ELSEVIER

Journal of Computational and Applied Mathematics 124 (2000) 281–302

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.nl/locate/cam

Interior-point methods

Florian A. Potra[a], Stephen J. Wright[b,*]

[a] *Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD 21250, USA*
[b] *Division of Mathematics and Computer Science, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439-4844, USA*

Received 18 November 1999; received in revised form 10 February 2000

Abstract

The modern era of interior-point methods dates to 1984, when Karmarkar proposed his algorithm for linear programming. In the years since then, algorithms and software for linear programming have become quite sophisticated, while extensions to more general classes of problems, such as convex quadratic programming, semi-definite programming, and nonconvex and nonlinear problems, have reached varying levels of maturity. We review some of the key developments in the area, and include comments on the complexity theory and practical algorithms for linear programming, semi-definite programming, monotone linear complementarity, and convex programming over sets that can be characterized by self-concordant barrier functions.

1. Introduction

In their survey article [6], Freund and Mizuno wrote

> Interior-point methods in mathematical programming have been the largest and most dramatic area of research in optimization since the development of the simplex method...Interior-point methods have permanently changed the landscape of mathematical programming theory, practice and computation... .

Although most research in the area was devoted to linear programming, the authors claimed that

> semidefinite programming is the most exciting development in mathematical programming in 1990s.

Although various interior-point methods had been considered one way or another from the 1950s, and investigated quite extensively during the 1960s [5], it was the publication of the seminal paper

* Corresponding author.
E-mail addresses: potra@math.umbc.edu (F.A. Potra), wright@mcs.anl.gov (S.J. Wright).

0377-0427/00/$ - see front matter
PII: S0377-0427(00)00433-7

of Karmarkar [11] that placed interior-point methods at the top of the agenda for many researchers. On the theoretical side, subsequent research led to improved computational complexity bounds for linear programming (LP), quadratic programming (QP), linear complementarity problems (LCP) semi-definite programming (SDP) and some classes of convex programming problems. On the computational side, high-quality software was eventually produced, much of it freely available. The general performance of computational tools for linear programming improved greatly, as the sudden appearance of credible competition spurred significant improvements in implementations of the simplex method.

In the first years after Karmarkar's initial paper, work in linear programming focused on algorithms that worked with the primal problem, but were more amenable to implementation than the original method or that had better complexity bounds. A particularly notable contribution from this period was Renegar's algorithm [21], which used upper bounds on the optimal objective value to form successively smaller subsets of the feasible set, each containing the solution, and used Newton's method to follow the analytic centers of these subsets to the primal optimum. A new era was inaugurated with Megiddo's paper [13], originally presented in 1987, which described a framework for primal–dual framework algorithms. The primal–dual viewpoint proved to be extremely productive. It yielded new algorithms with interesting theoretical properties, formed the basis of the best practical algorithms, and allowed for transparent extensions to convex programming and linear complementarity. In 1989, Mehrotra described a practical algorithm for linear programming that remains the basis of most current software; his work appeared in 1992 [14]. Meanwhile, Nesterov and Nemirovskii [16] were developing the theory of self-concordant functions, which allowed algorithms based on the primal log-barrier function for linear programming to be extended to wider classes of convex problems, particularly semi-definite programming and second-order cone programming (SOCP). Nesterov and Todd [17,18] extended the primal–dual approach along similar lines to a more restricted class of convex problems that still included SDP and SOCP. Other work on interior-point algorithms for SDPs, which have a wide variety of applications in such areas as control and structural optimization, was already well advanced by this point. Work on these algorithms gained additional impetus when it was recognized that approximate solutions of NP-hard problems could thereby be obtained in polynomial time.

We now outline the remainder of the paper. Section 2 discusses linear programming, outlining the pedigree of the most important algorithms and various computational issues. In Section 3, we discuss extensions to quadratic programming and linear complementarity problems, and compare the resulting algorithms with active-set methods. Semi-definite programming is the topic of Section 4. Section 5 contains some elements of the theory of self-concordant functions and self-scaled cones. Finally, we present some conclusions in Section 6.

There are many other areas of optimization in which areas in which interior-point approaches have made an impact, though in general the state of the art is less mature than for the areas mentioned above. General convex programming problems of the form

$$\min_x \ f(x) \quad \text{s.t. } g_i(x) \leqslant 0, \ \ i = 1, 2, \ldots, m$$

(where f and g_i, $i = 1, 2, \ldots, m$, are convex functions) can be solved by extensions of the primal–dual approach of Section 3. Interestingly, it is possible to prove superlinear convergence of these primal–dual algorithms without assuming linear independence of the active constraints at the solution. This observation prompted recent work on improving the convergence properties of other algorithms,

notably sequential quadratic programming. A number of researchers have used interior-point methods in algorithms for combinatorial and integer programming problems. (In some cases, the interior-point method is used to find an inexact solution of related problems in which the integrality constraints are relaxed.) In decomposition methods for large linear and convex problems, such as Dantzig–Wolfe/column generation and Benders' decomposition, interior-point methods have been used to find inexact solutions of the large master problems, or to approximately solve analytic center subproblems to generate test points. Additionally, application of interior-point methodology to non-convex nonlinear programming has occupied many researchers for some time now. The methods that have been proposed to date contain many ingredients, including primal–dual steps, barrier and merit functions, and scaled trust regions.

For references to work mentioned in the previous paragraph, and for many other results discussed but not cited in this paper, please see the bibliography of the technical report in [28].

A great deal of literature is available to the reader interested in learning more about interior-point methods. A number of recent books [27,29,23] give overviews of the area, from first principles to new results and practical considerations. Theoretical background on self-concordant functionals and related developments is described in [16,22]. Technical reports from the past five years can be obtained from the Interior-Point Methods Online Web site at `www.mcs.anl.gov/otc/InteriorPoint`.

2. Linear programming

We consider first the linear programming problem, which is undoubtedly the optimization problem solved most frequently in practice. Given a cost vector $c \in \mathbb{R}^n$, m linear equality constraints defined by a matrix $A \in \mathbb{R}^{m\times n}$ and a vector $b \in \mathbb{R}^m$, the linear programming problem can be stated in its standard form as

$$\min_x c^{\mathrm{T}}x \quad \text{s.t. } Ax = b,\ \ x \geqslant 0. \tag{2.1}$$

The restriction $x \geqslant 0$ applies componentwise, that is, all components of the vector $x \in \mathbb{R}^n$ are required to be nonnegative.

The simplex method developed by Dantzig between 1947 and 1951 has been the method of choice for linear programming. While performing very well in practice, its worst-case computational complexity is exponential, as shown by the example of Klee and Minty from 1972. The problem of existence of a (weakly) polynomial algorithm for solving linear programs with integer data was solved by Khachiyan in 1979. He proved that the ellipsoid method solves such programs in $O(n^2L)$ iterations, requiring a total of $O(n^4L)$ bit operations, where L is the length of a binary coding of the input data, that is

$$L = \sum_{i=0}^{m}\sum_{j=0}^{n}\lceil \log_2(|a_{ij}| + 1) + 1\rceil$$

with $a_{i0} = b_i$ and $a_{0j} = c_j$.

There are no known implementations of the ellipsoid method for linear programming that are remotely competitive with existing practical codes. The merit of the celebrated paper of Karmarkar [11] consisted not so much in lowering the bound on the computational complexity of LP to $O(nL)$ iterations, requiring a total of $O(n^{3.5}L)$ bit operations, as in the fact that it was possible to implement his algorithm with reasonable efficiency. The theoretical computational complexity of interior-point

methods for LP was eventually lowered to $O(\sqrt{n}L)$ iterations, requiring a total of $O(n^3L)$ bit operations by a number of authors. Goldfarb and Todd [8] provide a good reference for these complexity results. By using fast matrix multiplication techniques, the complexity estimates can be reduced further. Quite recently, Anstreicher [1] proposed an interior-point method, combining partial updating with a preconditioned gradient method, that has an overall complexity of $O(n^3/\log n)$ bit operations. The paper [1] contains references to recent complexity results for LP.

