

RESEARCH ARTICLE

Mixed Integer Nonlinear Programming Using Interior-Point Methods

Hande Y. Benson

*Department of Decision Sciences
Bennett S. LeBow School of Business
Drexel University
(November 28, 2007)*

In this paper, we outline a bilevel approach for solving mixed integer nonlinear programming problems. The approach combines a branch-and-bound algorithm in the outer iterations and an infeasible interior-point method in the inner iterations. We report on the details of the implementation, including the efficient pruning of the branch-and-bound tree via equilibrium constraints, warmstart strategies for interior-point methods, and the handling of infeasible subproblems, and present numerical results on a standard problem library. Our goal is to demonstrate the viability of interior-point methods, with suitable modifications, to be used within any MINLP framework, and the numerical results provided are quite encouraging.

Keywords: interior-point methods, nonlinear programming, integer programming

1. Introduction.

The optimization problem considered in this paper is the Mixed Integer Nonlinear Programming (MINLP) problem of the form

$$\begin{aligned} \min_{x,y} f(x,y) \\ \text{s.t. } h(x,y) \geq 0 \\ y_j \in D_j, \quad j = 1, \dots, p, \end{aligned} \quad (1)$$

where $x \in \mathbb{R}^n$, y_j lie in discrete sets D_j for $j = 1, \dots, p$, $f : \mathbb{R}^{n+p} \rightarrow \mathbb{R}$ and $h : \mathbb{R}^{n+p} \rightarrow \mathbb{R}^m$ are twice continuously differentiable. When $p = 0$, we have the standard nonlinear programming problem, and when $n = 0$, we have an integer programming problem. Some examples of MINLPs are weapons mix problems with nonlinear utility functions and integer numbers of weapons of each type, pipeline network design problems where there are a small number of pipe diameters to choose from, or circuit design problems where zero-one (binary) variables govern the behavior of the system.

There is quite extensive literature available for solving nonlinear programming problems which only involve continuous variables. Sequential linear programming algorithms ([24]), sequential quadratic programming algorithms ([14], [11]), and interior-point methods ([18], [20], [23]) are all popular approaches with efficient software implementations. The complicating aspect for MINLPs, however, is the

existence of the integer variables. There are two main reasons for this. First, general nonlinear solvers are not equipped to handle such variables in the problem formulation. Second, the solution set is nonconvex and existing methods of nonlinear optimization can only guarantee local optima. In the case of binary variables, there could be up to 2^p local optima for any given problem.

The existing software ([12], [16], [19], [22], [25], [1], [7]) for solving a problem of the form (1) employ a *bilevel* approach. The inner level continuous subproblems are obtained by relaxing the integer constraints and solved using a nonlinear optimization algorithm. The solutions obtained from the nonlinear solver may not have integer values for y . The outer level method is then used to obtain a solution feasible for the mixed integer programming problem. The outer level has two goals: (a) find a feasible solution to the problem and (b) find a global optimum to the problem. For example, the software MINLP [16] uses a sequential quadratic programming algorithm to solve the relaxed nonlinear programming problem and a branch-and-bound algorithm to solve the feasibility problem. SBB [12] is another software package that employs a branch-and-bound algorithm to solve the feasibility problem and can use a number of different nonlinear solvers to solve a relaxed subproblem.

Only one of the MINLP codes mentioned above, [7], uses a pure interior-point method for solving the lower level problem. The main perceived difficulties of an interior-point approach are the lack of warmstart and infeasibility detection capabilities and the handling of fixed variables. Warmstart capabilities are required to increase efficiency in a bilevel code where a sequence of related problems are solved. Reusing information from the previous problem's solution may speed up the solution process considerably. Also in a branch-and-bound approach, subproblems may become infeasible, and it is important to detect such cases early and reliably. For an infeasible interior-point method, it may be advantageous to start and remain infeasible throughout the solution process, and therefore, issuing a certificate of infeasibility for the problem in general is rather difficult. Additionally, during the branch-and-bound method, the upper and lower bounds of a problem may become equal, and such fixed variables lead to the optimal set of Lagrange multipliers being unbounded. Since an interior-point method seeks to find the analytic center of the face of optimal solutions, fixed variables can lead to numerical difficulties.

Despite these difficulties, interior-point methods still remain the state-of-the-art solvers for nonlinear programming. Studies such as [5] numerically demonstrate that interior-point solvers such as IPOPT [23], LOQO [20], and KNITRO [18] are highly efficient and are the only solvers capable of handling very large scale problems. Therefore, it is important to resolve difficulties associated with warmstarting, infeasibility detection, and fixed variables and implement an efficient and robust MINLP solver using interior-point methods.

While the major thrust of this paper is the presentation of an interior-point approach for solving the inner level nonlinear subproblems, we will also examine an improvement to the branch-and-bound algorithm for certain classes of MINLPs. Since it is possible to express a binary restriction on a variable as an equilibrium (or complementarity) constraint, we will first solve a version of the binary MINLP expressed as a mathematical program with equilibrium constraints (MPEC). A naive handling of this approach, however, ensures that the resulting problem has unbounded Lagrange multipliers and is nonconvex. We will handle the first issue via a penalty method as outlined in recent MPEC research such as [2] and [6], which bounds the set of optimal Lagrange multipliers and alleviates the associated numerical difficulties. The second issue is more serious, as general purpose nonlinear solvers guarantee only a local optimum. Nevertheless, the optimal solution yielded

by the MPEC reformulation can serve as a feasible solution to (1) and reduce the number of subproblems to be solved by the algorithm. Thus, in this paper, we propose an efficient way to incorporate the solution of an MPEC into the framework of our bilevel method and show that it can be effective in practice for reducing the size of the branch-and-bound tree.

The outline of the paper is as follows: We start by outlining the bilevel approach combining branch-and-bound in the outer iterations and an infeasible interior-point method in the inner iterations. An overview of the resulting solution algorithm is described in Section 2. Details of the algorithmic improvements, including the use of the MPEC reformulation for pruning the branch-and-bound tree, warmstart strategies for the nonlinear subproblems and the handling of infeasible subproblems and fixed variables are outlined in Sections 3 and 4. We present implementation details for our approach and favorable numerical results on several standard problem libraries in Section 5.

2. Overview of The Solution Algorithm.

The algorithm proposed for solving (1) is outlined in this section. The integrality of y is handled using a branch-and-bound approach in the outer iterations. In the inner iterations, we solve a continuous relaxation of the bound-modified problem. Here, we need to solve a nonlinear programming (NLP) problem, and we employ an interior-point method to do so. We will now discuss the two levels of our solution method. Note that this section is mostly an overview of two well-known methods, and further details can be obtained from [17] for the branch-and-bound method, and from [21] for interior-point methods. We have chosen to include the contents of this section in order to introduce the notation and the framework for the algorithmic improvements to be discussed in Sections 3 and 4.

2.1. Outer Iterations: Branch and Bound

In order to handle the integrality of y , we employ a branch-and-bound approach based on [17]. This approach builds a tree where each node is associated with a relaxation of (1). In fact, each node has a set of upper and lower bounds instead of the integrality requirements for y_j , $j = 1, \dots, p$. The root node, for example, has the property that each constraint

$$y_j \in D_j$$

is replaced with the bounds

$$\min(D_j) \leq y_j \leq \max(D_j),$$

where we assume without loss of generality that each D_j is a set of consecutive integers, and $\min(D_j)$ and $\max(D_j)$ are the (possibly infinite) minimum and maximum elements, respectively, in the set D_j . The resulting problem is a *nonlinear relaxation* of (1) and is the initial problem to be solved. Note that if this problem is unbounded, the assumption of differentiability of the objective and constraint functions ensures that the original MINLP is also unbounded.

