and Michael J. Todd. Solving combinatorial optimization problems using Karmarkar's algorithm. *Mathematical Programming*, 56:245–284, 1992.

[54] Shinji Mizuno, Michael J. Todd, and Yinyu Ye. On adaptive step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations Research*, 18:964–981, 1993.

[55] Renato D. C. Monteiro and Ilan Adler. Interior path following primal–dual algorithms. Part I: linear programming. *Mathematical Programming*, 44:27–41, 1989.

[56] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization.* John Wiley and Sons, New York, 1988.

[57] Aurelio R. L. Oliveira and Danny C. Sorensen. A new class of preconditioners for large-scale linear systems from interior point methods for linear programming. *Linear Algebra and its Applications*, 394:1–24, 2005.

[58] James M. Ortega and Werner C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Academic Press, New York, 1970.

[59] George Ch. Pflug. Scenario tree generation for multiperiod financial optimization by optimal discretization. *Mathematical Programming*, 89(2):251–271, 2001.

[60] Maziar Salahi, Jiming Peng, and Tamás Terlaky. On Mehrotra-type predictor-corrector algorithms. AdvOl Report 2005/4, McMaster University, 2005.

[61] Alexander Schrijver. *Theory of linear and integer programming.* John Wiley and Sons, New York, 1986.

[62] Ron Shamir. The efficiency of the simplex method: a survey. *Management Science*, 33(3):301–334, 1987.

[63] Richard Tapia, Yin Zhang, Matthew Saltzman, and Alan Weiser. The Mehrotra predictor-corrector interior-point method as a perturbed composite Newton method. *SIAM Journal on Optimization*, 6(1):47–56, 1996.

[64] Michael J. Todd and Yinyu Ye. A centered projective algorithm for linear programming. *Mathematics of Operations Research*, 15(3):508–529, 1990.

[65] Stephen A. Vavasis and Yinyu Ye. A primal–dual interior point method whose running time depends only on the constraint matrix. *Mathematical Programming*, 74(1):79–120, 1996.

[66] Margaret H. Wright. Interior methods for constrained optimization. In *Acta Numerica 1992*, pages 341–407. Cambridge University Press, 1992.

[67] Stephen J. Wright. *Primal-dual interior-point methods.* SIAM, Philadelphia, 1997.

[68] E. Alper Yıldırım and Stephen J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM Journal on Optimization*, 12(3):782–810, 2002.