The best of these complexity results, all of which are of major theoretical importance, are obtained as a consequence of global linear convergence with factor $1-c/\sqrt{n}$. In what follows we will describe a simple interior algorithm that achieves this rate. We assume that the linear program (2.1) has a strict interior, that is, the set

$$\mathscr{F}^0 \stackrel{\text{def}}{=} \{x \mid Ax = b,\ x > 0\}$$

is nonempty, and that the objective function is bounded below on the set of feasible points. Under these assumptions, (2.1) has a (not necessarily unique) solution.

By using a logarithmic barrier function to account for the bounds $x \geqslant 0$, we obtain the parameterized optimization problem

$$\min_x\ f(x;\mu) \stackrel{\text{def}}{=} \frac{1}{\mu}c^{\mathrm{T}}x - \sum_{i=1}^{n}\log x_i, \quad \text{s.t. } Ax = b, \tag{2.2}$$

where log denotes the natural logarithm and $\mu > 0$ denotes the barrier parameter. Because the logarithmic function requires its arguments to be positive, the solution $x(\mu)$ of (2.2) must belong to $\mathscr{F}^0$. It is well known (see, for example, [26, Theorem 5]) that for any sequence $\{\mu_k\}$ with $\mu_k \downarrow 0$, all limit points of $\{x(\mu_k)\}$ are solutions of (2.1).

The traditional SUMT approach [5] accounts for equality constraints by including a quadratic penalty term in the objective. When the constraints are linear, as in (2.1), it is simpler and more appropriate to handle them explicitly. By doing so, we devise a *primal barrier algorithm* in which a projected Newton method is used to find an approximate solution of (2.2) for a certain value of μ, and then μ is decreased. Note that

$$\nabla^2_{xx} f(x;\mu) = -X^{-2}, \quad \nabla_x f(x;\mu) = (1/\mu)c + X^{-1}e,$$

where $X = \operatorname{diag}(x_1, x_2, \ldots, x_n)$ and $e = (1, 1, \ldots, 1)^{\mathrm{T}}$. The projected Newton step Δx from a point x satisfies the following system:

$$\begin{bmatrix} -\mu X^{-2} & A^{\mathrm{T}} \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \lambda^+ \end{bmatrix} = - \begin{bmatrix} c + \mu X^{-1}e \\ Ax - b \end{bmatrix}, \tag{2.3}$$

so that Eq. (2.3) are the same as those that arise from a sequential quadratic programming algorithm applied to (2.2), modulo the scaling by μ in the first line of (2.3). A line search can be performed along Δx to find a new iterate $x + \alpha\Delta x$, where $\alpha > 0$ is the step length.

The prototype primal barrier algorithm can be specified as follows:

primal barrier algorithm
Given $x^0 \in \mathscr{F}^0$ and $\mu_0 > 0$;
Set $k \leftarrow 0$;
repeat
Obtain x^{k+1} by performing one or more Newton steps (2.3), starting at $x = x^k$, and fixing $\mu = \mu_k$;
Choose $\mu_{k+1} \in (0, \mu_k)$; $k \leftarrow k + 1$;
until some termination test is satisfied.

A *short-step* version of this algorithm takes a single Newton step at each iteration, with step length $\alpha = 1$, and sets

$$\mu_{k+1} = \mu_k \Big/ \left(1 + \frac{1}{8\sqrt{n}}\right). \tag{2.4}$$

It is known (see, for instance, [22, Section 2.4]) that if the feasible region of (2.1) is bounded, and x^0 is sufficiently close to $x(\mu_0)$ in a certain sense, then we obtain a point x^k whose objective value $c^{\mathrm{T}}x^k$ is within ε of the optimal value after

$$\mathrm{O}\left(\sqrt{n}\log\frac{n\mu_0}{\varepsilon}\right) \text{ iterations}, \tag{2.5}$$

where the constant factor disguised by the $\mathrm{O}(\cdot)$ depends on the properties of (2.1) but is independent of n and ε. For integer data of bitlength L, it is known that if $\varepsilon \leqslant 2^{-2L}$ then x^k can be rounded to an exact solution in $\mathrm{O}(n^3)$ arithmetic operations. Moreover, provided we can choose the initial point such that $\mu_0 \leqslant 2^{\beta L}$ for some positive constant β, the iteration complexity will be $\mathrm{O}(\sqrt{n}L)$.

The rate of decrease of μ in short-step methods is too slow to allow good practical behavior, so *long-step* variants have been proposed that decrease μ more rapidly, while possibly taking more than one Newton step for each μ_k and also using a line search. Although long-step algorithms have better practical behavior, the complexity estimates associated with them typically are no better than estimate (2.5) for the short-step approach. In fact, a recurring theme of worst-case complexity estimates for linear programming algorithms is that no useful relationship exists between the estimate and the practical behavior of the algorithm. Indeed, as we have seen above, the best-known iteration complexity bound is obtained from a rather slow linear convergence rate. Good practical performance is obtained by algorithms that are superlinearly convergent.

Better practical algorithms are obtained from the primal–dual framework. These methods recognize the importance of the path of solutions $x(\mu)$ to (2.2) in the design of algorithms, but differ from the approach above in that they treat the dual variables explicitly in the problem, rather than as adjuncts to the calculation of the primal iterates. The dual problem for (2.1) is

$$\max_{(\lambda,s)} b^{\mathrm{T}}\lambda \quad \text{s.t. } A^{\mathrm{T}}\lambda + s = c, \quad s \geqslant 0, \tag{2.6}$$

where $s \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}^m$, and the optimality conditions for x^* to be a solution of (2.1) and (λ^*, s^*)

to be a solution of (2.6) are that $(x,\lambda,s)=(x^*,\lambda^*,s^*)$ satisfies

$$Ax = b, \tag{2.7a}$$

$$A^{\mathrm{T}}\lambda + s = c, \tag{2.7b}$$

$$XSe = 0, \tag{2.7c}$$

$$(x,s) \geqslant 0, \tag{2.7d}$$

where $X=\mathrm{diag}(x_1,x_2,\ldots,x_n)$ and $S=\mathrm{diag}(s_1,s_2,\ldots,s_n)$. Primal–dual methods solve (2.1) and (2.6) simultaneously by generating a sequence of iterates (x^k,λ^k,s^k) that in the limit satisfies conditions (2.7). As mentioned above, the *central path* defined by the following perturbed variant of (2.7) plays an important role in algorithm design:

$$Ax = b, \tag{2.8a}$$

$$A^{\mathrm{T}}\lambda + s = c, \tag{2.8b}$$

$$XSe = \mu e, \tag{2.8c}$$

$$(x,s) > 0, \tag{2.8d}$$

where $\mu > 0$ parameterizes the path. Note that these conditions are simply the optimality conditions for the problem (2.2): If $(x(\mu),\lambda(\mu),s(\mu))$ satisfies (2.8), then $x(\mu)$ is a solution of (2.2). We have from (2.8c) that a key feature of the central path is that

$$x_i s_i = \mu \quad \text{for all } i = 1,2,\ldots,n, \tag{2.9}$$

that is, the pairwise products $x_i s_i$ are identical for all i.

In primal–dual algorithms, steps are generated by applying a perturbed Newton methods to the three equalities in (2.8), which form a nonlinear system in which the number of equations equals the number of unknowns. We constrain all iterates (x^k,λ^k,s^k) to have $(x^k,s^k)>0$, so that the matrices X and S remain positive diagonal throughout, ensuring that the perturbed Newton steps are well defined. Supposing that we are at a point (x,λ,s) with $(x,s)>0$ and the feasibility conditions $Ax=b$ and $A^{\mathrm{T}}\lambda + s = c$ are satisfied, the primal–dual step $(\Delta x,\Delta\lambda,\Delta s)$ is obtained from the following system:

$$\begin{bmatrix} 0 & A & 0 \\ A^{\mathrm{T}} & 0 & I \\ 0 & S & X \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \\ \Delta s \end{bmatrix} = - \begin{bmatrix} 0 \\ 0 \\ XSe - \sigma\mu e + r \end{bmatrix}, \tag{2.10}$$

where $\mu = x^{\mathrm{T}}s/n$, $\sigma \in [0,1]$, and r is a perturbation term, possibly chosen to incorporate higher-order information about the system (2.8), or additional terms to improve proximity to the central path.

