Noname manuscript No.
(will be inserted by the editor)

A Penalty-Interior-Point Algorithm for Nonlinear Constrained Optimization

Frank E. Curtis

April 26, 2011

Abstract Penalty and interior-point methods for nonlinear optimization problems have enjoyed great successes for decades. Penalty methods have proved to be effective for a variety of problem classes due to their regularization effects on the constraints. They have also been shown to allow for rapid infeasibility detection. Interior-point methods have become the workhorse in large-scale optimization due to their Newton-like qualities, both in terms of their scalability and convergence behavior. Each of these two strategies, however, have certain disadvantages that make their use either impractical or inefficient for certain classes of problems.

The goal of this paper is to present a penalty-interior-point method that possesses the advantages of penalty and interior-point techniques, but does not suffer from their disadvantages. Numerous attempts have been made along these lines in recent years, each with varying degrees of success. The novel feature of the algorithm in this paper is that our focus is not only on the formulation of the penalty-interior-point subproblem itself, but on the design of updates for the penalty and interior-point parameters. The updates we propose are designed so that rapid convergence to a solution of the nonlinear optimization problem or an infeasible stationary point is attained. We motivate the convergence properties of our algorithm and illustrate its practical performance on a large set of problems, including sets of problems that exhibit degeneracy or are infeasible.

Keywords nonlinear optimization, large-scale optimization, nonconvex optimization, constrained optimization, penalty methods, interior-point methods, penalty-interior-point methods

Mathematics Subject Classification (2000) 49M05 · 49M15 · 49M20 · 49M29 · 49M37 · 65F05 · 65F50 · 65K05 · 65K10 · 90C06 · 90C26 · 90C30 · 90C51

Frank E. Curtis
Dept. of Industrial & Systems Engineering, Lehigh University, Bethlehem, PA 18018, USA.
E-mail: frank.e.curtis@gmail.com

1 Introduction

We present an algorithm for nonlinear optimization problems of the form

$$\begin{aligned} &\min_x \ f(x) \\ &\text{subject to (s.t.) } c(x) \leq 0, \end{aligned} \tag{1.1}$$

where the objective $f : \mathbb{R}^n \to \mathbb{R}$ and the constraints $c : \mathbb{R}^n \to \mathbb{R}^t$ are continuously differentiable. Due to subsequent regularization techniques discussed in this paper, the solution method we describe for problem (1.1) can also be applied to problems with equality constraints $c^{\mathcal{E}}(x) = 0$, either by treating them as the pair of inequalities $c^{\mathcal{E}}(x) \leq 0$ and $-c^{\mathcal{E}}(x) \leq 0$ or, better yet, by relaxing them with a sufficient number of slack variables. For simplicity in our presentation, we focus on the inequality constrained case, but further details pertaining to our handling of equality constraints are presented in §4.

The main purpose of this paper is to describe a method for the solution of (1.1) that confronts two key challenges. First, we are interested in the solution of large-scale problems, and so solution techniques that require a computationally expensive procedure to be performed during each iteration (such as the solution of a linear or quadratic optimization subproblem) are inefficient for our purposes. Second, we aim for the method to be sufficiently regularized so that it may efficiently handle difficult constraint sets. This classification includes problems with constraint redundancies and (near) singularities in the constraint Jacobian, but in the extreme case we also aim for the method to be able to handle infeasible instances of (1.1). In these latter cases, a certificate of infeasibility can be returned to the user if the algorithm minimizes a feasibility violation measure such as

$$\min_x \ v(x) := \sum_{i=1}^{t} \max\{c^i(x), 0\} \tag{1.2}$$

at a point that does not satisfy the constraints of (1.1). A number of nonlinear optimization algorithms provide convergence guarantees to solutions of (1.2) if problem (1.1) is infeasible, but in many situations the rate of convergence to such points can be extremely slow. In contrast, our aim is to solve (1.1) or (1.2) *rapidly*.

One successful strategy for the solution of problem (1.1) is to employ a *penalty method*. Such an approach essentially removes the constraints of the problem and replaces them with appropriate cost terms in the objective so as to *penalize* violations in the original constraints. For example, if the cost terms correspond to an ℓ_1 norm (as in the feasibility violation measure $v(x)$ in (1.2)), then the penalty subproblem associated with (1.1) can be posed as

$$\begin{aligned} &\min_{x,s} \ \rho f(x) + \sum_{i=1}^{t} s^i \\ &\text{s.t. } c(x) - s \leq 0, \ s \geq 0, \end{aligned} \tag{1.3}$$

where $\rho \geq 0$ is a penalty parameter. A challenge in the implementation of such an approach is the design of a technique for decreasing ρ during the solution process so that (1.1), or (1.2) if (1.1) is infeasible, is solved efficiently. A second challenge is that many of the efficient implementations of penalty methods require the solution

of generally constrained linear or quadratic optimization subproblems numerous times throughout the solution process; e.g., see [5,7–9,19]. This fact makes such approaches intractable for many large-scale applications.

Interior-point methods present a more efficient alternative to penalty methods for the solution of large-scale problems. The methodology in such a strategy is to reduce the problem to one with only equality constraints through the addition of slack variables that are forced to remain *interior* to the non-negative orthant. The problem is then often reformulated as the logarithmic barrier subproblem

$$\begin{aligned} &\min_{x,r} \; f(x) - \mu \sum_{i=1}^{t} \ln r^i \\ &\text{s.t. } c(x) + r = 0 \quad (\text{with } r > 0) \end{aligned} \tag{1.4}$$

for $\mu > 0$. Similar to penalty methods, a challenge in the implementation of an interior-point method is the design of an effective strategy for updating μ, which must be driven to zero during the solution process to yield a solution to (1.1). A second challenge is that interior-point methods often lack a sophisticated mechanism for regularizing the constraints that is present in, say, a penalty approach. As a result, they can perform poorly on problems with difficult constraint sets.

In this paper, we motivate and describe an algorithm that employs both penalty and interior-point strategies. The hope is that by combining penalty and interior-point techniques in an appropriate manner, we may successfully maintain their advantages, but not suffer from their disadvantages. We join components of (1.3) and (1.4) to form the *penalty-interior-point* subproblem

$$\begin{aligned} &\min_{z} \; \phi(z;\rho,\mu) := \rho f(x) - \mu \sum_{i=1}^{t} (\ln r^i + \ln s^i) + \sum_{i=1}^{t} s^i \\ &\text{s.t. } \theta(z) := c(x) + r - s = 0 \quad (\text{with } (r,s) > 0) \end{aligned} \tag{1.5}$$

that is to be solved for the variables in $z := (x, r, s)$. This subproblem may be derived, for example, by applying an interior-point reformulation to the penalty subproblem (1.3) and then eliminating trivial equality constraints. Notice that if $\rho > 0$, $\mu = 0$, and a solution to (1.5) has $s = 0$, then it corresponds to a solution of the nonlinear optimization problem (1.1). Similarly, if $\rho = 0$ and $\mu = 0$, then a solution to (1.5) corresponds to a solution of the feasibility problem (1.2).

Our framework is not without significant challenges. Indeed, a daunting task in any approach that considers solving subproblem (1.5) in order to produce a solution to (1.1) or (1.2) is the design of an update strategy for *two* parameters: the penalty parameter ρ and the interior-point parameter μ. However, we believe that we have designed an approach that is both intuitively appealing and yields solid preliminary performance results. Rather than tackle updates for ρ and μ simultaneously, we have crafted a mechanism that updates their values sequentially during each iteration at the modest cost of only a few matrix-vector operations. Our numerical experiments illustrate that our updating scheme consistently results in fewer iterations than a more conservative strategy that only updates ρ and μ by monitoring problem function values over the course of numerous iterations.

A number of related penalty-interior-point-like algorithms have previously been proposed, analyzed, and tested. The first such methods that in some way resemble our approach were the *modified barrier* methods proposed in [23,27] (see also

[28]). These algorithms, originally developed to eradicate the ill-conditioning effects caused by classical logarithmic-barrier functions applied to inequality constrained problems, essentially incorporate Lagrange multiplier estimates to play the role of penalty parameters within a logarithmic barrier term. See also [15] for an extension of these methods to handle equality constraints through the use of augmented Lagrangians. More recent methods that also resemble our techniques are the *penalty-barrier* methods in [3,4], the *exterior-point* approach discussed in [29], the *interior-point* ℓ_1*-penalty* method proposed in [17], the *interior-point* ℓ_2*-penalty* method in [10,11], and the *penalty interior-point* algorithm in [1]. These algorithms (and ours) differ in detail, though they are all inspired by the benefits of regularization through penalties and the prospect of search direction calculations through linear system solves as in many interior-point methods.

The outline of this paper is as follows. In §2 we present theoretical foundations for our algorithm by illustrating that solving the penalty-interior-point subproblem (1.5) is an appropriate means for solving problem (1.1) or, if a feasible solution is not found, problem (1.2). We then develop and present our algorithm in §3. Numerical results on a large collection of nonlinear optimization test problems are provided in §4 and concluding remarks are presented in §5.

Notation: We use superscripts to denote (sets of) elements of a vector and subscripts to denote iteration numbers of an algorithm. Vectors comprised of stacked subvectors are written as ordered lists of subvectors; e.g., by $z = (x, r, s)$ we mean $z = [x^T \ r^T \ s^T]^T$. Displacements in variables are denoted by placing Δ before the variable name, and for the displacement corresponding to a particular iteration k, the subscript follows the variable name. For example, the step in x computed during iteration k is denoted as Δx_k. Scalars and vectors are denoted by lowercase letters, matrices are denoted by capital letters, and the capitalization of a vector name indicates the diagonal matrix formed by placing elements of that vector on the diagonal; e.g., $R := \text{diag}(r)$. I and e, respectively, denote the identity matrix and vector of all ones of any size, the expression $M_1 \succ M_2$ is used to indicate that the matrix $M_1 - M_2$ is positive definite, and all norms are considered ℓ_2 unless otherwise indicated. Finally, we denote the Hadamard (or Schur) product of two vectors e and s of the same length as the vector $e \circ s$, which has entries $(e \circ s)^i = e^i s^i$.