For a generic node in the branch-and-bound tree, we denote the upper and lower bounds associated with the node by u and l , respectively. Also, we denote by f^* the lowest objective function value over all feasible solutions of (1) found so far. If

no such feasible solutions have been found, we let $f^* = \infty$. For this generic node, we have three possible outcomes:

- (1) The problem at the node is infeasible: The nonlinear relaxation cannot yield feasible solutions to (1) as it has no feasible solutions itself. Therefore, we do not pursue this node further. Doing so is called *fathoming by infeasibility*.
- (2) The problem has an optimal solution (x^*, y^*) , where $y_j^* \in D_j$, $j = 1, \dots, p$: We have a feasible solution for (1), and we do not pursue this node further. Doing so is called *fathoming by integrality*. If $f(x^*, y^*) < f^*$, then we update f^* by setting it equal to $f(x^*, y^*)$.
- (3) The problem has an optimal solution (x^*, y^*) , where $y_j^* \notin D_j$ for some j : If $f(x^*, y^*) \geq f^*$, we do not pursue this node further, as any feasible solutions its descendants may yield to (1) will have optimal objective function values that are at least as large as $f(x^*, y^*)$. This is called *fathoming by bounds*. Otherwise, we create two children for this node. These children differ from the parent only in the bounds of y_j . That is, one child has the bounds

$$l_j \leq y_j \leq \lfloor y_j^* \rfloor$$

and the other has the bounds

$$\lceil y_j^* \rceil \leq y_j \leq u_j.$$

Thus, the branch-and-bound algorithm systematically partitions the feasible region in search of feasible solutions for (1). When all of the nodes in the tree have been fathomed, we have solved (1). If no feasible solutions have been found, then (1) is declared to be infeasible. Otherwise, the feasible solution with the lowest objective function value is the optimal solution for (1).

Note that in the above discussion, we assume that the optimal objective function values do not improve as we descend down the branch-and-bound tree. This is justified by noting that any bounds added to attain integrality of y will further restrict the solution and cannot improve the objective function. However, when solving a MINLP with nonconvex relaxations, the solution found for a node may be worse than those of its children. This is due to an interior-point method guaranteeing only a local optimum, and these solutions can be in different neighborhoods. Therefore, in this paper, we will limit our discussion and numerical testing to MINLPs with convex relaxations. A problem of the form (1) has convex relaxations if the objective function f is convex and the constraint functions h_i , $i = 1, \dots, m$ are concave.

2.2. Inner Iterations: An Infeasible Interior-Point Method

The branch-and-bound approach described above is based on repeatedly solving nonlinear relaxations of (1) subject to additional bound constraints on the discrete variables. At each node in the branch-and-bound tree, we solve a problem of the form

$$\begin{aligned} \min_{x,y} \quad & f(x, y) \\ \text{s.t.} \quad & h(x, y) \geq 0 \\ & l \leq y \leq u, \end{aligned} \tag{2}$$

where l and u are the (possibly infinite) lower and upper bounds on y . These bounds are set by the outer level algorithm and are modified for each nonlinear relaxation as described above. To solve (2), we use an interior-point method, for

which we now provide an overview. A more detailed explanation can be found in [21].

We start by adding nonnegative slacks: $w \in \mathbb{R}^m$ to the inequality constraints in (2), $g \in \mathbb{R}^p$ to the lower bounds on y , and $t \in \mathbb{R}^p$ to the upper bounds on y .

$$\begin{aligned}
 & \min_{x,y,g,t} f(x,y) \\
 & \text{s.t.} \quad h(x,y) - w = 0 \\
 & \quad \quad y - g = l \\
 & \quad \quad y + t = u \\
 & \quad \quad w, g, t \geq 0.
 \end{aligned} \tag{3}$$

Note that one or both of the bounds on y could be infinite. If any of the bounds is infinite, we simply omit the corresponding constraint.

We incorporate the slacks in a logarithmic barrier term in the objective function and eliminate the nonnegativity constraints:

$$\begin{aligned}
 & \min_{x,y,g,t} f(x,y) - \mu \sum_{i=1}^m \log w_i - \mu \sum_{j=1}^p \log g_j - \mu \sum_{j=1}^p \log t_j \\
 & \text{s.t.} \quad h(x,y) - w = 0 \\
 & \quad \quad y - g = l \\
 & \quad \quad y + t = u,
 \end{aligned} \tag{4}$$

where $\mu > 0$ is the barrier parameter.

Denoting the Lagrange multipliers for the system (4) by (λ, z, s) , the first order conditions for (4) are

$$\begin{aligned}
 & \nabla_x f(x,y) - A_x(x,y)^T \lambda = 0, \\
 & \nabla_y f(x,y) - A_y(x,y)^T \lambda - z + s = 0, \\
 & \quad \quad -\mu e + W \Lambda e = 0, \\
 & \quad \quad -\mu e + G Z e = 0, \\
 & \quad \quad -\mu e + S T e = 0, \\
 & \quad \quad h(x,y) - w = 0, \\
 & \quad \quad y - g = l, \\
 & \quad \quad y + t = u,
 \end{aligned} \tag{5}$$

where e is the vector of all ones of appropriate dimension, $A_x(x,y)$ is the transpose of the Jacobian of the constraints $h(x,y)$ with respect to x , $A_y(x,y)$ is the transpose of the Jacobian of the constraints $h(x,y)$ with respect to y , and W, Λ, G, Z, S, T are the diagonal matrices with the elements of $w, \lambda, g, z, s,$ and t on the diagonal, respectively.

step directions can be obtained from the following formulas:

$$\begin{aligned}
 \Delta w &= W\Lambda^{-1}(\mu W^{-1}e - \lambda - \Delta\lambda) \\
 \Delta g &= GZ^{-1}(\gamma_z - \Delta z) \\
 \Delta t &= S^{-1}T(\gamma_s - \Delta s) \\
 \Delta z &= G^{-1}Z(\nu + GZ^{-1}\gamma_z - \Delta x) \\
 \Delta s &= ST^{-1}(\Delta x - \tau + S^{-1}T\gamma_s).
 \end{aligned} \tag{9}$$

The algorithm then proceeds to a new estimate of the optimum by

$$\begin{aligned}
 x^{(k+1)} &= x^{(k)} + \alpha^{(k)} \Delta x^{(k)}, & y^{(k+1)} &= y^{(k)} + \alpha^{(k)} \Delta y^{(k)}, \\
 \lambda^{(k+1)} &= \lambda^{(k)} + \alpha^{(k)} \Delta \lambda^{(k)}, & w^{(k+1)} &= w^{(k)} + \alpha^{(k)} \Delta w^{(k)}, \\
 g^{(k+1)} &= g^{(k)} + \alpha^{(k)} \Delta g^{(k)}, & t^{(k+1)} &= t^{(k)} + \alpha^{(k)} \Delta t^{(k)}, \\
 z^{(k+1)} &= z^{(k)} + \alpha^{(k)} \Delta z^{(k)}, & s^{(k+1)} &= s^{(k)} + \alpha^{(k)} \Delta s^{(k)},
 \end{aligned} \tag{10}$$

where the superscripts denote the iteration number, $\alpha^{(k)}$ is chosen to ensure that the slacks $(w^{(k+1)}, g^{(k+1)}, t^{(k+1)})$ and the dual variables $(\lambda^{(k+1)}, z^{(k+1)}, s^{(k+1)})$ remain strictly positive and sufficient progress toward optimality and feasibility is attained. At each iteration, the value of the barrier parameter may also be updated as a function of $(W^{(k+1)}\Lambda^{(k+1)}e, G^{(k+1)}Z^{(k+1)}e, T^{(k+1)}S^{(k+1)}e)$. Both the notion of sufficient progress and the exact formula for the barrier parameter update vary from one solver to another, but the general principle remains the same.