Using the general step (2.10), we can state the basic framework for primal–dual methods as follows:

primal–dual algorithm
Given (x^0, λ^0, s^0) with $(x^0, s^0) > 0$;
Set $k \leftarrow 0$ and $\mu_0 = (x^0)^{\mathrm{T}} s^0 / n$;
repeat
Choose σ_k and r^k;
Solve (2.10) with $(x, \lambda, s) = (x^k, \lambda^k, s^k)$ and $(\mu, \sigma, r) = (\mu_k, \sigma_k, r^k)$
to obtain $(\Delta x^k, \Delta \lambda^k, \Delta s^k)$;
Choose step length $\alpha_k \in (0, 1]$ and set
$(x^{k+1}, \lambda^{k+1}, s^{k+1}) \leftarrow (x^k, \lambda^k, s^k) + \alpha_k(\Delta x^k, \Delta \lambda^k, \Delta s^k)$;
$\mu_{k+1} \leftarrow (x^{k+1})^{\mathrm{T}} s^{k+1} / n$; $k \leftarrow k + 1$;
until some termination test is satisfied.

The various algorithms that use this framework differ in the way that they choose the starting point, the centering parameter σ_k, the perturbation vector r^k, and the step α_k. the simplest algorithm – a short-step path-following method similar to the primal algorithm described above – sets

$$r^k = 0, \quad \sigma_k \equiv 1 - \frac{0.4}{\sqrt{n}}, \quad \alpha_k \equiv 1 \tag{2.11}$$

and, for suitable choice of a feasible starting point, achieves convergence to a feasible point (x, λ, s) with $x^{\mathrm{T}} s / n \leqslant \varepsilon$ for a given ε in

$$\mathrm{O}\left(\sqrt{n} \log \frac{\mu_0}{\varepsilon}\right) \text{ iterations.} \tag{2.12}$$

Note the similarity of both the algorithm and its complexity estimate to the corresponding primal algorithm. As in that case, algorithms with better practical performance but not necessarily better complexity estimates can be obtained through more aggressive, adaptive choices of the centering parameter (that is, σ_k closed to zero). They use a line search to maintain proximity to the central path. The proximity requirement dictates, implicitly or explicitly, that while condition (2.9) may be violated, the pairwise products must not be too different from each other. For example, some algorithms force the iterates to remain in l_2-neighborhoods of the central path of the form

$$\mathcal{N}(\beta) \stackrel{\text{def}}{=} \{(x, \lambda, s) \mid (x, s) > 0, \ ||Xs - \mu e||_2 \leqslant \beta\}. \tag{2.13}$$

A very interesting algorithm of this type is the Mizuno–Todd–Ye predictor corrector method which can be described as follows:

predictor–corrector algorithm
Given $(x^0, \lambda^0, s^0) \in \mathcal{N}(0.25)$
Set $k \leftarrow 0$ and $\mu_0 = (x^0)^{\mathrm{T}} s^0/n$;
repeat
Set $(x, \lambda, s) \leftarrow (x^k, \lambda^k, s^k)$ and $(\mu, \sigma, r) \leftarrow (\mu_k, 0, 0)$;
Solve (2.10) and set $(u, w, v) \leftarrow (\Delta x, \Delta\lambda, \Delta s)$;
to obtain $(\Delta x^k, \Delta\lambda^k, \Delta s^k)$;
Choose step length α_k as the largest $\alpha_k \in (0, 1]$ such that:
$(x, \lambda, s) + \alpha(u, w, v) \in \mathcal{N}(0.25)$
Set $(x, \lambda, s) \leftarrow (x, \lambda, s) + \alpha_k(u, w, v)$ and $(\mu, \sigma, r) \leftarrow (\mu_k, (1 - \alpha_k), 0)$;
Solve (2.10) and set
$(x^{k+1}, \lambda^{k+1}, s^{k+1}) \leftarrow (x, \lambda, s) + (\Delta x, \Delta\lambda, \Delta s)$;
$\mu_{k+1} \leftarrow (x^{k+1})^{\mathrm{T}} s^{k+1}/n$; $k \leftarrow k + 1$;
until some termination test is satisfied.

It can be proved that the above algorithm has the iteration complexity bound (2.12), the same as the short-step algorithm defined by (2.11). We note that the predictor–corrector method requires the solution of two linear systems per iteration (one in the predictor step and another one in the corrector step), while the short-step algorithm requires only the solution of one linear system per iteration. However, numerical experiments show that the predictor–corrector algorithm is significantly more efficient than the short-step algorithm. This is explained by the fact that while with the short-step algorithm μ_k decreases by a fixed factor at each step, i.e.,

$$\mu_{k+1} = \left(1 - \frac{0.4}{n}\right)\mu_k, \quad k = 0, 1, 2, \ldots \tag{2.14}$$

the predictor–corrector algorithm, by its adaptive choice of σ_k, allows μ_k to decrease faster, especially close to the solution. Ye et al. [30] proved that the predictor–corrector algorithm is quadratically convergent in the sense that

$$\mu_{k+1} \leqslant B\mu_k^2, \quad k = 0, 1, 2, \ldots \tag{2.15}$$

for some constant B independent of k. This constant may be large, so that (2.15) ensures a better decrease of μ_k that (2.14) only if μ_k is sufficiently small (specifically, $\mu_k < (1 - 0.4/n)/B$). There are examples in which quadratic convergence cannot be observed until quite late in the algorithm — the last few iterations. Even in these examples, the linear decrease factor in μ_k in early iterations is much better than $(1 - 0.4/n)$, because of the adaptive choice of σ_k.

Even better reductions of μ_k in the early iteration can be obtained by considering larger neighborhoods of the central path than the l_2-neighborhoods (2.13). The worst-case complexity bounds of the resulting algorithms deteriorates — $O(nL)$ instead of $O(\sqrt{n}L)$ — but the practical performance is better.

Quadratic convergence, or, more generally, superlinear convergence is also important for the following reason. The condition of the linear systems to be solved at each iteration often worsens as μ_k becomes small, and numerical problems are sometimes encountered. Superlinearly convergent

algorithms need to perform only a couple of iterations with these small μ_k. When μ_k is small enough, a projection can be used to identify an exact solution. A finite-termination strategy can also be implemented by using the Tapia indicators to decide which components of x and s are zero at the solution [4]. The use of a finite-termination strategy in conjunction with superlinearly convergent algorithms for linear programming is somewhat superfluous, since the domain range of μ_k values for which superlinear convergence is obtained appears to be similar to the range on which finite termination strategies are successful. Once the iterates enter this domain, the superlinear method typically converges in a few steps, and the savings obtained by invoking a finite termination strategy are not great.

In the above algorithms we assumed that a starting point satisfying exactly the linear constraints and lying in the interior of the region defined by the inequality constraints is given. In practice, however, it may be difficult to obtain such a starting point, so many efficient implementations of interior-point methods use starting points that lie in the interior of the region defined by the inequality constraints but do not necessarily satisfy the equality constraints. Such methods are called infeasible-interior-point methods, and they are more difficult to analyze. The first global convergence result for such methods was obtained by Kojima, Megiddo and Mizuno, while the first polynomial complexity result was given by Zhang [32]. The computational complexity of the infeasible-interior-point algorithms typically is worse than in the feasible case. An advantage is that these algorithms can solve problems for which no strictly feasible points exist. They also can be used to detect the infeasibility of certain linear programming problems.

A different way of dealing with infeasible starting points was proposed by Ye et al. [31]. Starting with a linear programming problem in standard form and with a possibly infeasible starting point whose x and s components are strictly positive, they construct a homogeneous self-dual linear program for which a strictly feasible starting point is readily available. The solution of the original problem is obtained easily from the solution of the homogeneous program. When the original linear program is infeasible, this fact can be ascertained easily from the solution of the homogeneous problem.