2 Theoretical Foundations

In this section, we discuss relationships between the nonlinear optimization problem (1.1), the feasibility problem (1.2), the penalty subproblem (1.3), and the penalty-interior-point subproblem (1.5). We argue that (1.3) and (1.5) have nice regularity properties no matter the properties of $c(x)$ and that solving (1.5) is an appropriate technique for solving (1.3), which in turn is appropriate for solving (1.1) or (1.2), depending on the selected value(s) for ρ. The results in this section are straightforward to prove and have appeared various times in the literature. Thus, formal proofs are not provided.

For now, we only make the following assumption.

Assumption 2.1 *The functions f and c are continuously differentiable on $\mathbb{R}^n$.*

Under this assumption alone, we have the following result stating that, regardless of the properties of the constraints in (1.1), the penalty and penalty-interior-

point subproblems are always regular in the sense that the Mangasarian-Fromovitz constraint qualification (MFCQ) is satisfied at *all* feasible points.

Theorem 2.2 *If $y = (x, s)$ is feasible for* (1.3)*, then MFCQ for* (1.3) *is satisfied at y. Similarly, if $z = (x, r, s)$ is feasible for* (1.5)*, then MFCQ for* (1.5) *is satisfied at z.*

Our goal now will be to argue that solving (1.5) for a sequence of interior-point parameters such that $\mu \to 0$ is an effective means for solving (1.3). Indeed, this should already be evident if one views (1.5) as the interior-point subproblem for (1.3), but our remarks highlight restrictions that should be placed on the slack variables and on the Lagrange multipliers during the optimization process. We require specifications of first-order optimality conditions for subproblems (1.3) and (1.5), which are given below. In fact, first-order optimality conditions for (1.3) include those for the feasibility problem (1.2) as a special case.

Definition 2.3 A point $(x_\rho, s_\rho, \lambda_\rho)$ is first-order optimal for (1.3) if it is feasible, satisfies the bounds $s_\rho \geq 0$ and $0 \leq \sigma_\rho \leq e$, and satisfies

$$\rho \nabla f(x_\rho) + \nabla c(x_\rho)\sigma_\rho = 0 \tag{2.1a}$$

$$\sigma_\rho \circ (c(x_\rho) - s_\rho) = 0 \tag{2.1b}$$

$$(e - \sigma_\rho) \circ s_\rho = 0. \tag{2.1c}$$

In particular, $(x_\times, s_\times, \sigma_\times)$ is first-order optimal for (1.2) if the above hold for $\rho = 0$.

Definition 2.4 A point (z, λ) is first-order optimal for (1.5) if $(r, s) > 0$ and

$$\rho \nabla f(x) + \nabla c(x)\lambda = 0 \tag{2.2a}$$

$$R\lambda - \mu e = 0 \tag{2.2b}$$

$$S(e - \lambda) - \mu e = 0 \tag{2.2c}$$

$$\theta(z) = 0. \tag{2.2d}$$

The following result links first-order optimal points of (1.5) and those of (1.3).

Theorem 2.5 *If for any $\rho \geq 0$ and $\mu = 0$, the point (z, λ) with $(r, s) \geq 0$ and $e \geq \lambda \geq 0$ is a first-order optimal point for* (1.5)*, then $(x_\rho, s_\rho, \sigma_\rho) = (x, s, \lambda)$ is a first-order optimal point for* (1.3)*.*

Of course, this last result considers the ideal situation in which a solution of (2.2) for $\mu = 0$ are obtained with r, s, and λ satisfying appropriate conditions. However, it is useful to state the theorem as it clearly illustrates the particular bounds that should be placed on the slack variables and Lagrange multipliers during the solution process. In particular, the slack variables should be forced to remain interior to the nonnegative orthant and each multiplier should be restricted to $(0, 1)$ so that, in the limit, a sequence of (approximate) solutions to (2.2) for $\mu \to 0$ will converge to a first-order optimal point for (1.3).

The hope in solving the penalty subproblem (1.3) is that, as long as the penalty parameter $\rho > 0$ has been set appropriately, a solution of (1.3) will correspond to a solution of the nonlinear optimization problem (1.1). (Note that it has already been established in Definition 2.3 that solutions of (1.3) for $\rho = 0$ correspond to solutions of the feasibility problem (1.2).) Thus, we are now ready to consider the relationship between these two problems, for which we require first-order optimality conditions for problem (1.1).

Definition 2.6 A point (x_*, σ_*) is first-order optimal for (1.1) if it is feasible, satisfies the bound $\sigma_* \geq 0$, and satisfies

$$\nabla f(x_*) + \nabla c(x_*)\sigma_* = 0 \tag{2.3a}$$
$$\sigma_* \circ c(x_*) = 0. \tag{2.3b}$$

The following result is well-known and the proof is straightforward.

Theorem 2.7 *If for $\rho > 0$ the point $(x_\rho, s_\rho, \sigma_\rho)$ is a first-order optimal point for* (1.3) *with $c(x_\rho) \leq 0$, then $(x_*, \sigma_*) = (x_\rho, \sigma_\rho/\rho)$ is a first-order optimal point for* (1.1).

This result confirms that solving the penalty problem (1.3) is an effective means for solving (1.1) or (1.2). That is, if $\rho > 0$ can be chosen small enough so that the solution of (1.3) provides a feasible solution to (1.1), then the slack variables will vanish and an optimal primal-dual pair for (1.1) is at hand. On the other hand, if no such ρ can be found, then the algorithm should reduce $\rho \to 0$ so that a certificate of infeasibility can be returned in the form of a primal-dual pair for (1.2) that corresponds to an infeasible point for (1.1).

Our last result shows that as long as the Lagrange multipliers in (2.3) are finite, there exists a sufficiently small penalty parameter such that a particular first-order optimal point of (1.1) corresponds to a first-order optimal point for (1.3). Thus, if (1.1) is feasible and a constraint qualification is assumed that guarantees finiteness of the Lagrange multipliers at all first-order optimal points, then there exists an overall lower bound on ρ needed for solving (1.1).

Theorem 2.8 *If (x_*, σ_*) is a first-order optimal point for* (1.1)*, then for all penalty parameter values $\rho \in (0, 1/\|\sigma_*\|_\infty]$, the point $(x_\rho, s_\rho, \sigma_\rho) = (x_*, 0, \rho\sigma_*)$ is a first-order optimal point for* (1.3)*. Thus, if the set of Lagrange multipliers corresponding to solutions of* (2.3) *is bounded, then there exists $\rho_* > 0$ such that any first-order optimal point for* (1.1) *corresponds to a first-order optimal point for* (1.3) *for $\rho = \rho_*$.*

One option for guaranteeing finiteness of the set of optimal Lagrange multipliers is to impose, for example, MFCQ on (1.1) itself. An assumption such as this is common, but it is important to note that it cannot be imposed if equality constraints are handled inappropriately. Thus, although we consider the inequality constrained form (1.1) for the majority of our discussion, it is crucial to outline reasonable means for solving generally constrained problems. We do this in §4.

This concludes our discussion of the theoretical foundations for our proposed algorithm. We have seen that the penalty-interior-point subproblem (1.5) is sufficiently regular in that MFCQ is satisfied at all feasible points. This same property holds for the penalty subproblem (1.3), to which (1.5) (in some sense) converges as $\mu \to 0$. Thus, we expect solving (1.5) to be a reasonable task, even if the constraints of problem (1.1) are difficult or infeasible. Moreover, we have seen that as long as the penalty and interior-point parameters are updated appropriately and the slack variables and Lagrange multipliers are restricted to remain in appropriate intervals, first-order optimal points for either (1.1) or (1.2) can be obtained by solving the first-order optimality conditions (2.2).

3 Algorithm Development

We develop the details of our algorithm in a couple stages so as to highlight the important aspects of each component. For given ρ and μ, the algorithm essentially amounts to a Newton iteration with a line search for solving (2.2), with additional functionalities for maintaining the slack variables and Lagrange multipliers in appropriate intervals, and to enforce descent on a merit function. We begin with the basics behind our step computation and line search, and then discuss strategies for updating the penalty and interior-point parameters during each iteration. Complete descriptions of a conservative and an aggressive algorithm are presented during the course of this section.

3.1 Search direction computation and line search

Our search direction computation is derived from a Newton iteration applied to the optimality conditions for problem (1.5) for given values of the penalty parameter ρ and the interior-point parameter μ. In particular, applying a Newton iteration to (2.2), we obtain the linear system

$$\begin{bmatrix} H_k & 0 & 0 & \nabla c(x_k) \\ 0 & \Omega_k & 0 & I \\ 0 & 0 & \Gamma_k & -I \\ \nabla c(x_k)^T & I & -I & 0 \end{bmatrix} \begin{bmatrix} \Delta x_k \\ \Delta r_k \\ \Delta s_k \\ \Delta \lambda_k \end{bmatrix} = - \begin{bmatrix} \rho \nabla f(x_k) + \nabla c(x_k)\lambda_k \\ \lambda_k - \mu R_k^{-1} e \\ e - \lambda_k - \mu S_k^{-1} e \\ \theta(z_k) \end{bmatrix}. \tag{3.1}$$

Defining the Lagrangian of (1.5) to be

$$L(z, \lambda; \rho, \mu) := \phi(z; \rho, \mu) + \lambda^T \theta(z),$$

an appropriate choice for H_k is any sufficiently positive definite approximation

$$H_k \approx \nabla^2_{xx} L(z_k, \lambda_k; \rho) = \rho \nabla^2_{xx} f(x_k) + \sum_{i=1}^{t} \lambda_k^i \nabla^2_{xx} c^i(x_k). \tag{3.2}$$

For the (diagonal) matrices Ω_k and Γ_k, there are two choices depending on whether the second and third blocks in (2.2) are scaled — with R and S, respectively — before or after derivatives are taken to derive the Newton system (3.1). If the equations are scaled beforehand, then this leads to a *primal* step computation with $\Omega_k := \mu R_k^{-2}$ and $\Gamma_k := \mu S_k^{-2}$. If the equations are scaled afterward, then we have a *primal-dual* step computation with $\Omega_k := R_k^{-1}\Lambda_k$ and $\Gamma_k := S_k^{-1}(I - \Lambda_k)$. It is generally accepted that computational experience supports the primal-dual option over the primal option, so we make this choice in our implementation.