The algorithm concludes that it has reached an optimal solution when the primal infeasibility, the dual infeasibility, and the average complementarity are all less than a given tolerance level. For (2), we have that

$$\begin{aligned}
 \text{primal infeasibility} &= \max\{\|\rho\|_\infty, \|\nu\|_\infty, \|\tau\|_\infty\}, \\
 \text{dual infeasibility} &= \max\{\|\sigma_x\|_\infty, \|\sigma_y\|_\infty\}, \\
 \text{average complementarity} &= \frac{w^T \lambda + g^T z + t^T s}{m + 2p},
 \end{aligned}$$

where $\|\cdot\|_\infty$ denotes the infinity norm.

In the next two sections, we will describe the algorithmic components designed to improve the performance of the bilevel approach discussed. Our focus will be on both reducing the number of nonlinear relaxations to be solved and also speeding up the solution of the individual relaxations.

3. Inner Level Algorithmic Improvements.

For the inner-level algorithm, we incorporate the exact primal-dual penalty approach introduced in [3] for linear programming and in [4] for nonlinear programming. This approach serves to improve the efficiency of the interior-point method by allowing for warmstarts and by identifying infeasible problems, two areas of concern. It also bounds optimal sets of Lagrange multipliers, which is useful for handling fixed variables, and will become a crucial part of the improvements to the outer level presented in the next section.

Warmstarting is the use of information obtained during the solution of a problem to solve the subsequent, closely-related problems. For the case of MINLP, warmstarting will refer specifically to setting the initial solution (including primal, dual, and slack variables) of a node to the optimal solution of its parent. Let us first

motivate the need for a special approach by examining what can go wrong when warmstarting using interior-point methods within a branch-and-bound framework.

Let the optimal solution at a node's parent be (x^*, y^*) with the corresponding optimal slack variables (w^*, g^*, t^*) and optimal dual variables (λ^*, z^*, s^*) . Since we have branched on this node using some variable y_j , we have that

$$l_j < y_j^* < u_j,$$

where l and u denote the lower and upper bounds of the discrete variables. This means that

$$\begin{aligned} g_j^* &> 0, & z_j^* &= 0 \\ t_j^* &> 0, & s_j^* &= 0. \end{aligned} \quad (11)$$

Without loss of generality, assume that

$$l_j < y_j^* < \lfloor y_j^* \rfloor,$$

at the current node.

Starting from $(x^*, y^*, w^*, g^*, t^*, \lambda^*, z^*, s^*)$ as the initial values for the primal, dual, and slack variables of the current node, we have that the reduced KKT system (8) at this initial solution is the same as the reduced KKT system at the optimal solution of the parent node, except for

$$\tau_j < 0.$$

Thus, the only possibly different term in the system is $\frac{s_j \tau_j}{t_j}$ on the right-hand side of the equation. By (11), however, we have that

$$\frac{s_j \tau_j}{t_j} = 0,$$

as well. Therefore, there is no change to the reduced KKT system from the optimal solution of the parent node. This means that there is no incentive for the iterates to move, and the algorithm will get stuck at this nonoptimal and, in fact, infeasible solution.

One possible remedy is to simply re-initialize the slack variable s_j away from 0. Doing so would modify both the diagonal of the matrix in (8) and the right-hand side of the reduced KKT system, forcing the variables to move. While this seems like a simple remedy, there are two drawbacks to this approach. First, the initialization is rather arbitrary and may adversely affect the efficiency algorithm at the current node. Secondly, note that simply re-initializing one slack variable may result in negative step directions for other dual and slack variables. Since the interior-point method shortens the steplength to keep such variables strictly positive, the algorithm may still become stuck.

The exact primal-dual penalty method proposed in [3] for linear programming and in [4] for nonlinear programming is a remedy that has been demonstrated to work for warmstarts. This method relaxes the nonnegativity constraint on the dual and slack variables and provides regularization for the matrix in the reduced KKT system (8). Thus, the optimal solution of the parent node can be used without modification to provide a warmstart, the regularization ensures that the variables that need to move indeed make progress, and the algorithm does not become stuck due to the nonnegativity constraints. Details of this approach for general

nonlinear programming problems are given in [4], and we provide an overview here for problems of the form (2).

The primal penalty problem corresponding to (2) has the form

$$\begin{aligned}
 & \min_{x,y,w,g,t,\xi_w,\xi_g,\xi_t} && f(x,y) + c_w^T \xi_w + c_g^T \xi_g + c_t^T \xi_t \\
 \text{s.t.} &&& h(x,y) - w = 0 \\
 &&& y - g = l \\
 &&& y + t = u \\
 &&& -\xi_w \leq w \leq b_\lambda \\
 &&& -\xi_g \leq g \leq b_z \\
 &&& -\xi_t \leq t \leq b_s \\
 &&& \xi_w, \xi_g, \xi_t \geq 0,
 \end{aligned} \tag{12}$$

where $c_w \in \mathbb{R}^m$ and $c_g, c_t \in \mathbb{R}^p$ are the primal penalty parameters, $b_\lambda \in \mathbb{R}^m$ and $b_z, b_s \in \mathbb{R}^p$ are the dual penalty parameters, and $\xi_w \in \mathbb{R}^m$ and $\xi_g, \xi_t \in \mathbb{R}^p$ are the primal relaxation variables. This new form of the primal penalty problem differs from the classical approach presented in [10] in two crucial aspects: (1) The slacks, (w, g, t) , rather than the constraint functions and the bounds themselves are relaxed, and (2) upper bounds are also added to the slacks. Both of these changes are made specifically for warmstarting, as relaxing the slacks removes their nonnegativity constraints and allows for longer steps and the upper bounds serve to relax the dual variables in a similar manner. For further details of the primal and dual problems, as well as a proof of the exactness of the penalty approach, the reader is referred to [3] and [4].

We follow the development of Section 2 in order to present the algorithm to solve (12). The logarithmic barrier problem associated with (12) is

$$\begin{aligned}
 & \min_{x,y,w,g,t,\xi_w,\xi_g,\xi_t} && f(x,y) + c_w^T \xi_w + c_g^T \xi_g + c_t^T \xi_t \\
 &&& -\mu \sum_{i=1}^m \log(\xi_{wi}) - \mu \sum_{i=1}^m \log(w_i + \xi_{wi}) - \mu \sum_{i=1}^m \log(b_{\lambda i} - w_i) \\
 &&& -\mu \sum_{j=1}^p \log(\xi_{gj}) - \mu \sum_{j=1}^p \log(g_j + \xi_{gj}) - \mu \sum_{j=1}^p \log(b_{zj} - g_j) \\
 &&& -\mu \sum_{j=1}^p \log(\xi_{tj}) - \mu \sum_{j=1}^p \log(t_j + \xi_{tj}) - \mu \sum_{j=1}^p \log(b_{sj} - t_j) \\
 \text{s.t.} &&& h(x,y) - w = 0 \\
 &&& y - g = l \\
 &&& y + t = u,
 \end{aligned} \tag{13}$$

where $\mu > 0$ is the barrier parameter. Letting (λ, z, s) once again denote the dual variables associated with the remaining constraints, the first-order conditions for

the Lagrangian of (13) can be written as

$$\begin{aligned}
h(x, y) - w &= 0 \\
y - g &= l \\
y + t &= u \\
\nabla f(x, y) - A(x, y)^T \lambda - z + s &= 0 \\
\lambda - \mu(W + \Xi_w)^{-1}e + \mu(B_\lambda - W)^{-1}e &= 0 \\
z - \mu(G + \Xi_g)^{-1}e + \mu(B_z - G)^{-1}e &= 0 \\
s - \mu(T + \Xi_t)^{-1}e + \mu(B_s - T)^{-1}e &= 0 \\
c_w - \mu\Xi_w^{-1}e - \mu(W + \Xi_w)^{-1}e &= 0 \\
c_g - \mu\Xi_g^{-1}e - \mu(G + \Xi_g)^{-1}e &= 0 \\
c_t - \mu\Xi_t^{-1}e - \mu(T + \Xi_t)^{-1}e &= 0,
\end{aligned}$$