The practical performance of a numerical algorithm is explained better by a probabilistic complexity analysis than by a worst-case complexity analysis. For example, the probabilistic computational complexity of the simplex method is strongly polynomial (that is, a polynomial in the dimension n of the problem only), which is closer to practical experience with this method than the exponential complexity of the worst-case analysis (see [3] and the literature cited therein). As mentioned above, the worst-case complexity of interior-point methods is weakly polynomial, in the sense that the iteration bounds are polynomials in the dimension n and the bitlength of the data L. In [2], it is shown that from a probabilistic point of view the iteration complexity of a class of interior-point methods is $O(\sqrt{n}\ln n)$. Thus the probabilistic complexity of this class on interior-point methods is strongly polynomial, that is, the complexity depends only on the dimension of the problem and not on the binary length of the data.

Most interior-point software for linear programming is based on Mehrotra's predictor–corrector algorithm [14], often with the higher-order enhancements described in [9]. This approach uses an adaptive choice of σ_k, selected by first solving for the pure Newton step (that is, setting $r = 0$ and $\sigma = 0$ in (2.10)). If this step makes good progress in reducing μ, we choose σ_k small so that the step actually taken is quite close to this pure Newton step. Otherwise, we enforce more centering and calculate a conservative direction by setting σ_k closer to 1. The perturbation vector r^k is chosen

to improve the similarity between system (2.10) and the original system (2.8) that it approximates. Gondzio's technique further enhances r^k by performing further solves of the system (2.10) with a variety of right-hand sides, where each solve reuses the factorization of the matrix and is therefore not too expensive to perform.

To turn this basic algorithmic approach into a useful piece of software, we must address many issues. These include problem formulation, presolving to reduce the problem size, choice of the step length, linear algebra techniques for solving (2.10), and user interfaces and input formats.

Possibly, the most interesting issues are associated with the linear algebra. Most codes deal with a partially eliminated form of (2.10), either eliminating Δs to obtain

$$\begin{bmatrix} 0 & A \\ A^{\mathrm{T}} & -X^{-1}S \end{bmatrix} \begin{bmatrix} \Delta\lambda \\ \Delta x \end{bmatrix} = - \begin{bmatrix} 0 \\ -X^{-1}(XSe - \sigma\mu e + r) \end{bmatrix} \tag{2.16}$$

or eliminating both Δs and Δx to obtain a system of the form

$$A(S^{-1}X)A^{\mathrm{T}}\Delta\lambda = t, \tag{2.17}$$

to which a sparse Cholesky algorithm is applied. A modified version of the latter form is used when dense columns are present in A. These columns may be treated as a low-rank update and handled via the Sherman–Morrison–Woodbury formula or, equivalently, via a Schur complement strategy applied to a system intermediate between (2.16) and (2.17). In many problems, the matrix in (2.17) becomes increasingly ill-conditioned as the iterates progress, eventually causing the Cholesky process to break down as negative pivot elements are encountered. A number of simple (and in some cases counterintuitive) patches have been proposed for overcoming this difficulty while still producing useful approximate solutions of (2.17) efficiently.

Despite many attempts, iterative solvers have not shown much promise as means to solve (2.17), at least for general linear programs. A possible reason is that, besides its poor conditioning, the matrix lacks the regular spectral properties of matrices obtained from discretizations of continuous operators. Some codes do, however, use preconditioned conjugate gradient as an alternative to iterative refinement for improving the accuracy, when the direct approach for solving (2.17) fails to produce a solution of sufficient accuracy. The preconditioner used in this case is simply the computed factorization of the matrix $A(S^{-1}X)A^{\mathrm{T}}$.

A number of interior-point linear programming codes are now available, both commercially and free of charge. Information can be obtained from the World-Wide Web via the URL mentioned earlier. It is difficult to make blanket statements about the relative efficiency of interior-point and simplex methods for linear programming, since significant improvements to the implementations of both techniques continue to be made. Interior-point methods tend to be faster on large problems and can better exploit multiprocessor platforms, because the expensive operations such as Cholesky factorization of (2.17) can be parallelized to some extent. They are not able to exploit "warm start" information — a good prior estimate of the solution, for instance, — to the same extent as simplex methods. For this reason, they are not well suited for use in contexts such as branch-and-bound or branch-and-cut algorithms for integer programming, which solve many closely related linear programs.

Several researchers have devised special interior-point algorithms for special cases of (2.1) that exploit the special properties of these cases in solving the linear systems at each iteration. One algorithm for network flow problems uses preconditioned conjugate–gradient methods for solving

(2.17), where the preconditioner is built from a spanning tree for the underlying network. For multicommodity flow problems, there is an algorithm for solving a version of (2.17) in which the block-diagonal part of the matrix is used to eliminate many of the variables, and a preconditioned conjugate–gradient method is applied to the remaining Schur complement. Various techniques have also been proposed for stochastic programming (two-stage linear problems with recourse) that exploit the problem structure in performing the linear algebra operations.

3. Extensions to convex quadratic programming and linear complementarity

The primal–dual algorithms of the preceding section are readily extended to convex quadratic programming (QP) and monotone linear complementarity problems (LCP), both classes being generalizations of linear programming. Indeed, many of the convergence and complexity properties of primal–dual algorithm were first elucidated in the literature with regard to monotone LCP rather than linear programming.

We state the convex QP as

$$\min_x \; c^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Qx \quad \text{s.t. } Ax = b, \quad x \geqslant 0, \tag{3.18}$$

where Q is a positive-semi-definite matrix. The monotone LCP is defined by square matrices M and N and a vector q, where M and N satisfy a monotonicity property: all vectors y and z that satisfy $My + Nz = 0$ have $y^{\mathrm{T}}z \geqslant 0$. This problem requires us to identify vectors y and z such that

$$My + Nz = q, \quad (y,z) \geqslant 0, \quad y^{\mathrm{T}}z = 0. \tag{3.19}$$

With some transformations, we can express the optimality conditions (2.7) for linear programming, and also the optimality conditions for (3.18), as a monotone LCP. Other problems fit under the LCP umbrella as well, including bimatrix games and equilibrium problems. The central path for this problem is defined by the following system, parametrized as in (2.8) by the positive scalar μ:

$$My + Nz = q, \tag{3.20a}$$

$$YZe = \mu e, \tag{3.20b}$$

$$(y,z) > 0 \tag{3.20c}$$

and a search direction from a point (y,z) satisfying (3.20a) and (3.20c) is obtained by solving a system of the form

$$\begin{bmatrix} M & N \\ Z & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix} = - \begin{bmatrix} 0 \\ YZe - \sigma\mu e + r \end{bmatrix}, \tag{3.21}$$

where $\mu = y^{\mathrm{T}}z/n$, $\sigma \in [0,1]$, and, as before, r is a perturbation term. The corresponding search direction system for the quadratic program (3.18) is identical to (2.10) except that the $(2,2)$ block in the coefficient matrix is replaced by Q. The primal–dual algorithmic framework and the many

variations within this framework are identical to the case of linear programming, with the minor difference that the step length should be the same for all variables. (In linear programming, different step lengths usually are taken for the primal variable x and the dual variables (λ, s).)

Complexity results are also similar to those obtained for the corresponding linear programming algorithm. For an appropriately chosen starting point (y^0, z^0) with $\mu_0 = (y^0)^{\mathrm{T}} z^0/n$, we obtain convergence to a point with $\mu \leqslant \varepsilon$ in

$$\mathrm{O}\left(n^{\tau} \log \frac{\mu_0}{\varepsilon}\right) \text{ iterations,}$$

where $\tau = \frac{1}{2}$, 1, or 2, depending on the algorithm. Fast local convergence results typically require an additional strict complementarity assumption that is automatically satisfied in the case of linear programming. Some authors have proposed superlinear algorithms that do not require strict complementarity, but these methods require accurate identification of the set of degenerate indices before the fast convergence becomes effective.