As stated, the linear system (3.1) forms the basis of our search direction computation. We incorporate an additional procedure, however, to adjust the slack variables during the solution process. The procedure, which we call a *slack reset* (see also the "magical steps" of [17, §6.1]), effectively eliminates the slack variables from much of the iteration, and proceeds in the following manner. Suppose that x_k is given so that $c(x_k)$ is known. Then, consider setting r_k and s_k so as to solve

$$\begin{aligned} \min_{r,s} \ & -\mu \sum_{i=1}^{t} (\ln r^i + \ln s^i) + \sum_{i=1}^{t} s^i \\ \text{s.t. } & c(x_k) + r - s = 0 \quad (\text{with } (r, s) > 0) \end{aligned} \tag{3.3}$$

for a given μ. (Note that, with x fixed at x_k, (1.5) reduces to (3.3).) This problem is convex and separable, and it has the unique solution defined by

$$\begin{aligned} r_k^i &= r^i(x_k;\mu) := \mu - \tfrac{1}{2}c^i(x_k) + \tfrac{1}{2}\sqrt{c^i(x_k)^2 + 4\mu^2} \\ \text{and} \quad s_k^i &= s^i(x_k;\mu) := \mu + \tfrac{1}{2}c^i(x_k) + \tfrac{1}{2}\sqrt{c^i(x_k)^2 + 4\mu^2} \end{aligned} \tag{3.4}$$

for all $i = 1, \ldots, t$. Thus, at any iterate x_k and for any $\mu > 0$, the slack variables r_k and s_k can be set explicitly and trivially to solve (3.3).

Applied at the start (or, equivalently, at the end) of every iteration, the important effects of this slack reset are two-fold. First, the obvious effect is that the constraints of (1.5) are satisfied throughout the solution process. This means that, with respect to the optimality conditions (2.2), the rest of the algorithm need only focus on satisfying the components (2.2a) and (2.2b)–(2.2c) pertaining to dual feasibility and complementarity, respectively. The more significant effect, however, relates to the fact that we can avoid having to define a *third* parameter (along with ρ and μ) that might otherwise be needed in a merit function to balance the priorities of minimizing the objective and satisfying the constraints of (1.5). That is, with $r_k = r(x_k;\mu)$ and $s_k = s(x_k;\mu)$ defined by (3.4), the component Δx_k obtained by solving (3.1) is a descent direction at $x = x_k$ for the merit function

$$\widetilde{\phi}(x;\rho,\mu) := \rho f(x) - \mu \sum_{i=1}^{t} (\ln r^i(x;\mu) + \ln s^i(x;\mu)) + \sum_{i=1}^{t} s^i(x;\mu).$$

We formalize this claim with Lemma 3.1 below, the proof of which is straightforward. We assume that H_k in (3.1) yields

$$\begin{aligned} \Delta x_k^T H_k \Delta x_k + \Delta r_k^T \Omega_k \Delta r_k + \Delta s_k^T \Gamma_k \Delta s_k &> 0 \\ \text{for all} \quad \Delta z_k \neq 0 \quad \text{such that} \quad \nabla c(x_k)^T \Delta x_k + \Delta r_k - \Delta s_k &= 0, \end{aligned} \tag{3.5}$$

which, since $(r_k, s_k) > 0$ and $\lambda_k \in (0,1)$ imply $\Omega_k \succ 0$ and $\Gamma_k \succ 0$, means

$$\Delta x_k^T H_k \Delta x_k > 0 \quad \text{for all} \quad \Delta x_k \neq 0 \quad \text{such that} \quad \nabla c(x_k)^T \Delta x_k = 0.$$

This is a standard positive-definiteness assumption for line search methods with search direction computations similar to (3.1).

Lemma 3.1 *Suppose* (3.5) *holds and let* $r_k = r(x_k;\mu)$ *and* $s_k = s(x_k;\mu)$ *be set as in* (3.4). *Then, the search direction* Δx_k *yielded by* (3.1) *satisfies*

$$\begin{aligned} \nabla\widetilde{\phi}(x_k;\rho,\mu)^T \Delta x_k &= \nabla\phi(z_k;\rho,\mu)^T \Delta z_k \\ &= -\Delta x_k^T H_k \Delta x_k - \Delta r_k^T \Omega_k \Delta r_k - \Delta s_k^T \Gamma_k \Delta s_k < 0 \end{aligned} \tag{3.6}$$

and so is a descent direction for $\widetilde{\phi}(x;\rho,\mu)$ *at* $x = x_k$.

It is also prudent to clearly state the relationship between (unconstrained) minimizers of the merit function $\widetilde{\phi}(x;\rho,\mu)$ and first-order optimal points of (1.5) when the slack variables are set as in (3.4) and the Lagrange multipliers are computed as in the Newton system (3.1). We do so with the following theorem. The proof of the result is straightforward.

Theorem 3.2 *Suppose that for fixed ρ and μ the sequence of iterates $\{x_k\}$ yields*

$$\nabla\widetilde{\phi}(x_k;\rho,\mu) \to 0, \tag{3.7}$$

r_k and s_k are set by (3.4) for all k, and the Lagrange multipliers $\{\lambda_k\}$ yield

$$\left\| \begin{bmatrix} R_k\lambda_k - \mu e \\ S_k(e-\lambda_k) - \mu e \end{bmatrix} \right\| \to 0. \tag{3.8}$$

Then, any limit point (z_*, λ_*) of $\{(z_k,\lambda_k)\}$ is first-order optimal for (1.5).

The backtracking line search procedure in our algorithm proceeds in the following manner. First, as in the implementation of many other interior-point methods, we restrict the search by observing fraction-to-the-boundary rules for the slack variables; see Definition 2.4 and Theorem 2.5. An appropriate feature in our case, however, is that these rules take into account our slack reset procedure; i.e., we require that the primal steplength $\alpha_k \in (0,1]$ satisfies

$$r(x_k + \alpha_k\Delta x_k;\mu) \geq \tau r_k \quad \text{and} \quad s(x_k + \alpha_k\Delta x_k;\mu) \geq \tau s_k \tag{3.9}$$

for a given $\tau \in (0,1)$. We also require that the steplength satisfies

$$\widetilde{\phi}(x_k + \alpha_k\Delta x_k;\rho,\mu) \leq \widetilde{\phi}(x_k;\rho,\mu) + \alpha_k\eta\nabla\widetilde{\phi}(x_k;\rho,\mu)^T\Delta x_k \tag{3.10}$$

for some $\eta \in (0,1)$. Once $\alpha_k \in (0,1]$ is set satisfying (3.9) and (3.10), we update

$$x_{k+1} \leftarrow x_k + \alpha_k\Delta x_k, \quad r_{k+1} \leftarrow r(x_{k+1};\mu), \quad \text{and} \quad s_{k+1} \leftarrow s(x_{k+1};\mu). \tag{3.11}$$

In the dual space, we impose a fraction-to-the-boundary rule to compute

$$\beta_k = \max\{\beta \in (0,1] : \lambda_k + \beta\Delta\lambda_k \in [\tau\lambda_k, e - \tau(e-\lambda_k)]\} \tag{3.12}$$

and then update

$$\lambda_{k+1} \leftarrow \lambda_k + \beta_k\Delta\lambda_k. \tag{3.13}$$

The rule (3.12) is imposed to ensure $\lambda_k \in (0,1)$ for all k.

Note that if the interior-point parameter μ is changed after the update (3.11), but before the start of the next iteration, then the slack variables should again be reset by (3.4) since this computation is trivial.

3.2 Updating the penalty and interior-point parameters

We are now ready to present our techniques for updating the penalty and interior-point parameters during the optimization process. In the pursuit of appropriate updates for these quantities, there are at least two significant challenges. The first is the design of updating techniques that result in rapid convergence to either a first-order optimal point of the nonlinear optimization problem (1.1) or the feasibility problem (1.2). The second is that the user (and so the algorithm) does not know beforehand which of these problems should be solved. The algorithm must determine dynamically the emphasis that should be placed on minimizing violations in feasibility and minimizing the objective of (1.1), bearing in mind that there should be a preference for solving (1.1) instead of only solving (1.2).

We present two updating schemes — one conservative and one aggressive — the latter of which we believe successfully addresses these challenges.

Our conservative strategy is detailed as Algorithm 1 below. At the heart of the algorithm is the conservative strategy for interior-point methods known as the Fiacco-McCormick approach [13]. In our context, this approach corresponds to setting fixed values for ρ and μ until (2.2) is satisfied to a sufficient accuracy, at which point μ is decreased and the procedure is repeated. Despite the fact that it suffers from important limitations, this monotone approach has been employed in various interior-point algorithms due to the foundation it provides for ensuring global convergence; e.g., see [6,32]. As for updating the penalty parameter ρ, Algorithm 1 uses a simple approach that only decreases ρ when a (sufficiently accurate) solution of (1.5) does not correspond to a (sufficiently accurate) solution to (1.1). We also include a safeguard to ensure that the iterates do not diverge from the feasible region. (Specific conditions corresponding to such statements as "satisfied to a sufficient accuracy" will be presented in detail in §4.)