where $\Xi_w, \Xi_g, \Xi_t, B_\lambda, B_z, B_s$ are the diagonal matrices with the entries of $\xi_w, \xi_g, \xi_t, b_\lambda, b_z, b_s$, respectively. Making the substitutions

$$\begin{aligned}
\psi_\lambda &= \mu(B_\lambda - W)^{-1}e \\
\psi_z &= \mu(B_z - G)^{-1}e \\
\psi_s &= \mu(B_s - T)^{-1}e,
\end{aligned}$$

we can rewrite the first-order conditions as

$$h(x, y) - w = 0 \quad (14)$$

$$y - g = l \quad (15)$$

$$y + t = u \quad (16)$$

$$\nabla f(x, y) - A(x, y)^T \lambda - z + s = 0 \quad (17)$$

$$(W + \Xi_w)(\Lambda + \Psi_\lambda)e = \mu e \quad (18)$$

$$(G + \Xi_g)(Z + \Psi_z)e = \mu e \quad (19)$$

$$(T + \Xi_t)(S + \Psi_s)e = \mu e \quad (20)$$

$$\Xi_w(C_w - \Lambda - \Psi_\lambda)e = \mu e \quad (21)$$

$$\Xi_g(C_g - Z - \Psi_z)e = \mu e \quad (22)$$

$$\Xi_t(C_t - S - \Psi_s)e = \mu e \quad (23)$$

$$\Psi_\lambda(B_\lambda - W)e = \mu e \quad (24)$$

$$\Psi_z(B_z - G)e = \mu e \quad (25)$$

$$\Psi_s(B_s - T)e = \mu e, \quad (26)$$

where $\Psi_\lambda, \Psi_z, \Psi_s, C_w, C_g, C_t$ are the diagonal matrices with the entries of $\psi_\lambda, \psi_z, \psi_s, c_w, c_g, c_t$, respectively. Note that the new variables $(\psi_\lambda, \psi_z, \psi_s)$ serve to relax the nonnegativity requirements on the dual variables (λ, z, s) , so we refer to them as the dual relaxation variables.

Applying Newton's Method to (14)-(26), and eliminating the step directions for $w, g, t, z, s, \xi_w, \xi_g, \xi_t, \psi_\lambda, \psi_z$, and ψ_s , the reduced KKT system arising in the solution

of the penalty problem (12) has the same form as (8) with

$$\begin{aligned}
 E &= \left(((\Lambda + \Psi_\lambda)^{-1}(W + \Xi_w) + \Xi_w(C_w - \Lambda - \Psi_\lambda)^{-1})^{-1} + \Psi_\lambda(B_\lambda - W)^{-1} \right)^{-1} \\
 D_g &= \left((Z + \Psi_z)^{-1}(G + \Xi_g) + \Xi_g(C_g - Z - \Psi_z)^{-1} \right)^{-1} + \Psi_z(B_z - G)^{-1} \\
 D_t &= \left((S + \Psi_s)^{-1}(T + \Xi_t) + \Xi_t(C_t - S - \Psi_s)^{-1} \right)^{-1} + \Psi_s(B_s - T)^{-1} \\
 \gamma_w &= \left((\Lambda + \Psi_\lambda)^{-1}(W + \Xi_w) + \Xi_w(C_w - \Lambda - \Psi_\lambda)^{-1} \right)^{-1} \\
 &\quad \left(\mu(\Lambda + \Psi_\lambda)^{-1}e - \mu(C_w - \Lambda - \Psi_\lambda)^{-1}e - w \right) - \left(\mu(B_w - W)^{-1}e - \psi_\lambda \right) \\
 \gamma_z &= \left((Z + \Psi_z)^{-1}(G + \Xi_g) + \Xi_g(C_g - Z - \Psi_z)^{-1} \right)^{-1} \\
 &\quad \left(\mu(Z + \Psi_z)^{-1}e - \mu(C_g - Z - \Psi_z)^{-1}e - g \right) - \left(\mu(B_z - G)^{-1}e - \psi_z \right) \\
 \gamma_s &= \left((S + \Psi_s)^{-1}(T + \Xi_t) + \Xi_t(C_t - S - \Psi_s)^{-1} \right)^{-1} \\
 &\quad \left(\mu(S + \Psi_s)^{-1}e - \mu(C_t - S - \Psi_s)^{-1}e - t \right) - \left(\mu(B_s - T)^{-1}e - \psi_s \right).
 \end{aligned} \tag{27}$$

The steplength, $\alpha^{(k)}$, at each iteration k is chosen to ensure that

$$\begin{array}{llll}
 w^{(k+1)} + \xi_w^{(k+1)} & > 0, & g^{(k+1)} + \xi_g^{(k+1)} & > 0, & t^{(k+1)} + \xi_t^{(k+1)} & > 0, \\
 \lambda^{(k+1)} + \psi_\lambda^{(k+1)} & > 0, & z^{(k+1)} + \psi_z^{(k+1)} & > 0, & s^{(k+1)} + \psi_s^{(k+1)} & > 0, \\
 \xi_w^{(k+1)} & > 0, & \xi_g^{(k+1)} & > 0, & \xi_t^{(k+1)} & > 0, \\
 \psi_\lambda^{(k+1)} & > 0, & \psi_z^{(k+1)} & > 0, & \psi_s^{(k+1)} & > 0, \\
 b_\lambda - w^{(k+1)} & > 0, & b_z - g^{(k+1)} & > 0, & b_s - t^{(k+1)} & > 0, \\
 c_w - \lambda^{(k+1)} - \psi_\lambda^{(k+1)} & > 0, & c_g - z^{(k+1)} - \psi_z^{(k+1)} & > 0, & c_t - s^{(k+1)} - \psi_s^{(k+1)} & > 0,
 \end{array}$$

and sufficient progress toward optimality and feasibility is made. The barrier parameter, μ , may be updated at each iteration as a function of $(W + \Xi_w)(\Lambda + \Psi_\lambda)e$, $(G + \Xi_g)(Z + \Psi_z)e$, $(T + \Xi_t)(S + \Psi_s)e$, $\Xi_w(C_w - \Lambda - \Psi_\lambda)e$, $\Xi_g(C_g - Z - \Psi_z)e$, $\Xi_t(C_t - S - \Psi_s)e$, $\Psi_\lambda(B_\lambda - W)e$, $\Psi_z(B_z - G)e$, and $\Psi_s(B_s - T)e$.

There are several things to note about this approach. First, the sparsity structure of the reduced KKT matrix of the penalty problem is the same as the sparsity structure of (8). There are also no additional function evaluations or other time consuming computations required. This means that solving the penalty problem (12) instead of (2) does not require significant additional computational effort. Second, by modifying D and E , the relaxation/penalty scheme is said to regularize the reduced KKT matrix, providing numerical stability as well as aiding in warmstarting. Third, steplength control no longer relies on the dual and slack variables of the original problem, thereby allowing for longer steps in the initial iterations to ensure that the algorithm does not become stuck.

The primal-dual penalty approach presents an ideal remedy to the warmstarting issues of an interior-point method. For each node, we can use the optimal primal, dual, and slack variable values of its parent as the initial solution, and simply re-initialize the primal and dual relaxation variables in order to facilitate the original variables to move toward a new optimum. The penalty parameters need to be chosen large enough to admit the optimal solution of the node, and warmstart information may be useful to determine appropriate values. They may also need to be updated during the course of the algorithm.