The LCP algorithms can, in fact, be extended to a wider class of problems involving the so-called sufficient matrices. Instead of requiring M and N to satisfy the monotonicity property defined above, we require there to exist a nonnegative constant κ such that

$$y^T z \geqslant -4\kappa \sum_{i \mid y_i z_i > 0} y_i z_i \quad \text{for all } y, z \text{ with } My + Nz = 0.$$

The complexity estimates for interior-point methods applied to such problems depends on the parameter κ, so that the complexity is not polynomial on the whole class of sufficient matrices. Potra and Sheng [19] propose a large-step infeasible-interior-point method for solving $P_*(\kappa)$-matrix linear complementarity problems with a number of strong properties. The algorithm generates points in a large neighborhood of an infeasible central path, and each iteration requires only one matrix factorization. If the problem has a solution, the algorithm converges from an arbitrary positive starting point. The computational complexity of the algorithm depends on the quality of the starting point. If a well centered starting point is feasible or close to being feasible, it has $\mathrm{O}((1+\kappa)\sqrt{n}L)$-iteration complexity. In cases in which such a starting point is not readily available, a modified version of the algorithm terminates in $\mathrm{O}((1+\kappa)^2 nL)$ steps either by finding a solution or by determining that the problem is not solvable. Finally, high-order local convergence is proved for problems having a strictly complementary solution. We note that while the properties of the algorithm (e.g. computational complexity) depend on κ, the algorithm itself does not.

Primal–dual methods have been applied to many practical applications of (3.18) and (3.19), including portfolio optimization, optimal control, and ℓ_1 regression (see [28] for references).

The interior-point approach has a number of advantages over the active-set approach from a computational point of view. It is difficult for an active-set algorithm to exploit any structure inherent in both Q and A without redesigning most of its complex linear algebra operations: the operations of adding a constraint to the active set, deleting a constraint, evaluating Lagrange multiplier estimates, calculating the search direction, and so on. In the interior-point approach, on the other hand, the only complex linear algebra operation is solution of the linear system (3.21) — and this operation, though expensive, is relatively straightforward. Since the structure and dimension of the linear system remain the same at all iterations, the routines for solving the linear systems can exploit fully the properties of the systems arising from each problem class or instance. In fact, the algorithm can

be implemented to high efficiency using an object-oriented approach, in which the implementer of each new problem class needs to supply only code for the factorization and solution of the systems (3.21), optimized for the structure of the new class, along with a number of simple operations such as inner-product calculations. Code that implements upper-level decisions (choice of parameter σ, vector r, steplength α) remains efficient across the gamut of applications of (3.19) and can simply be reused by all applications.

We note, however, that active-set methods would still require much less execution time than interior-point methods in many contexts, especially when "warm start" information is available and when the problem is generic enough that not much benefit is gained by exploiting its structure.

The extension of primal–dual algorithms from linear programming to convex QP is so straightforward that a number of the interior-point linear programming codes have recently been extended to handle problems in the class (3.18) as well. In their linear algebra calculations, most of these codes treat both Q and A as general sparse matrices, and hence are efficient across a wide range of applications. By contrast, implementations of active-set methods for (3.18) that are capable of handling even moderately sized problems have not been widely available.

4. Semi-definite programming

As mentioned in the introduction, semi-definite programming (SDP) has been one of the most active areas of optimization research in the 1990s. SDP consists in minimizing a linear functional of a matrix subject to linear equality and inequality constraints, where the inequalities include membership of the cone of positive-semi-definite matrices. SDP is a broad paradigm; it includes as special cases linear programming, (linearly constrained) QP, quadratically constrained QP and other optimization problems (see [16,25]). Semi-definite programming has numerous applications in such diverse areas as optimal control, combinatorial optimization, structural optimization, pattern recognition, trace factor analysis in statistics, matrix completions, etc. See the excellent survey paper [25] for some instances. It was only after the advent of interior-point methods, however, that efficient solution methods for SDP problems were available. During the past few years an impressive number of interior-point methods for SDP have been proposed. Some of them have been successfully implemented and used to solve important application problems. However the theory and practice of interior-point methods for SDP has not yet reached the level of maturity of interior-point methods for LP, QP, and LCP. One reason that the study of interior-point methods for SDP is extremely important is that while LP, QP, and LCP can also be solved by other methods (e.g. the simplex method or Lemke's method), interior-point methods appear to be the only efficient methods for solving general SDP problems presently known.

To define the SDP, we introduce the notation $\mathscr{S}\mathbb{R}^{n\times n}$ to represent the set of $n \times n$ symmetric matrices, and the inner product $X \bullet Z$ of two matrices in this set, which is defined as

$$X \bullet Z = \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} z_{ij}.$$

The SDP in standard form is then

$$\min_{X} C \bullet X \quad \text{s.t. } X \succcurlyeq 0,\ A_i \bullet X = b_i,\ i = 1,2,\ldots,m, \tag{4.22}$$

where $X \in \mathcal{S}\mathbb{R}^{n\times n}$, and its associated dual problem is

$$\max_{y,S} b^{\mathrm{T}}\lambda \quad \text{s.t.} \sum_{i=1}^{m} \lambda_i A_i + S = C,\ S \succcurlyeq 0, \tag{4.23}$$

where $S \in \mathcal{S}\mathbb{R}^{n\times n}$ and $\lambda \in \mathbb{R}^m$.

In what follows, we will consider only primal–dual interior-point methods that simultaneously solve the primal and dual problems. Points on the central path for (4.22), (4.23) are defined by the following parametrized system:

$$\sum_{i=1}^{m} \lambda_i A_i + S = C, \tag{4.24a}$$

$$A_i \bullet X = b_i, \quad i = 1,2,\ldots,m, \tag{4.24b}$$

$$XS = \mu I, \tag{4.24c}$$

$$X \succcurlyeq 0, \quad S \succcurlyeq 0, \tag{4.24d}$$

where as usual μ is the positive parameter. Unlike the corresponding equations for linear programming, system (4.24a), (4.24b), (4.24c) is not quite "square", since the variables reside in the space $\mathcal{S}\mathbb{R}^{n\times n} \times \mathbb{R}^m \times \mathcal{S}\mathbb{R}^{n\times n}$ while the range space of the equations is $\mathcal{S}\mathbb{R}^{n\times n} \times \mathbb{R}^m \times \mathbb{R}^{n\times n}$. In particular, the product of two symmetric matrices (see (4.24c)) is not necessarily symmetric. Before Newton's method can be applied the domain and range have to be reconciled. The various primal–dual algorithms differ partly in the manner in which they achieve this reconciliation.

The paper of Todd [24] is witness to the intensity of research in SDP interior-point methods: It describes 20 techniques for obtaining search directions for SDP, among the most notable being the following:

(1) the AHO search direction proposed by Alizadeh, Haeberly and Overton;
(2) the KSH/HRVW/M search direction independently proposed by Kojima, Shindoh and Hara; Helmberg, Rendl, Vanderbei and Wolkowicz; and later rediscovered by Monteiro;
(3) the NT direction introduced by Nesterov and Todd.

Most of the search directions for SDP are obtained by replacing Eq. (4.24c) by a "symmetric" one whose range lies in $\mathcal{S}\mathbb{R}^{n\times n}$

$$\Theta(X,S) = 0. \tag{4.25}$$

Primal–dual methods are then derived as perturbed Newton's methods applied to (4.24a), (4.24b), (4.25). Examples of symmetrizations (4.25) include the Monteiro–Zhang family [15], in which

$$\Theta(X,S) = H_P(XS),$$

where

$$H_P(M) = \tfrac{1}{2}[PMP^{-1} + (PMP^{-1})^{\mathrm{T}}]$$

(with a given a nonsingular matrix $P \in \mathbb{R}^{n\times n}$) is the symmetrization operator of Zhang. The search directions (1)–(3) mentioned above are obtained by taking P equal to I, $S^{1/2}$, and $[S^{1/2}(S^{1/2}X\,S^{1/2})^{-1/2}\,S^{1/2}]^{1/2}$, respectively.

Even if the SDP has integer data, its solution cannot in general be expressed in terms of rational numbers, so that the exact solution cannot be obtained in a finite number of bit operations. We say that an interior-point method for SDP "is polynomial" if there is a positive constant ω such that the distance to optimum (or the duality gap) is reduced by a factor of $2^{-O(L)}$ in at most $O(n^{\omega}L)$ iterations. In this case, we will say that the interior-point method has $O(n^{\omega}L)$ iteration complexity. The iteration complexity appears to be dependent on the choice of search direction. The best results obtained to date show that some feasible interior-point methods based on small neighborhoods for the central path have $O(\sqrt{n}L)$ iteration complexity for all three search directions mentioned above.