Algorithm 1 Penalty-interior-point algorithm with conservative updates

1: (Initialization): Choose a fraction-to-the-boundary parameter $\tau \in (0,1)$, sufficient decrease parameter $\eta \in (0,1)$, backtracking parameter $\gamma \in (0,1)$, reduction parameters $\kappa_\rho \in (0,1)$ and $\kappa_\mu \in (0,1)$, and infeasibility warning level $\omega > 0$. Choose an initial penalty parameter $\rho > 0$ and interior-point parameter $\mu > 0$. Choose an initial primal-dual iterate (x_0, λ_0) and set the slack variables (r_0, s_0) by (3.4). Set $k \leftarrow 0$.
2: (Optimality check): If (2.2) with $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is sufficiently small, then terminate; x_k is a first-order optimal point for (1.1).
3: (Infeasibility check): If (2.2) with $(\rho, \mu) = (0,0)$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, then terminate; x_k is an infeasible stationary point for (1.1).
4: (Hessian computation): Set H_k so (3.5) holds, $\Omega_k := R_k^{-1}\Lambda_k$, and $\Gamma_k := S_k^{-1}(I - \Lambda_k)$.
5: (Search direction computation): Compute $(\Delta z_k, \Delta\lambda_k)$ by (3.1).
6: (Line search): Set α_k as the largest value in $\{\gamma^0, \gamma^1, \gamma^2, \dots\}$ such that (3.9) and (3.10) are satisfied and set β_k by (3.12).
7: (Iterate update): Set z_{k+1} and λ_{k+1} by (3.11) and (3.13), respectively.
8: (Interior-point parameter update): If (2.2) is satisfied to a sufficient accuracy, then set $\mu \leftarrow \kappa_\mu \mu$ and reset r_{k+1} and s_{k+1} by (3.4).
9: (Penalty parameter update): If (2.2) with $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, or if $v(x_{k+1}) > \max\{v(x_0), v(x_k), \omega\}$, then set $\rho \leftarrow \kappa_\rho \rho$.
10: (k increment): Set $k \leftarrow k + 1$ and go to step 2.

The conservative updating strategy in Algorithm 1 is representative of the updating schemes that have been implemented in other penalty-interior-point-like algorithms. Indeed, the algorithm in [1] essentially follows this same approach, except that it defines a unique penalty parameter for each slack variable. The algorithms in [10,17] update the penalty parameter by monitoring a feasibility violation measure over the course of all iterations — another type of conservative technique — though their method for updating the interior-point parameter is the same as the Fiacco-McCormick approach described above. Finally, the methods in [4,29] either update their parameters by a fixed factor during every iteration or by monitoring problem function values over the course of the optimization process.

These conservative approaches will perform well in situations when problem (1.1) is feasible and the initial penalty parameter value is sufficiently small so that obtaining a first-order optimal point for (1.5) immediately yields a first-order

optimal point for (1.1). However, in other cases — such as when the initial penalty parameter is too large, the nonlinear optimization problem (1.1) is infeasible, or if a conservative update for μ leads to slow convergence — they may lead to slow convergence. For example, in the context of Algorithm 1, each distinct value for the penalty parameter ρ potentially requires the computational efforts of a complete run of a standard interior-point algorithm. This can be exceedingly expensive if ρ must eventually be reduced to (near) zero.

Our main approach is a more aggressive strategy as it considers changes in ρ and μ *during every iteration*. In such a context, it is important to note that, in practice, it is reasonable to set the primal-dual matrix on the left-hand side of (3.1) as a fixed quantity throughout a given iteration. The values for ρ and μ normally influence the values for H_k, Ω_k, and Γ_k, but since we will be interested in comparing the solutions of (3.1) for various pairs (ρ, μ), it is beneficial to fix these matrices so that only one matrix factorization is required per iteration (see §4 for further details on how this can be done). In this manner, changes in ρ and μ during the search direction computation will only effect the right-hand side of (3.1). There are two key drawbacks to performing the operations in this way, both of which are accounted for in our algorithm. The first drawback is that not having H_k depend on ρ may hinder the algorithm from converging quickly on infeasible problems. We rectify this through a condition that may decrease ρ quite rapidly, when appropriate. The second drawback is that not having Ω_k and Γ_k depend on μ means that we are in danger of losing the descent property for the merit function $\widetilde{\phi}(x; \rho, \mu)$ proved in Lemma 3.1. We rectify this by enforcing an explicit condition on the directional derivative of this function.

During iteration k, we define a finite set of potential penalty parameter values $\mathcal{R}_k$ and a finite set of potential interior-point parameters $\mathcal{M}_k$. In the set $\mathcal{R}_k$, we include the current ρ and lesser values; i.e., we do not allow the penalty parameter to increase. In the set $\mathcal{M}_k$, we also only include the current μ and lesser values, but remark that variants of our approach may include greater values in order to allow the interior-point parameter to increase (assuming appropriate safeguards are in place); see [26]. See §4 for our choices of $\mathcal{R}_k$ and $\mathcal{M}_k$ in our implementation. In our strategy below, we begin by fixing ρ as the largest value for which we have an *admissible pair* (ρ, μ) with $\rho \in \mathcal{R}_k$ and $\mu \in \mathcal{M}_k$. The definition of an admissible pair changes depending on whether or not the current iterate is sufficiently feasible, but in either case the definition is based on a series of conditions defined in terms of the parameters themselves and the resulting search direction. Once ρ has been fixed in this manner, we then set μ so that the resulting pair (ρ, μ) is still admissible and the resulting search direction satisfies an additional condition.

The cornerstones of our approach can be summarized as two main ideas: (1) For updating the penalty parameter, *define a quantity with known desirable properties related to the pursuit of primal feasibility* on which a series of updating conditions can be based, and (2) for updating the interior-point parameter, *define a measure related to the pursuit of dual feasibility and complementarity* through which the algorithm can promote long steps and fast convergence. These types of ideas have appeared separately before in the literature, namely in the steering rules for updating a penalty parameter described in [9] and the adaptive barrier rules for updating an interior-point parameter described in [26]. However, to the best of our knowledge, this is the first work in which these distinct strategies have been combined.

For updating ρ, we define the type of quantity mentioned in the previous paragraph by considering the following linear model of the penalty-interior-point function $\phi(z;\rho,\mu)$ about a given point z:

$$l(\Delta z;\rho,\mu,z) := \phi(z;\rho,\mu) + \nabla\phi(z;\rho,\mu)^T\Delta z.$$

At an iterate z_k, let $\Delta z_k^{\rho,\mu}$ denote the search direction computed by (3.1) for given values of ρ and μ on the right-hand side. (Similar notation is also used below for the search direction component $\Delta x_k^{\rho,\mu}$.) The reduction in $l(\Delta z;\rho,\mu,z)$ at z_k for such a computed search direction is then

$$\begin{aligned}\Delta l(\Delta z_k^{\rho,\mu};\rho,\mu,z_k) &:= l(0;\rho,\mu,z_k) - l(\Delta z_k^{\rho,\mu};\rho,\mu,z_k)\\ &= -\nabla\phi(z_k;\rho,\mu)^T\Delta z_k^{\rho,\mu}.\end{aligned}$$

Since (3.1), $\theta(z_k) = 0$, and (3.5) yield (3.6), this reduction is positive for any nonzero step. In particular, a nonzero solution to (3.1) for $\rho = 0$ yields

$$\Delta l(\Delta z_k^{0,\mu};0,\mu,z_k) > 0. \tag{3.14}$$

The fact that we know this quantity is strictly positive is of central importance since we can use it as an appropriate measure of the potential progress toward primal feasibility that is possible from the current iterate.

If a given iterate is infeasible, then we define an admissible pair (ρ,μ) as any satisfying the conditions outlined in the following bullets.

– We aim to ensure that the progress toward attaining (linearized) primal feasibility is proportional to the level that would be obtained with $\rho = 0$. To this end, along with the linear model of $\phi(z;\rho,\mu)$ defined above, we also define the linear model of the merit function $\widetilde{\phi}(x;\rho,\mu)$ given by

$$\widetilde{l}(\Delta x;\rho,\mu,x) := \widetilde{\phi}(x;\rho,\mu) + \nabla\widetilde{\phi}(x;\rho,\mu)^T\Delta x.$$

A given search direction $\Delta x_k^{\rho,\mu}$ computed by (3.1) yields the reduction

$$\begin{aligned}\Delta\widetilde{l}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k) &:= \widetilde{l}(0;\rho,\mu,x_k) - \widetilde{l}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k)\\ &= -\nabla\widetilde{\phi}(x_k;\rho,\mu)^T\Delta x_k^{\rho,\mu}.\end{aligned}$$

For the value of μ used to compute r_k and s_k, Lemma 3.1 reveals that the model reductions $\Delta l(\Delta z_k^{\rho,\mu};\rho,\mu,z_k)$ and $\Delta\widetilde{l}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k)$ coincide. However, for varying μ, these reductions may differ. We can always be sure that (3.14) holds when $\rho = 0$, but it is in fact a reduction in $\widetilde{l}(\Delta x;0,\mu,x_k)$ that is our best indication of progress toward primal feasibility. Thus, we require an admissible pair (ρ,μ) to be one such that the resulting search direction satisfies

$$\Delta\widetilde{l}(\Delta x_k^{\rho,\mu};0,\mu,x_k) \geq \epsilon_1\Delta l(\Delta z_k^{0,\mu};0,\mu,z_k) > 0 \tag{3.15}$$

for some constant $\epsilon_1 \in (0,1)$. (Note that for μ set as the value used to compute r_k and s_k and for ρ sufficiently small, (3.15) will be satisfied, so (3.15) is satisfiable as long as this $\mu \in \mathcal{M}_k$ and $\mathcal{R}_k$ includes sufficiently small values.)

– Applying block elimination to (3.1), we find that an appropriate quadratic model of the merit function $\widetilde{\phi}(x;\rho,\mu)$ is given by

$$\widetilde{q}(\Delta x;\rho,\mu,x,\lambda) := \widetilde{l}(\Delta x;\rho,\mu,x) + \tfrac{1}{2}\Delta x^T(H + \nabla c(x)(\Omega^{-1}+\Gamma^{-1})^{-1}\nabla c(x)^T)\Delta x,$$

and the reduction in this quantity yielded by Δx_k at (x_k,λ_k) is given by

$$\begin{aligned}\Delta\widetilde{q}(\Delta x_k;\rho,\mu,x_k,\lambda_k) &:= \widetilde{q}(0;\rho,\mu,x_k,\lambda_k) - \widetilde{q}(\Delta x_k;\rho,\mu,x_k,\lambda_k)\\ &= \Delta\widetilde{l}(\Delta x_k;\rho,\mu,x_k)\\ &\quad - \tfrac{1}{2}\Delta x_k^T(H_k + \nabla c(x_k)(\Omega_k^{-1}+\Gamma_k^{-1})^{-1}\nabla c(x_k)^T)\Delta x_k.\end{aligned}$$

For the value of μ used to compute r_k and s_k, this reduction will be positive, but for varying μ, this is not a guarantee. Moreover, as in [9], it is important that this reduction is sufficiently positive so that the resulting search direction is one of significant descent for the merit function $\widetilde{\phi}(z;\rho,\mu)$. We enforce this condition by stating that an admissible pair must yield

$$\Delta\widetilde{q}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k,\lambda_k) \geq \epsilon_2\Delta l(\Delta z_k^{0,\mu};0,\mu,z_k) > 0 \tag{3.16}$$

where $\epsilon_2 \in (0,1)$ is a given constant. (Note that for μ set as the value used to compute r_k and s_k and for ρ sufficiently small, (3.16) will be satisfied.)