3.1. Setting and Updating the Penalty Parameters

The most important aspect of setting the initial values of the penalty parameters is to ensure that they are sufficiently larger than those components of the current iterate for which they serve as upper bounds. Using our earlier notation, we let the solution at the parent node be (x^*, y^*) with the corresponding optimal slack variables (w^*, g^*, t^*) and optimal dual variables (λ^*, z^*, s^*) . The penalty parameters

are set as follows:

$$\begin{aligned} b_\lambda &= w^* + \kappa_w & c_w &= \lambda^* + \psi_\lambda^{(0)} + \kappa_\lambda \\ b_z &= g^* + \kappa_{gt} & c_g &= z^* + \psi_z^{(0)} + \kappa_{zs} \\ b_s &= t^* + \kappa_{gt} & c_t &= s^* + \psi_s^{(0)} + \kappa_{zs}, \end{aligned}$$

where

$$\begin{aligned} \kappa_w &= \max(h(x^*, y^*), w^*, 1.0) \\ \kappa_\lambda &= \max(\lambda^*, 1.0) \\ \kappa_{gt} &= \max(g^*, t^*, 1.0) \\ \kappa_{zs} &= \max(z^*, s^*, \nabla_y f(x^*, y^*), A_y(x^*, y^*)^T \lambda^*, 1.0). \end{aligned}$$

The relaxation variables are initialized as

$$\begin{aligned} \xi_w^{(0)} &= \beta_p & \psi_\lambda^{(0)} &= \beta_d \\ \xi_g^{(0)} &= \max(l - x^*, 0) + \beta_p & \psi_z^{(0)} &= \beta_d \\ \xi_t^{(0)} &= \max(x^* - u, 0) + \beta_p & \psi_s^{(0)} &= \beta_d \end{aligned} \quad (28)$$

where β_p and β_d are primal and dual constants with default values of $10^{-4} \max(\kappa_w, \kappa_{gt})$ and $10^{-4} \max(\kappa_\lambda, \kappa_{zs})$, respectively. These initializations are generally sufficient after a warmstart to start the penalty method without moving the iterates too far from the current point. Note that the relaxation is performed using a variable, so if a larger relaxation is needed, the variables, $\xi_w, \xi_g, \xi_t, \psi_\lambda, \psi_z$, and ψ_s , will move as necessary.

Since the initial values of the penalty parameters, $b_\lambda, b_z, b_s, c_w, c_g$, and c_t , may not be large enough to admit the optimal solution, we also need an updating scheme for these parameters. Given the relaxation, an optimal solution can always be found for (12), and one possible “static” updating scheme is to solve a problem to optimality and to increase the penalty parameters if their corresponding relaxation variables are not sufficiently close to zero. However, this may require multiple solves of a problem and substantially increase the number of iterations necessary to find the optimal solution. Instead, we can use a “dynamic” updating scheme, where the penalty parameters are checked at the end of each iteration and updated as follows:

$$\begin{aligned} \text{If } w_i^{(k+1)} &> 0.9b_{\lambda_i}^{(k)}, & \text{then } b_{\lambda_i}^{(k+1)} &= 10b_{\lambda_i}^{(k)}, & i &= 1, \dots, m. \\ \text{If } g_j^{(k+1)} &> 0.9b_{z_j}^{(k)}, & \text{then } b_{z_j}^{(k+1)} &= 10b_{z_j}^{(k)}, & j &= 1, \dots, p. \\ \text{If } t_j^{(k+1)} &> 0.9b_{s_j}^{(k)}, & \text{then } b_{s_j}^{(k+1)} &= 10b_{s_j}^{(k)}, & j &= 1, \dots, p. \\ \text{If } \lambda_i^{(k+1)} + \psi_{\lambda_i}^{(k)} &> 0.9c_{wi}^{(k)}, & \text{then } c_{wi}^{(k+1)} &= 10c_{wi}^{(k)}, & i &= 1, \dots, m. \\ \text{If } z_j^{(k+1)} + \psi_{z_j}^{(k)} &> 0.9c_{gj}^{(k)}, & \text{then } c_{gj}^{(k+1)} &= 10c_{gj}^{(k)}, & j &= 1, \dots, p. \\ \text{If } s_j^{(k+1)} + \psi_{s_j}^{(k)} &> 0.9c_{tj}^{(k)}, & \text{then } c_{tj}^{(k+1)} &= 10c_{tj}^{(k)}, & j &= 1, \dots, p. \end{aligned}$$

3.2. Infeasibility Detection.

In the preceding discussion, we established what can go wrong when warmstarting an interior-point method and proposed the exact primal-dual penalty approach as a remedy. Another concern for improving the inner level algorithm within our bilevel framework was the efficient identification of infeasible problems at the nodes of the branch-and-bound tree. For example, a capacitated facility location problem

can have many facility configurations that do not produce enough total capacity to meet demand. Such nodes in the tree should be identified and fathomed quickly.

An interior-point method such as the one described in Section 2 is also known as an *infeasible* interior-point method. This terminology is used to indicate that the initial values for the primal variables x and y are not required to be feasible for the problem. In fact, the iterates are not even required to be feasible until optimality is also attained. Therefore, an infeasible interior-point method can potentially iterate forever when attempting to solve an infeasible problem. In practice, the solver will stop after reaching a preset iteration limit, and the result will be inconclusive. This leads to a failure in the overall bilevel algorithm, as we cannot produce a certificate of optimality or infeasibility at a node.

The primal-dual penalty method described as a remedy for warmstarting can also aid in infeasibility identification. Since all of the slack variables are relaxed, the penalty problem (12) always possesses a feasible solution. In addition, the upper bounds on the slack variables guarantee that an optimal solution to (12) always exists. Therefore, a provably convergent NLP algorithm is guaranteed to find an optimal solution to (12). If this solution has the property that $\xi_{wi} \rightarrow a$, $\xi_{gj} \rightarrow a$, or $\xi_{tj} \rightarrow a$ for at least one $i = 1, \dots, m$ or $j = 1, \dots, p$ for some scalar $a > 0$ as c_{wi} , c_{gj} , or $c_{tj} \rightarrow \infty$, then the original problem is infeasible.

It is impractical to allow a penalty parameter to become infinite. Therefore, a feasibility restoration phase similar to Phase I of SNOPT [13] can be used, in that the problem

$$\begin{aligned}
 & \min_{x,y,w,g,t,\xi_w,\xi_g,\xi_t} && c_w^T \xi_w + c_g^T \xi_g + c_t^T \xi_t \\
 \text{s.t.} &&& h(x,y) - w = 0 \\
 &&& y - g = l \\
 &&& y + t = u \\
 &&& -\xi_w \leq w \leq b_\lambda \\
 &&& -\xi_g \leq g \leq b_z \\
 &&& -\xi_t \leq t \leq b_s \\
 &&& \xi_w, \xi_g, \xi_t \geq 0,
 \end{aligned} \tag{29}$$

is solved in order to minimize infeasibility. It differs from SNOPT's version in that the slack variables are still bounded above by the dual penalty parameters. Since these parameters get updated whenever necessary, we can always find a feasible solution to (29). If the optimal objective function value is nonzero (numerically, greater than the infeasibility tolerance), a certificate of infeasibility can be issued.

3.3. Treatment of Fixed Variables.

Fixed variables present challenges to an interior-point method, in that simply setting the lower and upper bounds of a variable equal to one another may lead to numerical troubles. Consider the following problem:

$$\begin{aligned}
 & \min_y && y^2 \\
 \text{s.t.} &&& 1 \leq y \leq 1.
 \end{aligned} \tag{30}$$

The optimality conditions of (30) are:

$$\begin{aligned} y - g &= 1 \\ y + t &= 1 \\ 2y - z + s &= 0 \\ gz &= 0 \\ ts &= 0. \end{aligned}$$

When $y = 1$, we have both g and t equal 0 and at the optimal solution, the dual variables z and s are free to take on any nonnegative values as long as they satisfy the equality

$$z - s = 2.$$

Therefore, an interior-point method that seeks to find the center of the optimal face of solutions will have z and s tending to infinity and numerical problems will arise as the optimal solution of $y = 1$ is approached.