Monteiro and Zhang [15] proved that algorithms acting in large neighborhoods of the central path have $O(nL)$ iteration complexity if based on the NT direction and $O(n^{3/2}L)$ if based on the KSH/HRVW/M search direction. They also gave iteration complexity bounds (which depend on the condition number of matrices J_x and J_s defined by $P^{\mathrm{T}}P = X^{-1/2}J_xX^{1/2} = S^{-1/2}J_sS^{1/2}$) for algorithms acting in the large neighborhood that are based on the MZ* family of directions. This family is a subclass of the MZ family that contains the NT and the KSH/HRVW/M directions but not the AHO direction. So far, no complexity results are known for algorithms based on the large neighborhood and the AHO direction.

The analysis of infeasible interior-point algorithms for SDP is considerably more difficult than that of their feasible counterparts. The first complexity result in this respect was obtained by Kojima, Shindoh, and Hara, who showed that an infeasible-interior-point potential reduction method for SDP has $O(n^{5/2}L)$ iteration complexity. Subsequently, Zhang analyzed an infeasible-interior-point method, based on the KSH/HRVW/M search direction, that has $O(n^2L)$ iteration complexity when acting in the semi-large neighborhood and $O(n^{5/2}L)$ iteration complexity in the large neighborhood of the central path. The analysis of the Mizuno–Todd–Ye predictor–corrector method for infeasible starting points was performed independently by Kojima, Shida and Shindoh and Potra and Sheng. The analysis in the latter paper shows that the iteration complexity depends on the quality of the starting point. If the problem has a solution, then the algorithm is globally convergent. If the starting point is feasible or close to feasible, the algorithms finds an optimal solution in at most $O(\sqrt{n}L)$ iterations. If the starting point is large enough according to some specific criteria, then the algorithm terminates in at most $O(nL)$ steps either by finding a strictly complementary solution or by determining that the primal–dual problem has no solution of norm less than a specified size.

Superlinear convergence is especially important for SDP since no finite termination schemes exist for such problems. As predicted by theory and confirmed by numerical experiments, the condition number of the linear systems defining the search directions increases like $1/\mu$, so that the respective systems become quite ill conditioned as we approach the solution. As we observed in the case of linear programming, an interior-point method that is not superlinearly convergent is unlikely to obtain high accuracy in practice. On the other hand, superlinearly convergent interior-point methods often achieve good accuracy (duality measure of 10^{-10} or better) in substantially fewer iterations than indicated by the worse-case global linear convergence rate indicated by the analysis.

The local convergence analysis for interior-point algorithms for SDP is much more challenging than for linear programming. Kojima, Shida and Shindoh [12] established superlinear convergence of the Mizuno–Todd–Ye predictor–corrector algorithm based on the KSH/HRVW/M search direction under the following three assumptions:

(A) SDP has a strictly complementary solution;

(B) SDP is nondegenerate in the sense that the Jacobian matrix of its KKT system is nonsingular;
(C) the iterates converge tangentially to the central path in the sense that the size of the neighborhood containing the iterates must approach zero namely,

$$\lim_{k\to\infty} ||(X^k)^{1/2}S^k(X^k)^{1/2} - (X^k \bullet S^k/n)I||_F/(X^k \bullet S^k/n) = 0.$$

Assumptions (B) and (C) are quite restrictive; similar conditions are not required for the superlinear convergence of interior-point methods for linear programming or QP. Potra and Sheng [20] proved superlinear convergence of the same algorithm under assumption (A) together with the following condition:

(D) $$\lim_{k\to\infty} X^kS^k/\sqrt{X^k \bullet S^k} = 0,$$

which is clearly weaker than (C). Of course both (C) and (D) can be enforced by the algorithm, but the practical efficiency of such an approach is questionable. From a theoretical point of view, however, it is known from [20] that a modified version of the algorithm of [12] that uses several corrector steps in order to enforce (C) has polynomial complexity and is superlinearly convergent under assumption (A) only. It is well known that assumption (A) is necessary for superlinear convergence of interior-point methods that take Newton-like steps even in the QP case. (However, there are methods for convex QP and monotone LCP that attain superlinear convergence by making explicit guesses of the set of degenerate indices.)

Kojima et al. [12] also gave an example suggesting that interior-point algorithms for SDP based on the KSH/HRVW/M search direction are unlikely to be superlinearly convergent without imposing a condition like (C) or (D). In a later paper they showed that a predictor–corrector algorithm using the AHO direction is quadratically convergent under assumptions (A) and (B). They also proved that the algorithm is globally convergent, but no polynomial complexity bounds have yet been found. It appears that the use of the AHO direction in the corrector step has a strong effect on centering. This property is exploited in a recent paper of Ji et al. [10] who proved that the Mizuno–Todd–Ye algorithm, based on the MZ-family is superlinear under assumptions (A) and (D). They also showed that under assumptions (A) and (B) the algorithm has Q-order 1.5 if scaling matrices in the corrector step have bounded condition number, and Q-order 2 if the scaling matrices in both predictor and corrector step have bounded condition number. In particular, these results apply for the AHO direction, where the scaling matrix is the identity matrix. References to the results cited above can be found in [10].

Over the past several years we have witnessed an intense research effort on the use of SDP for finding approximate solution of (NP-hard) combinatorial optimization problems. In what follows, we will describe the technique of Goemans and Williamson, which yields an approximate solution whose value is within 13% of optimality for the MAX CUT problem [7].

In MAX CUT, we are presented with an undirected graph with N whose edges w_{ij} have nonnegative weights. The problem is choose a subset $\mathcal{S} \subset \{1,2,\ldots,N\}$ so that the sum of weights of the edges that cross from $\mathcal{S}$ to its complement is minimized. In other words, we aim to choose $\mathcal{S}$ to maximize the objective

$$w(\mathcal{S}) \stackrel{\text{def}}{=} \sum_{i\in\mathcal{S}, j\notin\mathcal{S}} w_{ij}.$$

This problem can be restated as an integer quadratic program by introducing variables y_i, $i = 1,2,\ldots,N$, such that $y_i = 1$ for $i \in \mathscr{S}$ and $y_i = -1$ for $i \notin \mathscr{S}$. We then have

$$\max_y \frac{1}{2}\sum_{i<j} w_{ij}(1 - y_i y_j) \quad \text{s.t. } y_i \in \{-1,1\} \text{ for all } i = 1,2,\ldots,N. \tag{4.26}$$

This problem is NP-complete. Goemans and Williamson replace the variables $y_i \in \mathbb{R}$ by vectors $v_i \in \mathbb{R}^N$ and consider instead the problem

$$\max_{v_1,v_2,\ldots,v_N} \frac{1}{2}\sum_{i<j} w_{ij}(1 - v_i^{\mathrm{T}} v_j) \quad \text{s.t. } ||v_i|| = 1 \text{ for all } i = 1,2,\ldots,N. \tag{4.27}$$

This problem is a relaxation of (4.26) because any feasible point y for (4.26) corresponds to a feasible point

$$v_i = (y_i, 0, 0, \ldots, 0)^{\mathrm{T}}, \quad i = 1,2,\ldots,N$$

for (4.27). Problem (4.27) can be formulated as an SDP by changing variables $v_1, v_2, \ldots, v_N$ to a matrix $Y \in \mathbb{R}^{N\times N}$, such that

$$Y = V^{\mathrm{T}}V \quad \text{where } V = [v_1, v_2, \ldots, v_N].$$

The constraints $||v_i|| = 1$ can be expressed simply as $Y_{ii} = 1$, and since $Y = V^{\mathrm{T}}V$, we must have Y semi-definite. The transformed version of (4.27) is then

$$\max \frac{1}{2}\sum_{i<j} w_{ij}(1 - Y_{ij}) \quad \text{s.t. } Y_{ii} = 1,\ i = 1,2,\ldots,N \quad \text{and} \quad Y \succcurlyeq 0,$$

which has the form (4.22) for appropriate definitions of C and A_i, $i = 1,2,\ldots,N$. We can recover V from Y by performing a Cholesky factorization. The final step of recovering an approximate solution to the original problem (4.26) is performed by choosing a random vector $r \in \mathbb{R}^N$, and setting

$$y_i = \begin{cases} 1 & \text{if } r^{\mathrm{T}} v_i > 0, \\ -1 & \text{if } r^{\mathrm{T}} v_i \leqslant 0. \end{cases}$$

A fairly simple geometric argument shows that the expected value of the solution so obtained has objective value at least 0.87856 of the optimal solution to (4.26).