– If the current iterate satisfies (2.2) sufficiently accurately for $(\rho,\mu) = (0,0)$, then the algorithm is likely converging to an infeasible stationary point. In such cases, we note that in [5], in the context of a penalty-sequential-quadratic-programming method, it is possible to attain superlinear convergence to an infeasible stationary point (see Definition 2.3) if the penalty parameter is driven to zero according to $o(\|(x_k,s_k,\sigma_k) - (x_\times,s_\times,\sigma_\times)\|)$. Thus, with the hope of mimicking this behavior, we require that an admissible pair (ρ,μ) has

$$\rho \leq \left\| \begin{bmatrix} \nabla c(x_k)\lambda_k \\ R_k\lambda_k \\ S_k(e-\lambda_k) \end{bmatrix} \right\|^2. \tag{3.17}$$

Note that if the current iterate is not in the neighborhood of an infeasible stationary point, then the right-hand side of (3.17) will be relatively large and the satisfaction of this condition will not impose a decrease in ρ.

In summary, if the current iterate is not sufficiently feasible, then (3.15) and (3.16) ensure progress in linearized feasibility and that the computed search direction is one of significant descent for our merit function, and (3.17) promotes fast convergence to infeasible stationary points. If, on the other hand, the current iterate is already sufficiently feasible, then we do not enforce any of (3.15), (3.16), or (3.17), but require

$$\Delta\widetilde{q}(\Delta x_k^{\rho,\mu};\rho,\mu,x_k,\lambda_k) > 0 \tag{3.18}$$

so that the computed direction is again one of descent for $\widetilde{\phi}(x;\rho,\mu)$ at x_k.

If, for ρ fixed as described above and all $\mu \in \mathcal{M}_k$, $\mathcal{A}_k$ denotes the set of all admissible pairs (ρ,μ), then we choose μ based on the strategy in the next bullet.

– In addition to its effect on primal feasibility, a given search direction $(\Delta z, \Delta\lambda)$ will move toward or, potentially, away from satisfying the dual feasibility and complementarity conditions in (2.2). In particular, the chosen value for μ will effect the progress obtained toward both of these goals. A measure of the effect that a given search direction has on these quantities is the quality function

$$m(\Delta z, \Delta\lambda; \rho, \mu, z) := \left\| \begin{bmatrix} \rho\nabla f(x) + \nabla c(x)(\lambda + \Delta\lambda) \\ (R + \Delta R)(\lambda + \Delta\lambda) \\ (S + \Delta S)(e - \lambda - \Delta\lambda) \end{bmatrix} \right\|_\infty . \tag{3.19}$$

We choose μ as the largest value in $\mathcal{M}_k$ such that the computed search direction approximately minimizes this function. Specifically, if $\bar{\mu}$ is the value such that the computed direction minimizes (3.19) over all $\mu \in \mathcal{M}_k$ with $(\rho, \mu) \in \mathcal{A}_k$, then we choose μ as the largest value such that $(\rho, \mu) \in \mathcal{A}_k$ and

$$m(\Delta z_k^{\rho,\mu}, \Delta\lambda_k^{\rho,\mu}; \rho, \mu, z_k) \leq \epsilon_3 m(\Delta z_k^{\rho,\bar{\mu}}, \Delta\lambda_k^{\rho,\bar{\mu}}; \rho, \bar{\mu}, z_k), \tag{3.20}$$

where $\epsilon_3 > 1$ is a given constant. Here, it is important to note that ρ is fixed during the selection of μ, since otherwise the approximate minimization of (3.19) may drive ρ to zero unnecessarily. We also remark that the strategy in [26] is to choose μ so as to minimize a quality function, whereas our approach is to find a *large* value that only approximately minimizes it. We found this latter strategy to be more effective in our numerical experiments.

Our approach can be implemented efficiently as Algorithm 2 below. Naturally, this algorithm can be fine-tuned and enhanced so that, e.g., search directions for certain pairs of ρ and μ are not calculated if they will not be used, but for simplicity in our presentation, we present a straightforward approach that is easier to follow. We also note that, in practice, one should perhaps follow the algorithm in [26] and cut trial search directions by fraction-to-the-boundary rules and trust region radii related to recent search direction norms before evaluating the quantities in (3.15), (3.16), (3.18), and (3.19). For clarity in our presentation, we suppress such details here, but note that they are included in our implementation discussed in §4.

4 Numerical Results

Algorithms 1 and 2 have been implemented in MATLAB. Henceforth, we refer to our implementation of these algorithms as `PIPAL-c` and `PIPAL-a`, respectively, which stand for Penalty-Interior-Point ALgorithm, conservative and aggressive.[1] In this section, we discuss details of our implementation and present results for AMPL [14] versions of problems in the CUTEr [16,18] collection. We compare the results obtained by `PIPAL-c` and `PIPAL-a` with those obtained by the interior-point algorithm implemented in the well-established `IPOPT` software [33].[2]

Our experiments support the claim that when an interior-point method is performing at its best, it is difficult to beat such an approach in terms of efficiency. For one thing, our penalty-interior-point subproblem (1.5) clearly has many more

[1] In Hindi, pipal is the name for the Sacred Fig, or Bo-Tree, native to southern Asia, and sacred to Buddhists. It is a symbol of happiness, prosperity, longevity, and good luck.

[2] http://www.coin-or.org/Ipopt/

Algorithm 2 Penalty-interior-point algorithm with aggressive updates

1: (Initialization): Choose a fraction-to-the-boundary parameter $\tau \in (0,1)$, sufficient decrease parameter $\eta \in (0,1)$, backtracking parameter $\gamma \in (0,1)$, reduction parameters $\kappa_\rho \in (0,1)$ and $\kappa_\mu \in (0,1)$, infeasibility warning level $\omega > 0$, and updating parameters $\epsilon_1, \epsilon_2 \in (0,1)$ and $\epsilon_3 > 1$. Choose a penalty parameter $\rho > 0$ and interior-point parameter $\mu > 0$. Choose a primal-dual iterate (x_0, λ_0) and set the slack variables (r_0, s_0) by (3.4). Set $k \leftarrow 0$.
2: (Optimality check): If (2.2) with $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is sufficiently small, then terminate; x_k is first-order optimal for (1.1).
3: (Infeasibility check): If (2.2) with $(\rho, \mu) = (0,0)$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, then terminate; x_k is an infeasible stationary point for (1.1).
4: (Hessian computation): Set H_k so (3.5) holds, $\Omega_k := R_k^{-1}\Lambda_k$, and $\Gamma_k := S_k^{-1}(I - \Lambda_k)$.
5: (Parameter update initialization): Choose $\mathcal{R}_k$ as an infinite set of points in $(0, \rho]$ (that includes ρ) and $\mathcal{M}_k$ as a finite set of points in $(0, \mu]$ (that includes μ).
6: (Feasibility direction computations): For all $\mu \in \mathcal{M}_k$, compute $\Delta z_k^{0,\mu}$ by (3.1) with $\rho = 0$.
7: (Penalty parameter update 1): If $v(x_k)$ is not sufficiently small, then set ρ as the largest value in $\mathcal{R}_k$ such that for at least one $\mu \in \mathcal{M}_k$, the pair (ρ, μ) is *admissible* in that it satisfies (3.17) and the solution to (3.1) corresponding to (ρ, μ) satisfies (3.15) and (3.16). Otherwise, if $v(x_k)$ is sufficiently small, then set ρ as the largest value in $\mathcal{R}_k$ such that for at least one $\mu \in \mathcal{M}_k$, the pair (ρ, μ) is *admissible* in that the solution to (3.1) corresponding to (ρ, μ) satisfies (3.18). In either case, let $\mathcal{A}_k$ be the set of all admissible pairs corresponding to the chosen ρ with $\mu \in \mathcal{M}_k$.
8: (Interior-point parameter update 1): Set $\bar{\mu}$ as the value in $\mathcal{M}_k$ such that with $\mu = \bar{\mu}$, we have $(\rho, \mu) \in \mathcal{A}_k$ and (3.19) is minimized. Then, set μ as the largest value in $\mathcal{M}_k$ such that $(\rho, \mu) \in \mathcal{A}_k$ and the search direction corresponding to (ρ, μ) satisfies (3.20).
9: (Search direction selection): Set $(\Delta z_k, \Delta\lambda_k)$ as the solution to (3.1) corresponding to (ρ, μ).
10: (Line search): Set α_k as the largest value in $\{\gamma^0, \gamma^1, \gamma^2, \dots\}$ such that (3.9) and (3.10) are satisfied and set β_k by (3.12).
11: (Iterate update): Set z_{k+1} and λ_{k+1} by (3.11) and (3.13), respectively.
12: (Interior-point parameter update 2): If (2.2) is satisfied to a sufficient accuracy, then set $\mu \leftarrow \kappa_\mu \mu$ and reset r_{k+1} and s_{k+1} by (3.4).
13: (Penalty parameter update 2): If (2.2) with $\mu = 0$ is satisfied to a sufficient accuracy and $v(x_k)$ is not sufficiently small, or if $v(x_{k+1}) > \max\{v(x_0), v(x_k), \omega\}$, then set $\rho \leftarrow \kappa_\rho \rho$.
14: (k increment): Set $k \leftarrow k + 1$ and go to step 2.

degrees of freedom than (1.4) (i.e., the penalty parameter and the additional slack variables), the presence of which may result in slower convergence in many cases. However, the results below highlight two key strengths of our approach: (1) Our algorithm is practically as efficient on easy problems, and (2) our algorithm exhibits superior behavior on degenerate and infeasible problems. This suggests that, with a more sophisticated implementation, our algorithm has the potential to be a successful general-purpose solver, and represents an advancement in the development of nonlinear optimization methods that are equally robust and efficient on problems with difficult constraints. This latter distinction is especially important in, for two examples, the context of mathematical programs with complementarity constraints [31], a class of problems where great advancements have been made possible by penalty techniques (e.g., see [22,24]), and the context of mixed-integer nonlinear optimization [20], where a potentially large number of infeasible subproblems can arise in branch-and-bound frameworks (e.g., see [2,30]).