Note that changing the sparsity structure of the problem by removing the fixed variables may add considerable computational effort to the algorithm by requiring a symbolic Cholesky factorization step at each node. Also, simply replacing the row and column of the reduced KKT matrix corresponding to the fixed variable with zeroes may require additional time during each function, Jacobian, and Hessian evaluation, and possibly lead to numerical difficulties due to rank deficiency.

Above, we outlined penalty methods as a remedy for warmstarting and infeasibility detection. Another benefit of the primal-dual penalty approach is that the slack and dual variables are all bounded above by the dual and primal penalty parameters, respectively. Therefore, the penalty approach also resolves the problem of having unbounded Lagrange multipliers and allows for the incorporation of fixed variables within the interior-point framework.

4. Outer Level Algorithmic Improvements.

As discussed in Section 2, one means of pruning is the branch-and-bound tree is fathoming by bounds. Any node in the tree can be fathomed if its optimal objective function value is not better than that of a feasible solution to (1). Therefore, it is important to find feasible solutions to (1) quickly, and it is also important that these solutions have low objective function values in order to get a substantial reduction in the size of the branch-and-bound tree. Within the outer level algorithm, various search techniques such as depth-first to quickly reach a feasible solution or breadth-first to focus on quality solutions can be employed. However, it is hard to balance these goals, and algorithm performance may vary greatly from one problem to another.

In order to find a feasible solution of good quality for binary MINLPs, we propose the following approach based on the use of equilibrium constraints. The binary constraint

$$y_j \in \{0, 1\} \tag{31}$$

can also be expressed as

$$y_j(1 - y_j) = 0. \tag{32}$$

It is generally not advisable to convert (31) to (32), because the resulting problem is nonconvex with every feasible solution to (1) a local optimum. Thus, a general-purpose NLP algorithm can only guarantee a feasible solution for the original problem and not necessarily an optimal one. However, since our goal here is to find a good feasible solution, using (32) may help to prune the tree.

Note that (32) can be replaced with the equivalent set of conditions

$$-y_j(1 - y_j) \geq 0, \quad y_j \geq 0, \quad (1 - y_j) \geq 0.$$

Thus, when all its discrete variables are constrained to be binary, the problem (1) can be expressed as a mathematical program with equilibrium constraints (MPEC) as follows:

$$\begin{aligned} \min_{x,y} f(x, y) \\ \text{s.t. } h_i(x, y) \geq 0, \quad i = 1, \dots, m \\ -y_j(1 - y_j) \geq 0, \quad j = 1, \dots, p \\ 0 \leq y_j \leq 1, \quad j = 1, \dots, p. \end{aligned} \quad (33)$$

It is well-known that an MPEC always has an unbounded set of optimal Lagrange multipliers. We refer the reader to [2] and [6] for details. As discussed above, the primal-dual penalty approach presented in the previous section is ideally suited for handling such problems since they provide upper bounds for the Lagrange multipliers. Therefore, we will be using the same approach to solve (33).

The problem differs from the nonlinear relaxation (2) only in the addition of the constraints

$$-y_j(1 - y_j) \geq 0, \quad j = 1, \dots, p.$$

Therefore, denoting the Lagrange multipliers of these constraints by $\tilde{\lambda}$ and their slacks by \tilde{w} , the reduced KKT system to be solved at each iteration is given by

$$\begin{bmatrix} -H_{xx} & -H_{xy} & A_x^T & 0 \\ -H_{xy} & -(H_{yy} + D_g + D_t - 2\tilde{\lambda}) & A_y^T & 2Y - I \\ A_x & A_y & E & 0 \\ 0 & 2Y - I & 0 & \tilde{W}\tilde{\Lambda}^{-1} \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \\ \Delta \tilde{\lambda} \end{bmatrix} = \begin{bmatrix} \sigma_y - D_g \hat{\nu} - D_t \hat{\tau} - (2Y - I)\tilde{\lambda} \\ \rho + E\gamma_w \\ \tilde{\rho} + \tilde{W}\tilde{\Lambda}^{-1}\gamma_{\tilde{w}} \end{bmatrix}, \quad (34)$$

where $H_{xx}, H_{xy}, H_{yy}, A_x, A_y, \sigma_x, \sigma_y, \nu, \tau, \rho, \gamma_w, \gamma_z$, and γ_s are defined as in (6), D_g, D_t , and E are defined as in (27), and

$$\begin{aligned} \tilde{\rho} &= Y(I - Y)e + \tilde{w} \\ \gamma_{\tilde{w}} &= \mu\tilde{W}^{-1}e - \tilde{\lambda}. \end{aligned}$$

We can still further reduce this system by letting

$$\Delta \tilde{\lambda} = \tilde{\Lambda}\tilde{W}^{-1} \left(\tilde{\rho} + \tilde{W}\tilde{\Lambda}^{-1}\gamma_{\tilde{w}} - (2Y - I)\Delta y \right),$$

and substituting to obtain

$$\begin{bmatrix} -H_{xx} & -H_{xy} & A_x^T \\ -H_{xy} & -(H_{yy} + \tilde{D}) & A_y^T \\ A_x & A_y & E \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta \lambda \end{bmatrix} = \begin{bmatrix} \sigma_y - D_g \hat{\nu} - D_t \hat{\tau} - (2Y - I)(\tilde{\lambda} + \tilde{\Lambda}\tilde{W}^{-1}\tilde{\rho} + \gamma_{\tilde{w}}) \\ \rho + E\gamma_w \end{bmatrix}, \quad (35)$$

where $\tilde{D} = D_g + D_t - 2\tilde{\Lambda} + (2Y - I)\tilde{\Lambda}\tilde{W}^{-1}(2Y - I)$.

Note that (35) has the same sparsity structure as (8). This ensures that the computational effort required to solve (33) is the same as (2), and, more importantly, switching between the two problems can be done seamlessly. Especially for large problems, an effort equivalent to solving only one additional nonlinear relaxation can be well worth the substantial reduction in the number of nodes in the branch-and-bound tree. The numerical results shown in the next section, in fact, illustrate that savings of up to 44% can be achieved using the MPEC approach.

While this approach is presented for problems where all of the discrete variables are binary, it can also be useful for more general MINLPs. For example, $y \in \{d_1, d_2, \dots, d_r\}$ can be expressed as a linear combination of binary variables z_j , $j = 1, \dots, r$ as

$$y = \sum_{j=1}^r d_j z_j, \quad (36)$$

with the additional constraint that

$$\sum_{j=1}^r z_j = 1. \quad (37)$$

However, in some cases, doing so will greatly increase the size of the problem.

5. Numerical Results

We have tested our MINLP approach on a test set of 61 problems from literature. The problems have been gathered from the MINLPLib [9] test suite, and some of them are also included in the MacMINLP test suite [15]. MacMINLP, compiled by Leyffer, consists of MINLPs written in AMPL. Problems from MINLPLib, which was compiled by the GAMS Corporation, were written in GAMS [8] format and converted to AMPL using the *gams2xx* facility, also provided by the GAMS Corporation. Out of these libraries, we have picked problems for which nondifferentiability did not cause numerical difficulties. That is, rather than analytically verifying differentiability for each model, we removed models which had numerical errors during function, Jacobian, or Hessian evaluations. Similarly, we included all of the problems with convex relaxations and those problems with nonconvex relaxations where the branch-and-bound tree remained consistent.