Similar relaxations have been obtained for many other combinatorial problems, showing that is possible to find good approximate solutions to many NP-complete problems by using polynomial algorithms. Such relaxations are also useful if we seek *exact* solutions of the combinatorial problem by means of a branch-and-bound or branch-and-cut strategy. Relaxations can be solved at each node of the tree (in which some of the degrees of freedom are eliminated and some additional constraints are introduced) to obtain both a bound on the optimal solution and in some cases a candidate feasible solution for the original problem. Since the relaxations to be solved at adjacent nodes of the tree are similar, it is desirable to use solution information at one node to "warm start" the SDP algorithm at a child node.

5. Convex programming

One of the most surprising results in interior-point methods is the fact that interior-point algorithms from LP can be extended to general convex programming problems, at least in a theoretical

sense. The key to such an extension was provided in [16]. These authors explored the properties of self-concordant functions, and described techniques in which the inequality constraints in a convex programming problem are replaced by self-concordant barrier terms in the objective function. They derived polynomial algorithms by applying Newton-like methods to the resulting parametrized reformulations.

The fundamental property of self-concordant functions is that their third derivative can be bounded by some expression involving their second derivative at each point in their domain. This property implies that the second derivative does not fluctuate too rapidly in a relative sense, so that the function does not deviate too much from the second-order approximation on which Newton's method is based. Hence, we can expect Newton's method to perform reasonably well on such a function.

Given a finite-dimensional real vector space $\mathscr{V}$, an open, nonempty convex set $\mathscr{S} \subset \mathscr{V}$, and a closed convex set $\mathscr{T} \subset \mathscr{V}$ with nonempty interior, we have the following formal definition.

Definition 1. The function $F : \mathscr{S} \to \mathbb{R}$ is *self-concordant* if it is convex and if the following inequality holds for all $x \in \mathscr{S}$ and all $h \in \mathscr{V}$:

$$|D^3F(x)[h,h,h]| \leqslant 2(D^2F(x)[h,h])^{3/2}, \tag{5.28}$$

where $D^kF[h_1, h_2, \ldots, h_k]$ denotes the kth differential of F along the directions $h_1, h_2, \ldots, h_k$.

F is called *strongly self-concordant* if $F(x_i) \to \infty$ for all sequences $x_i \in \mathscr{S}$ that converge to a point on the boundary of $\mathscr{S}$.

F is a *ϑ-self-concordant barrier* for $\mathscr{T}$ if it is a strongly self-concordant function for $\operatorname{int} \mathscr{T}$, and the parameter

$$\vartheta \stackrel{\text{def}}{=} \sup_{x \in \operatorname{int} \mathscr{T}} F'(x)^{\mathrm{T}}[F''(x)]^{-1}F'(x) \tag{5.29}$$

is finite.

Note that the exponent $\frac{3}{2}$ on the right-hand side of (5.28) makes the condition independent of the scaling of the direction h. It is shown in [16, Corollary 2.3.3], that if $\mathscr{T} \neq \mathscr{V}$, then the parameter ϑ is no smaller than 1.

It is easy to show that log-barrier function of Section 2 is an n-self-concordant barrier for the positive orthant $\mathbb{R}^n_+$ if we take

$$\mathscr{V} = \mathbb{R}^n, \ \mathscr{T} = \mathbb{R}^n_+, \ F(x) = -\sum_{i=1}^{n} \log x_i.$$

where $\mathbb{R}^n_+$ denotes the positive orthant. Another interesting case is the second-order cone (or "ice-cream cone"), for which we have

$$\mathscr{V} = \mathbb{R}^{n+1}, \qquad \mathscr{T} = \{(x,t) \,|\, ||x||_2 \leqslant t\}, \qquad F(x,t) = -\log(t^2 - ||x||^2), \tag{5.30}$$

where $t \in \mathbb{R}$ and $x \in \mathbb{R}^n$. In this case, F is a two-self-concordant barrier. Second-order cone programming consists in minimizing a linear function subject to linear equality constraints together with inequality constraints induced by second-order cones. Convex quadratically constrained quadratic programs can be posed in this form, along with sum-of-norms problems and many other applications.

A third important case is the cone of positive-semi-definite matrices, for which we have

$\mathscr{V} = n \times n$ symmetric matrices,

$\mathscr{T} = n \times n$ symmetric positive-semi-definite matrices,

$F(X) = -\log \det X$

for which F is an n-self-concordant barrier. This barrier function can be used to model the constraint $X \succcurlyeq 0$ in (4.22).

Self-concordant barrier functions allow us to generalize the primal barrier method of Section 2 to problems of the form

$$\min \langle c, x \rangle \quad \text{s.t. } Ax = b, \ x \in \mathscr{T}, \tag{5.31}$$

where $\mathscr{T}$ is a closed convex set, $\langle c, x \rangle$ denotes a linear functional on the underlying vector space $\mathscr{V}$, and A is a linear operator. Similarly to (2.2), we define the barrier subproblem to be

$$\min_x f(x;\mu) \stackrel{\text{def}}{=} \frac{1}{\mu}\langle c, x \rangle + F(x) \quad \text{s.t. } Ax = b, \tag{5.32}$$

where $F(x)$ is a self-concordant barrier and $\mu > 0$ is the barrier parameter. Note that by the Definition 1, $f(x;\mu)$ is also a strongly self-concordant function. The primal barrier algorithm for (5.31) based on (5.32) is as follows:

primal barrier algorithm
Given $x^0 \in \operatorname{int} \mathscr{T}$ and $\mu_0 > 0$;
Set $k \leftarrow 0$;
repeat
    Obtain $x^{k+1} \in \operatorname{int} \mathscr{T}$ by performing one or more projected Newton steps
        for $f(\cdot;\mu_k)$, starting at $x = x^k$;
    Choose $\mu_{k+1} \in (0, \mu_k)$;
until some termination test is satisfied.

As in Sections 2–4, the worst-case complexity of algorithms of this type depends on the parameter ϑ associated with F but not on any properties of the data that defines the problem instance. For example, we can define a short-step method in which a single full Newton step is taken for each value of k, and μ is decreased according to

$$\mu_{k+1} = \mu_k \Big/ \left(1 + \frac{1}{8\sqrt{\vartheta}}\right).$$

Given a starting point with appropriate properties, we obtain an iterate x^k whose objective $\langle c, x^k \rangle$ is within ε of the optimum in

$$\mathrm{O}\left(\sqrt{\vartheta} \log \frac{\vartheta \mu_0}{\varepsilon}\right) \text{ iterations.}$$

Long-step variants are discussed in [16]. The practical behavior of the methods does, of course, depend strongly on the properties of the particular problem instance.

The primal–dual algorithms of Section 2 can also be extended to more general problems by means of the theory of self-scaled cones developed in [17,18]. The basic problem considered is the conic programming problem

$$\min\langle c,x\rangle \quad \text{s.t. } Ax=b,\ x\in K, \tag{5.33}$$

where $K\subset\mathbb{R}^n$ is a closed convex cone, that is, a closed convex set for which $x\in K\Rightarrow tx\in K$ for all nonnegative scalars t, and A denotes a linear operator from $\mathbb{R}^n$ to $\mathbb{R}^m$. The dual cone for K is denoted by K^* and defined as

$$K^*\stackrel{\text{def}}{=}\{s\mid\langle s,x\rangle\geqslant 0 \text{ for all } x\in K\}$$

and we can write the dual instance of (5.33) as

$$\max\langle b,\lambda\rangle \quad \text{s.t. } A^*\lambda+s=c,\ \ s\in K^*, \tag{5.34}$$

where A^* denotes the adjoint of A. The duality relationships between (5.33) and (5.34) are more complex than in linear programming, but if either problem has a feasible point that lies in the interior of K or K^*, respectively, the strong duality property holds. This property is that when the optimal value of either (5.33) or (5.34) is finite, then both problems have finite optimal values, and these values are the same.