All experiments were run on an 8-core AMD Opteron 6128 machine with 2.0GHz clock speed running Debian GNU/Linux and MATLAB 7.11 (R2010b).

4.1 Implementation details

This paper has described an algorithm for the inequality constrained problem (1.1), but our methods can be applied to problems with equality constraints as well. Two options for incorporating an equality constraint $c^i(x) = 0$ into our formulation are to split it into the pair of inequalities $c^i(x) \leq 0$ and $c^i(x) \geq 0$, or to introduce a slack variable $a^i \geq 0$ and impose the inequalities $-a^i \leq c^i(x) \leq a^i$, and then apply the techniques above for inequality constrained optimization. However, there are significant drawbacks to both of these approaches. In the former case, MFCQ fails to hold, meaning that the theoretical foundations in §2 do not apply, and in the latter case, the penalty-interior-point subproblem necessarily involves the definition of even more slack variables, resulting in many more degrees of freedom. An alternative type of reformulation that involves fewer slack variables is to replace $c^i(x) = 0$ with

$$c^i(x) = a^i - b^i, \quad (a^i, b^i) \geq 0,$$

and then penalize constraint violations in the objective through the term $a^i + b^i$. Including these constraints and objective term in the penalty subproblem (1.3) is reasonable as the resulting subproblem again satisfies MFCQ, but in this case it is not necessary to add more slack variables when deriving the corresponding penalty-interior-point subproblem (since trivial equality constraints can be eliminated). This is the strategy we employ for handling equality constraints in our code.

`PIPAL-c` and `PIPAL-a` both require checks of the feasibility violation measure $v(x_k)$ and of the first-order optimality conditions (2.2). We define these checks in our implementation in the following manner. First, we observe the condition

$$v(x_k) \leq \epsilon v(x_0) \tag{4.1}$$

where $\epsilon \in (0, 1)$ is a small constant. If this condition is satisfied, then $v(x_k)$ is deemed sufficiently small, since then the level of infeasibility has decreased significantly with respect to the starting point. As for the first-order optimality conditions, we also define a relative tolerance, in this case with respect to the gradient of the penalty subproblem objective at the current iterate:

$$\left\| \begin{bmatrix} \rho \nabla f(x_k) + \nabla c(x_k) \lambda_k \\ R_k \lambda_k - \mu e \\ S_k (e - \lambda_k) - \mu e \end{bmatrix} \right\|_\infty \leq \max\{\epsilon, \mu\} \max\{1, \|\rho \nabla f(x_k)\|_\infty\}. \tag{4.2}$$

Equations (2.2) are said to be satisfied sufficiently accurately if (4.2) holds. Note that this condition takes into account the fact that $\theta(z_k) = 0$ for all k.

In summary, conditions (4.1) and (4.2) represent those that are checked as stopping conditions for the algorithm. If both conditions hold for the current ρ and $\mu = 0$, then the current point is deemed first-order optimal for (1.1). If (4.1) does not hold, but (4.2) holds for $(\rho, \mu) = (0, 0)$, then the current point is deemed an infeasible stationary point for (1.1). If no point satisfying either of these two sets of conditions has been found, then we terminate after a limit of 1000 iterations.

The input parameters for our implementation are set in the following manner. First, we choose the initial primal iterate to be that set by the AMPL model, assuming it has been provided, and otherwise choose it to be the origin. The initial Lagrange multipliers for equality constraints are set to zero (as they are required to lie in the interval $(-1, 1)$ throughout the solution process) and the

initial Lagrange multipliers for the inequality constraints are set to $\frac{1}{2}$. As for the remaining input parameters chosen in our implementation, including the value of ϵ defined in (4.1) and (4.2), they are given in Table 4.1. In `PIPAL-a`, we choose the sets of penalty and interior-point parameters to be

$$\begin{aligned} \mathcal{R}_k = &\ \{\kappa_\rho^4\rho, \kappa_\rho^3\rho, \kappa_\rho^2\rho, \kappa_\rho\rho, \rho\} \\ \text{and } \mathcal{M}_k = &\ \{\kappa_\mu^{10}\mu, \kappa_\mu^9\mu, \ldots, \kappa_\mu\mu, \mu\}, \end{aligned}$$

where (ρ, μ) is the current pair carried over from the previous iteration. Note that not having $\mathcal{R}_k$ include arbitrarily small values may mean that an accessible pair, as defined in Algorithm 2, may not be found during a given iteration. However, we found this to be an extremely rare event in our experiments, and found it sufficient to simply maintain the current (ρ, μ) in such cases.

Param.	Val.	Param.	Val.
ϵ	1e-6	ω	1e-1
τ	1e-2	ϵ_1	1e-2
η	1e-8	ϵ_2	1e-2
γ	5e-1	ϵ_3	1+1e-2
κ_ρ	5e-1	ρ	1e-1
κ_μ	1e-1	μ	1e-1

Table 4.1 Input parameter values for `PIPAL-c` and `PIPAL-a`.

The feasibility and search direction computations in steps 6 through 9 of `PIPAL-a` (i.e., Algorithm 2) are facilitated in our implementation by noting that with only one matrix factorization and at most three sets of forward and backward solves, the solution of (3.1) for any given pair (ρ, μ) can be obtained. Indeed, if we define $\Delta^{\rho,\mu}$ as the solution to (3.1) for certain values of ρ and μ, then

$$\Delta^{\rho,\mu} = \left(\frac{\rho}{\bar{\rho}} + \frac{\mu}{\bar{\mu}} - 1\right)\Delta^{\bar{\rho},\bar{\mu}} + \left(1 - \frac{\mu}{\bar{\mu}}\right)\Delta^{\bar{\rho},0} + \left(1 - \frac{\rho}{\bar{\rho}}\right)\Delta^{0,\bar{\mu}}. \tag{4.3}$$

Thus, with $\bar{\rho}$ and $\bar{\mu}$ set as the values for the penalty and interior-point parameters, respectively, at the start of the iteration, the construction of $\Delta^{\rho,\mu}$ for any pair (ρ, μ) is performed easily by taking linear combinations of $\Delta^{\bar{\rho},\bar{\mu}}$, $\Delta^{\bar{\rho},0}$, and $\Delta^{0,\bar{\mu}}$.

Along with the comments in the preceding paragraph, we note if the initial Newton system computed during iteration k — with H_k set to the Hessian of the Lagrangian as in (3.2) — satisfies (3.5), then indeed only one matrix factorization is required per iteration. However, if (3.5) is not satisfied, then we shift the eigenvalues of H_k by adding a multiple of the identity matrix to it so that the resulting system has the correct inertia. The shift is performed with an iterative scheme that first tries a small multiple of the identity, and then increases it exponentially, if necessary, until the Newton matrix in (3.1) has n positive eigenvalues.

4.2 Experiments with the CUTEr collection

We ran three sets of numerical experiments. In the first set, we applied `PIPAL-c`, `PIPAL-a`, and `IPOPT` to problems in the CUTEr collection. The initial set of prob-

lems that we considered is freely available.[3] However, due to memory limitations in our implementation, we removed large problems for which the Newton system (3.1) involved more than ten thousand variables. We also removed problems for which none of the algorithms were able to find a solution. The remaining set included 422 problems. The problems were run with AMPL's presolver turned off.

The results for this set of experiments are summarized in Figure 4.1. In the figure, we provide four performance profiles. The profiles on the top and the bottom-left compare iteration counts for each pair of algorithms; see [25]. For example, in the profile on the top-left, `PIPAL-c` is "positive" and `PIPAL-a` is "negative". If, for a particular problem, the "positive" algorithm requires fewer iterations to locate a first-order optimal or infeasible stationary point, then this is represented by a positive-valued bar corresponding to the ratio of iteration counts for the two algorithms. Similarly, if the "negative" algorithm requires fewer iterations, then this is represented by a negative-valued bar. If one algorithm solved a problem and the other did not, then this is represented by a bar of height 1 or -1. In this representation, one algorithm dominates another by having more (large) bars in its direction. Corresponding to Figure 4.1, Table 4.2 lists the characterizations of "positive" and "negative" algorithms in each of the profiles.

Plot location	''Positive''	''Negative''
Top-left	PIPAL-c	PIPAL-a
Top-right	PIPAL-c	IPOPT
Bottom-left	PIPAL-a	IPOPT

Table 4.2 "Positive" and "negative" orientations of algorithms in the performance profiles on the top and bottom-left of Figures 4.1, 4.2, 4.3, and 4.4.

The plot provided on the bottom-right of Figure 4.1 considers all three algorithms together and has the form of a logarithmic performance profile as proposed in [12]. Here, the leftmost values indicate the proportion of times each algorithm solves a given problem using the least number of iterations. The sum of these values exceeds one as ties are present. The right-most values represent the robustness of each approach; i.e., it provides the percentage of times that the problem is solved. In the overall plot, one algorithm dominates another by having its corresponding line above and to the left of those corresponding to the other algorithms.

The pair-wise comparisons and the logarithmic profile in Figure 4.1 suggest that all three algorithms are relatively robust, though `PIPAL-a` and especially `IPOPT` have an edge in terms of efficiency. `PIPAL-c` and `PIPAL-a` perform nearly identically in approximately half of the problems in the set, which suggests that for many of the problems a decrease in the penalty parameter was not necessary and was not required by the conditions in our dynamic updating strategy.