The algorithm implemented is essentially a branch-and-bound code as described in Section 2 that formulates a continuous relaxation of the form (2) at each node. The interior-point code internally converts (2) to (12) and solves each relaxation until all nodes are fathomed. In addition, if the problem is a binary MINLP, the MPEC formulation is used. After the continuous relaxation solved at the root node, the interior-point code internally converts (1) to (33) and solves it using the penalty approach as described in Section 4. The optimal solution of the MPEC is a feasible solution to (1) and the corresponding objective function value serves to fathom nodes in the branch-and-bound tree.

The MPEC reformulation could have been solved before the start of the branch-and-bound code, in order to avoid interrupting the flow of the algorithm. However, we have found that the solution of the continuous relaxation can provide a good warmstart for solving the MPEC reformulation and may aid in finding a local optimum with a lower objective function value than an optimum found after a user-supplied or system-default initial solution. In fact, we have found that this is

NAME	#node-MPEC	#node	f_{MPEC}^*	f^*	MPEC opt?
alan	6	20	2.925000E+000	2.925000E+000	Yes
batch	38	38	4.128580E+005	2.855065E+005	No
batchdes	8	8	1.812017E+005	1.674277E+005	No
ex1223	10	20	4.579582E+000	4.579582E+000	Yes
ex1223a	4	10	4.579582E+000	4.579582E+000	Yes
ex1223b	6	14	4.579582E+000	4.579582E+000	Yes
ex1225	16	16	4.800000E+001	3.100000E+001	No
ex3	14	26	8.237255E+001	6.800974E+001	No
ex4	158	158	2.891999E+005	-8.064136E+000	No
fac3	70	98	3.643282E+007	3.198231E+007	No
fuel	4	6	8.566119E+003	8.566119E+003	Yes
gbd	2	4	2.200000E+000	2.200000E+000	Yes
gkocis	8	8	-1.411002E+000	-1.923099E+000	No
johnall	2	305	-2.247302E+002	-2.247302E+002	Yes
meanvarx	8	31	1.440406E+001	1.436923E+001	No
nous1	4	4	1.567072E+000	1.567072E+000	Yes
oer	2	2	-1.923098E+000	-1.923099E+000	Yes
procel	4	6	-7.273556E+000	-7.273556E+000	Yes
ravem	198	198	4.071004E+005	2.695902E+005	No
st_e14	6	14	4.579582E+000	4.579582E+000	Yes
st_miqp1	14	14	3.805000E+002	2.810000E+002	No
st_miqp4	2	2	-4.574000E+003	-4.574000E+003	Yes
st_test1	14	14	3.000000E+000	4.367746E-012	No
st_test5	28	28	-1.100000E+002	-1.100000E+002	Yes
st_test6	24	24	6.610000E+002	4.710000E+002	No
synthes1	4	4	7.092732E+000	6.009759E+000	No
synthes2	8	12	7.303531E+001	7.303531E+001	Yes
synthes3	14	22	8.237255E+001	6.800974E+001	No

Table 1. Numerical results on binary MINLPs from the MINLPlib test suite.

a significant advantage gained at a negligible cost, since, as described in Section 4, there is no significant computational effort required when switching back and forth from the solution of the MPEC reformulation. Additionally, if the optimal solution to (1) is found at the root node, there is no need to solve the MPEC.

All nonlinear relaxations (2) and the MPEC models for binary MINLPs (33) have been solved using modified versions of the state-of-the-art interior-point code LOQO [21]. The modifications were made to Version 6.06 of the algorithm and compiled with the AMPL Solver Interface Version 20020221.

We start with a description of the effect of the use of the MPEC reformulation on the number of nodes in the branch-and-bound tree. This effect is only measured on problems where all of the discrete variables are constrained to be binary, as observed in 28 of the 61 problems. In Table 1, the columns labeled “#node-MPEC” and “#node” denote the number of nodes in the branch-and-bound tree with and without the MPEC reformulation, respectively. While the approach does not improve every problem, when it does, the effect can be significant. A problem such as *alan* can go from having 20 nodes to 6, an improvement of 70%. Even more dramatic is the improvement in the problem *johnall*, which had the number of nodes reduced by over 99% from 305 to 2. As *johnall* is one of the larger problems in the test set, and the runtime savings are substantial in this case. We have also listed the optimal objective function values found at the MPEC reformulation (denoted by f_{MPEC}^*), and at the optimal solution of the MINLP (denoted by f^*) in Table 1. For 14 of the 28 problems in Table 1, the MPEC reformulation actually found the optimal solution to the problem, as indicated in the final column labeled “MPEC Opt?” and the computational effort is spent exploring the branch-and-bound tree until all nodes are fathomed by bounds.

In Table 2, we present results highlighting the effect of the primal-dual penalty approach on the interior-point method. In our testing, we have the primal-dual penalty approach determine the subproblems to be solved, and the columns “WarmIters” and “ColdIters” provide the average number of iterations over those subproblems after a warmstart using the primal-dual penalty approach and a coldstart using the original form of the interior-point method, respectively. The column “%Improv” indicates the percentage improvement in the number of iterations. This

number is not always positive, but the warmstart approach is never more than 19% worse than the coldstart approach. The worsening can be remedied in many cases using different initial values for the penalty parameters and the relaxation variables. In the remaining 52 of the 61 problems solved, the percentage improvement can range from 1-87% and is about 43.65% on the average. Especially for a large problem such as *johnall*, the 84.26% savings can represent a significant improvement. In fact, since the MPEC reformulation cannot be solved without the aid of a penalty approach, the overall savings are even more substantial.

We also provide the number of subproblems solved and the number of infeasible problems identified by the penalty approach in the columns “#nodes” and “#inf,” respectively. Since the original formulation of the interior-point method has no mechanism with which to issue a certificate of infeasibility, the coldstart algorithm goes to an iteration limit of 500 for each infeasible subproblem, after making a significant computational effort. This means that for problems such as *du_opt_5* and *fac3*, the branch-and-bound algorithm would fail after encountering an infeasible node. Moreover, the regularization effect of the penalty approach allows the algorithm to solve several subproblems of *batch* and *fuel* whereas the coldstart algorithm goes to its iteration limit.

6. Conclusion.

In this paper, we described the solution of a mixed-integer nonlinear programming problem using an interior-point method within the context of a branch-and-bound algorithm. We resolved the issues of warmstarting, infeasibility detection, and fixed-variable handling for the interior-point method, and we also introduced an equilibrium constraint based approach to reducing the size of the branch-and-bound tree. In doing so, we used the exact primal-dual penalty method of [3] and [4]. The resulting algorithm was implemented using the interior-point code LOQO [21] and tested on a suite of MINLPs. The numerical testing yielded encouraging results.

As discussed, interior-point codes have been shown to be computationally superior to other approaches in studies such as [5] for large problems. Therefore, the proposed approach is especially attractive for large MINLPs, where an interior-point code may be the only means of obtaining a solution to each continuous relaxation in a reasonable amount of time. The use of the primal-dual penalty method further improves the robustness and the efficiency of this approach.

The next step in this study is to incorporate the proposed approach within a more efficient algorithm to handle the integer variables and introduce heuristics for generating feasible solutions quickly. Numerical results in [3] and [4] demonstrate the strong performance of the primal-dual penalty approach under a variety of problem modifications, including the addition of constraints and variables. Thus, we are optimistic that the performance improvements demonstrated in this paper will continue to be applicable when used within any integer programming framework.