K is a self-scaled cone when its interior $\operatorname{int}K$ is the domain of a self-concordant barrier function F with certain strong properties that allow us to define algorithms in which the primal and dual variables are treated in a perfectly symmetric fashion and play interchangeable roles. The full elucidation of these properties is quite complicated. It suffices to note here that the three cones mentioned above – the positive orthant $\mathbb{R}^n_+$, the second-order cone (5.30), and the cone of positive-semi-definite symmetric matrices – are the most interesting self-scaled cones, and their associated barrier functions are the logarithmic functions mentioned above.

To build algorithms from the properties of self-scaled cones and their barrier functions, the Nesterov–Todd theory defines a *scaling point* for a given pair $x\in\operatorname{int}K$, $s\in\operatorname{int}K^*$ to be the unique point w such that $H(w)x=s$, where $H(\cdot)$ is the Hessian of the barrier function. In the case of linear programming, it is easy to verify that w is the vector in $\mathbb{R}^n$ whose elements are $\sqrt{x_i/s_i}$. The Nesterov–Todd search directions are obtained as projected steepest descent direction for the primal and dual barrier subproblems (that is, (5.32) and its dual counterpart), where a weighted inner product involving the matrix $H(w)$ is used to define the projections onto the spaces defined by the linear constraints $Ax=b$ and $A^*\lambda+s=c$, respectively. The resulting directions satisfy the following linear system:

$$\begin{bmatrix}0 & A & 0\\ A^* & 0 & I\\ 0 & H(w) & I\end{bmatrix}\begin{bmatrix}\Delta\lambda\\ \Delta x\\ \Delta s\end{bmatrix}=-\begin{bmatrix}0\\ 0\\ s+\sigma\mu\nabla F(x)\end{bmatrix}, \tag{5.35}$$

where $\mu=\langle x,s\rangle/\vartheta$. (The correspondence with (2.10) is complete if we choose the perturbation term to be $r=0$.) By choosing the starting point appropriately, and designing schemes to choose the parameters σ and step lengths along these directions, we obtain polynomial algorithms for this general setting. The NT direction in the previous section is the specialization of the above search directions for semi-definite programming.

6. Conclusions

Interior-point methods remain an active and fruitful area of research, although the frenetic pace that characterized the area has slowed in recent years. Interior-point codes for linear programming codes have become mainstream and continue to undergo development, although the competition from the simplex method is stiff. Semi-definite programming has proved to be an area of major impact. Applications to quadratic programming show considerable promise, because of the superior ability of the interior-point approach to exploit problem structure efficiently. The influence on nonlinear programming theory and practice has yet to be determined, even though significant research has already been devoted to this topic. Use of the interior-point approach in decomposition methods appears promising, though no rigorous comparative studies with alternative approaches have been performed. Applications to integer programming problems have been tried by a number of researchers, but the interior-point approach is hamstrung here by competition from the simplex method with its superior warm-start capabilities.

Acknowledgements

This work was supported in part by NSF under grant DMS-9996154 and by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

We are grateful to an anonymous referee for a speedy but thorough review.

References

[1] K.M. Anstreicher, Linear programming in $O([n^3/\ln n]L)$ operations, CORE Discussion Paper 9746, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, January 1999, SIAM J. Optim., in preparation.

[2] K.M. Anstreicher, J. Ji, F.A. Potra, Y. Ye, Average performance of a self-dual interior-point algorithm for linear programming, in: P. Pardalos (Ed.), Complexity in Numerical Optimization, World Scientific, Singapore, 1993, pp. 1–15.

[3] K.H. Borgwardt, The Simplex Method: A Probabilistic Analysis, Springer, Berlin, 1987.

[4] A.S. El-Bakry, R.A. Tapia, Y. Zhang, A study of indicators for identifying zero variables in interior-point methods, SIAM Rev. 36 (1) (1994) 45–72.

[5] A.V. Fiacco, G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, New York, 1968 (reprinted by SIAM, Philadelphia, PA, 1990).

[6] R.M. Freund, S. Mizuno, Interior point methods: current status and future directions, Optima 51 (1996) 1–9.

[7] M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, J. Assoc. Comput. Mach. 42 (6) (1995) 1115–1145.

[8] D. Goldfarb, M.J. Todd, Linear programming, in: G.L. Nemhauser, A.H.G. Rinnooy Kan, M.J. Todd (Eds.), Optimization, North-Holland, Amsterdam, 1989, pp. 73–170.

[9] J. Gondzio, Multiple centrality corrections in a primal–dual method for linear programming, Comput. Optim. Appl. 6 (1996) 137–156.

[10] J. Ji, F.A. Potra, R. Sheng, On the local convergence of a predictor–corrector method for semidefinite programming, SIAM J. Optim. 10 (1999) 195–210.

[11] N. Karmarkar, A new polynomial-time algorithm for linear programming, Combinatorica 4 (1984) 373–395.

[12] M. Kojima, M. Shida, S. Shindoh, Local convergence of predictor–corrector infeasible-interior-point algorithms for SDPs and SDLCPs, Math. Programming, Ser. A 80 (2) (1998) 129–160.

[13] N. Megiddo, Pathways to the optimal set in linear programming, in: N. Megiddo (Eds.), Progress in Mathematical Programming: Interior-Point and Related Methods Springer, New York, 1989, pp. 131–158 (Chapter 8).
[14] S. Mehrotra, On the implementation of a primal–dual interior point method, SIAM J. Optim. 2 (1992) 575–601.
[15] R.D.C. Monteiro, Y. Zhang, A unified analysis for a class of long-step primal–dual path-following interior-point algorithms for semidefinite programming, Math. Programming Ser. A 81 (3) (1998) 281–299.
[16] Yu.E. Nesterov, A.S. Nemirovskii, Interior Point Polynomial Methods in Convex Programming: Theory and Applications, SIAM, Philadelphia, PA, 1994.
[17] Yu.E. Nesterov, M.J. Todd, Self-scaled barriers and interior-point methods for convex programming, Math. Oper. Res. 22 (1997) 1–42.
[18] Yu.E. Nesterov, M.J. Todd, Primal–dual interior-point methods for self-scaled cones, SIAM J. Optim. 8 (1998) 324–362.
[19] F.A. Potra, R. Sheng, A large-step infeasible-interior-point method for the P_*-matrix LCP, SIAM J. Optim. 7 (2) (1997) 318–335.
[20] F.A. Potra, R. Sheng, Superlinear convergence of interior-point algorithms for semidefinite programming, J. Optim. Theory Appl. 99 (1) (1998) 103–119.
[21] J. Renegar, A polynomial-time algorithm, based on Newton's method, for linear programming, Math. Programming 40 (1988) 59–93.
[22] J. Renegar, A mathematical view of interior-point methods in convex optimization, unpublished notes, June 1999.
[23] C. Roos, J.-Ph. Vial, T. Terlaky, Theory and Algorithms for Linear Optimization: An Interior Point Approach, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, 1997.
[24] M.J. Todd, A study of search directions in primal–dual interior-point methods for semidefinite programming, Technical Report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, February 1999.
[25] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Rev. 38 (1) (1996) 49–95.
[26] M.H. Wright, Interior methods for constrained optimization, in: Acta Numer. 1992, Cambridge University Press, Cambridge, 1992, pp. 341–407.
[27] S.J. Wright, Primal–Dual Interior-Point Methods, SIAM, Philadelphia, PA, 1997.
[28] S.J. Wright, Recent developments in interior-point methods, preprint ANL/MCS-P783-0999, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, September 1999.
[29] Y. Ye, Interior Point Algorithms: Theory and Analysis, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, New York, 1997.
[30] Y. Ye, O. Güler, R.A. Tapia, Y. Zhang, A quadratically convergent $O(\sqrt{n}L)$-iteration algorithm for linear programming, Math. Programming Ser. A 59 (1993) 151–162.
[31] Y. Ye, M.J. Todd, S. Mizuno, An $O(\sqrt{n}L)$-iteration homogeneous and self-dual linear programming algorithm, Math. Oper. Res. 19 (1994) 53–67.
[32] Y. Zhang, On the convergence of a class of infeasible-interior-point methods for the horizontal linear complementarity problem, SIAM J. Optim. 4 (1994) 208–227.