A clearer comparison between our two approaches can be made if we restrict our attention to problems for which `PIPAL-c` required a decrease in the penalty parameter in order to find a sufficiently accurate optimal solution. In other words, if we restrict our attention to the problems in our set for which the initial penalty parameter was too large, we can pinpoint situations in which the conservative updating strategy in `PIPAL-a` might have been slow to decrease this parameter.

[3] http://orfe.princeton.edu/~rvdb/ampl/nlmodels/cute/

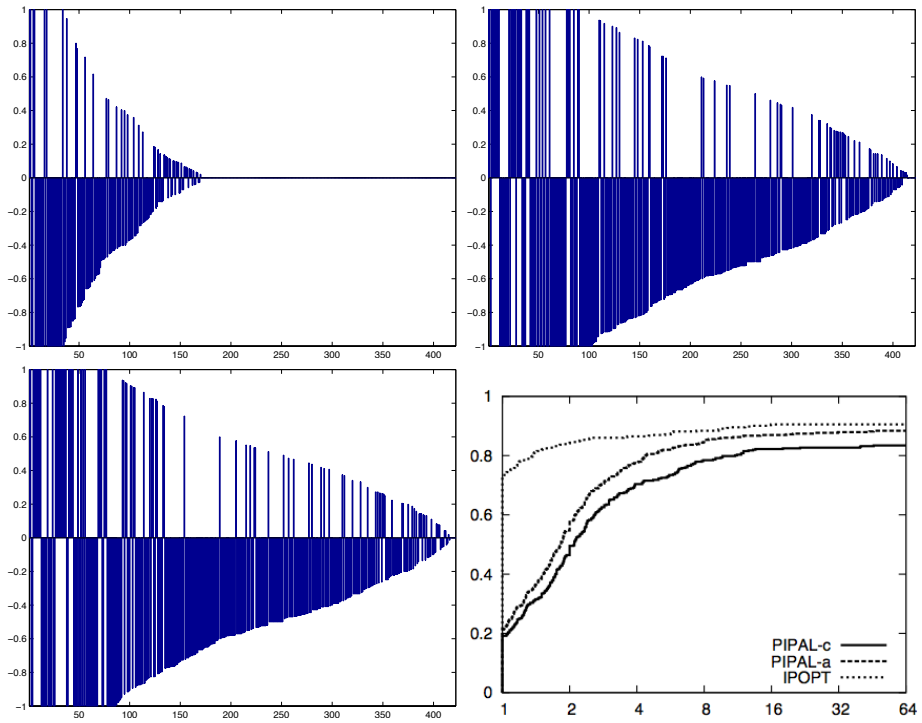

Fig. 4.1 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on problems from the CUTEr collection.

In this more focused setting, the pair-wise comparisons and the logarithmic profile in Figure 4.2 (involving the 136 problems for which a decrease in the penalty parameter was required) now all suggest that `PIPAL-c` is the weakest algorithm. It is not as robust as the other methods and is clearly less efficient. This provides evidence to support our development of a more aggressive updating strategy for the penalty and interior-point parameters. Indeed, `PIPAL-a` is a much more efficient algorithm, and is more competitive with the interior-point approach of `IPOPT`. These results suggest that with a more sophisticated implementation of `PIPAL-a`, it has the potential to be an effective general-purpose solver.

4.3 Experiments with degenerate versions of a set of CUTEr problems

We ran our second set of numerical experiments with degenerate variants of a subset of problems from §4.2, namely those in the Hock-Schittkowski set [21]. In particular, we created degenerate instances by manipulating the AMPL models so that for each constraint, $c^i(x) = 0$ or $c^i(x) \leq 0$, we added

$$-c^i(x)^2 \leq 0.$$

These changes have no effect on the feasible region, but clearly make each problem degenerate as constraint gradients for the added active constraints all converge to

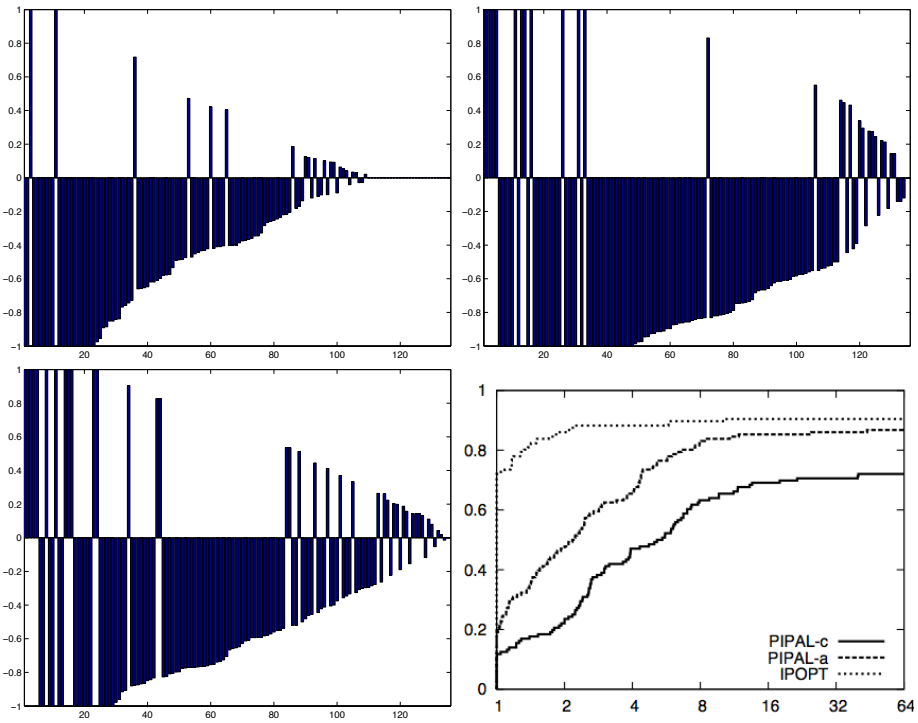

Fig. 4.2 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on problems from CUTEr for which `PIPAL-c` required a decrease in the penalty parameter.

zero at any solution. Thus, MFCQ fails at all solution points. We admit that creating instances in this manner only produces a certain type of degeneracy, but these models are sufficient for illustrating the robustness of our approach on certain rank-deficient problems.

After solving these models and noting those that were solved by at least one of the algorithms, we obtained a set of 114 problems. The results for these problems are summarized in Figure 4.3. (The plots all have similar meanings as in the previous subsection; refer to Table 4.2.) Clearly, the picture is much different than in §4.2. `IPOPT` is not only less efficient than both `PIPAL-c` and `PIPAL-a`, but it also lags slightly in terms of robustness. `PIPAL-a` and `PIPAL-c` are competitive, but still we see an advantage in terms of both efficiency and robustness when using the aggressive parameter updates implemented in `PIPAL-a`. It is also worthwhile to note that the robustness of `PIPAL-a` was seemingly unaffected with the addition of redundant (and degeneracy-causing) constraints.

4.4 Experiments with infeasible versions of a set of CUTEr problems

We ran our third set of numerical experiments with infeasible variants of the Hock-Schittkowski problems [21]. In particular, we created infeasible instances by manipulating the AMPL models so that, for each constraint $c^i(x) = 0$ or $c^i(x) \leq 0$,

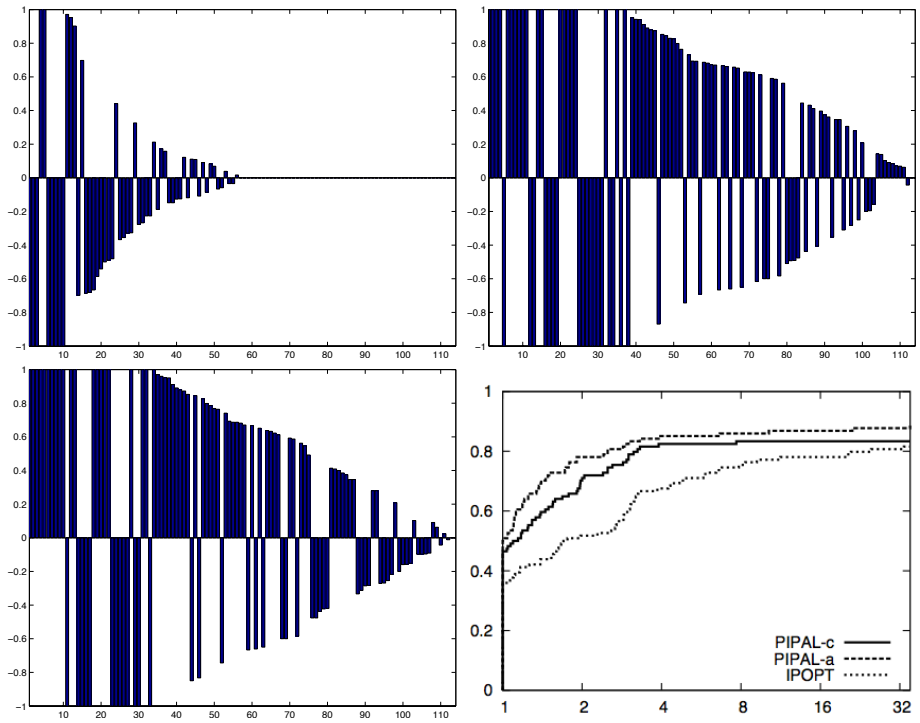

Fig. 4.3 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on degenerate variants of the Hock-Schittkowski problems from the CUTEr collection.

we added

$$c^i(x)^2 \leq -1.$$

Clearly, this made each problem infeasible, and if an algorithm converges to an infeasible stationary point at which an added constraint is active, then the corresponding constraint gradient converges to zero.

After solving these models and noting those that were solved by at least one of the algorithms, we obtained a set of 90 problems. The results for these problems are summarized in Figure 4.4. (The plots all have similar meanings as in the previous subsections; refer to Table 4.2.) Again, the picture is much different than in §4.2. `IPOPT` is as efficient as `PIPAL-a` on a small number of problems, but fails for many of the remaining problems. `PIPAL-a` and `PIPAL-c` are much more robust, providing evidence of the benefits of regularization through penalization. However, again, `PIPAL-a` represents an improvement in efficiency over `PIPAL-c`, suggesting that aggressive updates for ρ and μ are beneficial.

5 Conclusion

We have developed and tested a penalty-interior-point algorithm for large-scale constrained optimization. As a penalty method, the algorithm is designed to handle problems with difficult constraint sets and infeasible problem instances as

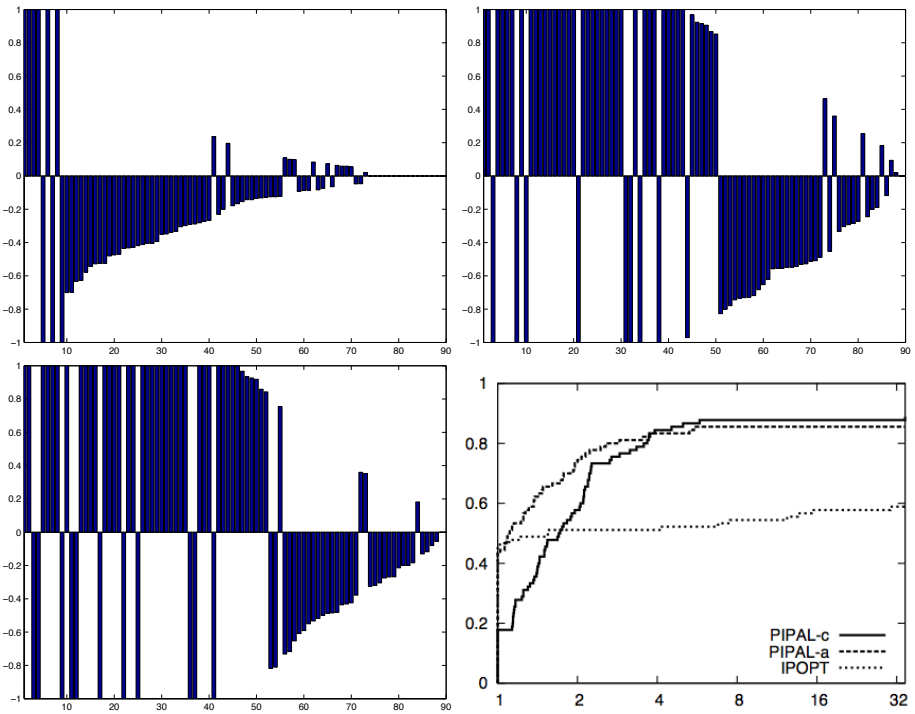

Fig. 4.4 Performance profiles comparing iteration counts for `PIPAL-c`, `PIPAL-a`, and `IPOPT` on infeasible variants of the Hock-Schittkowski problems from the CUTEr collection.

comfortably as it handles feasible problems with easier constraints. As an interior-point method, it is designed to have linear system solves, rather than expensive linear or quadratic optimization subproblem solves, as the main computational component per iteration. It has been stressed throughout this work that in order for a method of this type to be practical as a general-purpose solver, the updates for the penalty and interior-point parameters have to be considered with great care. We believe that the strategy we have developed is novel and sound, and is the main contributor toward the encouraging numerical results that we have obtained, especially for degenerate and infeasible problem instances.

Further enhancements to our framework are needed in order to make our approach and implementation competitive with standard interior-point methodologies applied to large-scale problems with nice characteristics, as well as to make it truly robust on difficult and infeasible problems. Such enhancements may include the tuning of the various input parameters, the incorporation of trust region fallback techniques for cases when the line search leads to significantly cut steps repeatedly throughout the optimization, and further investigations into the removal of unnecessary degrees of freedom when they are not needed.

We believe that there are great opportunities for tailoring our approach to particular problem classes, such as the nonlinear optimization subproblems in mixed-integer frameworks or mathematical programs with complementarity constraints. In the former context our method may be tailored to yield even greater benefits in

the rapid detection of infeasibility, and in the latter context the regularity of our penalty-interior-point subproblem has natural advantages in terms of guaranteeing convergence toward desirable types of stationary points.

Acknowledgements The author would like to thank Richard A. Waltz for numerous productive conversations about the presented methods and for aiding in the software development. He would also like to thank Jorge Nocedal, whose comments greatly improved the presentation.

References

1. H. Y. Benson, A. Sen, and D. F. Shanno. Convergence Analysis of an Interior-Point Method for Nonconvex Nonlinear Programming. *SIAM Journal on Optimization*, submitted, 2008.
2. B. Borchers and J. E. Mitchell. An Improved Branch and Bound Algorithm for Mixed Integer Nonlinear Programming. *Computers and Operations Research*, 21(4):359–367, 1994.
3. M. G. Breitfeld and D. F. Shanno. A Globally Convergent Penalty-Barrier Algorithm for Nonlinear Programming and its Computational Performance. RUTCOR Research Report, RRR 12-94, Rutgers University, New Brunswick, NJ, 1994.
4. M. G. Breitfeld and D. F. Shanno. Computational Experience with Penalty-Barrier Methods for Nonlinear Programming. *Annals of Operations Research*, 62(1):439–463, 1996.
5. R. H. Byrd, F. E. Curtis, and J. Nocedal. Infeasibility Detection and SQP Methods for Nonlinear Optimization. *SIAM Journal on Optimization*, 20(5):2281–2299, 2008.
6. R. H. Byrd, J.-Ch. Gilbert, and J. Nocedal. A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming. *Mathematical Programming*, 89(1):149–185, 2000.
7. R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. An Algorithm for Nonlinear Optimization Using Linear Programming and Equality Constrained Subproblems. *Mathematical Programming*, 100(1):27–48, 2004.
8. R. H. Byrd, N. I. M. Gould, J. Nocedal, and R. A. Waltz. On the Convergence of Successive Linear-Quadratic Programming Algorithms. *SIAM Journal on Optimization*, 16(2):471, 2005.
9. R. H. Byrd, J. Nocedal, and R. A. Waltz. Steering Exact Penalty Methods for Nonlinear Programming. *Optimization Methods and Software*, 23(2):197–213, 2008.
10. L. Chen and D. Goldfarb. Interior-Point l2 Penalty Methods for Nonlinear Programming with Strong Global Convergence Properties. *Mathematical Programming*, 108(1):1–36, 2006.
11. L. Chen and D. Goldfarb. On the Fast Local Convergence of Interior-Point l2 Penalty Methods for Nonlinear Programming, Technical Report, Department of Industrial Engineering and Operations Research, Columbia University, New York, NY, USA, 2006.
12. E. D. Dolan and J. J. Moré. Benchmarking Optimization Software with Performance Profiles. *Mathematical Programming, Series A*, 91:201–213, 2002.
13. A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. Classics in Applied Mathematics. SIAM, Philadelphia, PA, USA, 1990.
14. R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole, 2002.
15. D. Goldfarb, R. A. Polyak, K. Scheinberg, and I. Yuzefovich. A Modified Barrier-Augmented Lagrangian Method for Constrained Minimization. *Computational Optimization and Applications*, 14:55–74, 1999.
16. N. I. M. Gould, I. Bongartz, A. R. Conn, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995.
17. N. I. M. Gould, D. Orban, and Ph. L. Toint. An Interior-Point l1-Penalty Method for Nonlinear Optimization, Technical Report, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, 2003.
18. N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and SifDec: A Constrained and Unconstrained Testing Environment, Revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003.

19. N. I. M. Gould and D. P. Robinson. A Second Derivative SQP Method: Global Convergence. *SIAM Journal on Optimization*, submitted, 2010.
20. I. E. Grossmann. Review of Nonlinear Mixed-Integer and Disjunctive Programming Techniques. *Optimization and Engineering*, 3(3):227–252, 2002.
21. W. Hock and K. Schittkowski. Test Examples for Nonlinear Programming Codes. *Journal of Optimization Theory and Applications*, 30(1):127–129, 1980.
22. X. Hu and D. Ralph. Convergence of a Penalty Method for Mathematical Programming with Complementarity Constraints. *Journal of Optimization Theory and Applications*, 123(2):365–390, 2004.
23. K. Jittorntrum and M. Osborne. A Modified Barrier Function Method with Improved Rate of Convergence for Degenerate Problems. *Journal of the Australian Mathematical Society, Series B*, 21:305–329, 1980.
24. S. Leyffer, G. Lopez-Calva, and J. Nocedal. Interior Methods for Mathematical Programs with Complementarity Constraints. *SIAM Journal on Optimization*, 17(1):52, 2006.
25. J. L. Morales. A numerical study of limited memory BFGS methods. *Applied Mathematics Letters*, 15(4):481–488, 2002.
26. J. Nocedal, A. Wächter, and R. A. Waltz. Adaptive Barrier Update Strategies for Nonlinear Interior Methods. *SIAM Journal on Optimization*, 19(4):1674–1693, 2009.
27. R. A. Polyak. Smooth Optimization Methods for Solving Nonlinear Extremal and Equilibrium Problems with Constraints. In *Eleventh International Symposium on Mathematical Programming*, Bonn, Germany, 1982.
28. R. A. Polyak. Modified Barrier Functions (Theory and Methods). *Mathematical Programming*, 54(1-3):177–222, 1992.
29. R. A. Polyak. Primal-Dual Exterior Point Method for Convex Optimization. *Optimization Methods and Software*, 23(1):141–160, February 2008.
30. I. Quesada and I. E. Grossmann. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers & Chemical Engineering*, 16(10-11):937–947, 1992.
31. H. Scheel and S. Scholtes. Mathematical Programs with Complementarity Constraints: Stationarity, Optimality, and Sensitivity. *Mathematics of Operations Research*, 25(1):1–22, 2000.
32. A. Wächter and L. T. Biegler. Line Search Filter Methods for Nonlinear Programming: Motivation and Global Convergence. *SIAM Journal on Optimization*, 16:1–31, 2005.
33. A. Wächter and L. T. Biegler. On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57, 2006.