References

- [1] K. Abhishek, S. Leyffer, and J. Linderoth, *FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs*, Tech. Rep. Preprint ANL/MCS-P1374-0906, Argonne National Laboratory, Mathematics and Science Division (2006).
- [2] M. Anitescu, *Nonlinear programs with unbounded lagrange multiplier sets*, Tech. Rep. ANL/MCS-P793-0200, Argonne National Labs.
- [3] H. Benson and D. Shanno, *An exact primal-dual penalty method approach to warmstarting interior-*

Problem	WarmIters	ColdIters	%Impr	#Nodes	#Inf	$f(x^*)$
alan	8.40	9.20	8.70	6	1	2.93E+00
batch*	22.06	36.06	38.83	38	4	2.86E+05
batchdes	24.83	30.67	19.02	8	2	1.67E+05
du-opt	25.69	34.68	25.91	131	0	3.56E+00
du-opt5	14.54	24.31	40.19	109	18	8.07E+00
eg_all_s	14.61	24.71	40.88	192	7	7.92E+00
eg_disc_s	14.56	19.21	24.21	8	0	5.76E+00
eg_disc2_s	15	20.86	28.08	14	0	5.64E+00
eg_int_s	13.9	31.8	56.29	10	0	7.46E+00
ex1223	15.20	15.90	4.40	10	0	4.58E+00
ex1223a	14.80	14.60	-1.37	4	0	4.58E+00
ex1223b	12.67	12.83	1.30	6	0	4.58E+00
ex1225	14.92	12.58	-18.54	16	4	3.10E+01
ex3	16.21	16.93	4.22	14	0	6.80E+01
ex4	28.44	23.99	-18.55	158	0	-8.06E+00
fac3	24.66	25.38	2.83	70	18	3.20E+07
fuel*	8.75	63.67	86.26	4	0	8.57E+03
gbd	8.00	7.00	-14.29	2	1	2.20E+00
gear	15.24	17.24	11.60	25	0	8.20E-05
gkocis	11.88	11.75	-1.06	8	0	-1.92E+00
johnall	2.00	13.00	84.26	2	0	-2.25E+02
meanvarx	9.50	14.13	32.74	8	0	1.44E+01
nous1	15.00	21.00	28.57	4	2	1.57E+00
nvs03	14	16.8	16.67	8	3	1.60E+01
nvs04	20.25	53.25	61.97	4	0	7.20E-01
nvs06	10.25	12.25	16.33	4	0	1.77E+00
nvs08	11.5	13.67	15.85	6	0	2.34E+01
nvs10	10	14.75	32.20	4	0	-3.11E+02
nvs11	9.38	14.25	34.21	8	0	-4.31E+02
nvs12	9.25	14.58	36.57	12	0	-4.81E+02
nvs13	10.25	15.5	33.87	20	0	-5.85E+02
nvs14	9.97	24.85	59.89	66	6	-4.04E+04
nvs15	13	18	27.78	8	0	1.00E+00
nvs17	9.69	15.89	39.04	64	0	-1.10E+03
nvs18	9.88	15.02	34.26	48	0	-7.78E+02
nvs19	9.7	16.27	40.35	101	0	-1.10E+03
nvs20	12.13	12.5	3.00	24	0	2.31E+02
nvs23	10.63	27.49	61.34	218	2	-1.13E+03
nvs24	10.3	28.61	64.00	310	4	-1.03E+03
oaer	12.00	15.00	20.00	2	0	-1.92E+00
prob02	14	14	0.00	2	0	1.12E+03
prob03	9	7.75	-16.13	4	0	1.00E+01
procel	8.00	8.50	5.88	4	0	-1.92E+00
ravem	28.03	38.67	27.53	198	6	2.70E+05
st_e14	12.67	12.83	1.30	6	0	4.58E+00
st_miqp1	8.30	7.90	-5.06	14	5	2.81E+02
st_miqp2	9	9.43	4.55	10	3	2.00E+00
st_miqp4	11.50	15.50	25.81	2	0	-4.57E+03
st_test1	3.21	6.14	47.67	14	0	4.37E-12
st_test3	9.62	8.92	-7.76	13	0	-7.00E+00
st_test4	8.63	14.38	40.00	8	0	-7.00E+00
st_test5	10.57	15.14	30.19	28	14	-1.10E+02
st_test6	10.38	12.19	14.87	24	8	4.71E+02
st_test8	9	15.5	41.94	2	0	-2.96E+04
st_testgr1	11.19	16.74	33.13	94	6	-1.28E+01
st_testgr3	12.67	25.86	51.03	1196	198	-2.06E+01
st_testph4	8.25	9	8.33	4	0	-8.05E+01
synthes1	12.00	10.50	-14.29	4	0	6.01E+00
synthes2	14.13	16.88	16.30	8	0	7.30E+01
synthes3	16.31	16.54	1.40	14	0	6.80E+01
tloss	47.34	119.31	60.33	101	15	1.63E+01
OVERALL	15.60	27.67	43.65	3490	327	

Table 2. Comparison of the warmstarting primal-dual penalty approach to coldstarts on the MacMINLP and MINLPlib test suites. WarmIters and ColdIters are the average numbers of warmstart and coldstart iterations, respectively, %Impr is the percentage of improvement in the number of iterations, #Nodes is the number of nodes in the branch-and-bound tree, and #Inf is the number of infeasible nodes identified by the primal-dual penalty approach. “*” indicates that the coldstart approach failed on a subproblem with a solution.

point methods for linear programming, Computational Optimization and Applications (to appear) (2005).

[4] ———, *Interior-point methods for nonconvex nonlinear programming: Regularization and warm-starts*, Computational Optimization and Applications (to appear) (2005).

[5] H. Benson, D. Shanno, and R. Vanderbei, *Interior-point methods for nonconvex nonlinear programming: Filter methods and merit functions*, Computational Optimization and Applications 23 (2002), pp. 257–272.

[6] H.Y. Benson et al., *Interior point algorithms, penalty methods and equilibrium problems*, Computational Optimization and Applications 34(2) (2006), pp. 155–182.

[7] P. Bonami et al., *An algorithmic framework for convex mixed integer nonlinear programs*, Tech. Rep. Research Report RC23771, IBM, to appear in Discrete Optimization, in press.

- [8] A. Brooke, D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, Scientific Press (1988).
- [9] M. Bussieck, A. Drud, and A. Meeraus, *MINLPLib - a collection of test models for mixed-integer nonlinear programming*, INFORMS Journal on Computing 15(1).
- [10] R. Fletcher, *Practical Methods of Optimization*, J. Wiley and Sons, Chichester, England (1987).
- [11] R. Fletcher and S. Leyffer, *User manual for filterSQP*, Tech. Rep. NA-181, University of Dundee Report (1998).
- [12] GAMS, *GAMS-SBB user notes*, march 2001.
- [13] P. Gill, W. Murray, and M. Saunders, *User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming*, Tech. rep., Systems Optimization Laboratory, Stanford University, Stanford, CA (1997).
- [14] ———, *SNOPT: An SQP algorithm for large-scale constrained optimization*, SIAM Journal on Optimization 12 (2002), pp. 979–1006.
- [15] S. Leyffer, *MacMINLP : ampl collection of mixed integer nonlinear programs*, <http://www-unix.mcs.anl.gov/~leyffer/MacMINLP>.
- [16] ———, *Integrating SQP and branch-and-bound for mixed integer nonlinear programming*, Tech. Rep. NA-182, Department of Mathematics, University of Dundee (1998).
- [17] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*, Wiley, New York (1988).
- [18] J. Nocedal and R.A. Waltz, *Knitro 2.0 user's manual*, Tech. Rep. OTC 02-2002, Optimization Technology Center, Northwestern University (2002).
- [19] N. Sahinidis, *Baron: A general purpose global optimization software package*, Journal of Global Optimization 8(2) (1996), pp. 201–205.
- [20] R. Vanderbei, *LOQO user's manual—version 3.10*, Optimization Methods and Software 12 (1999), pp. 485–514.
- [21] R. Vanderbei and D. Shanno, *An interior-point algorithm for nonconvex nonlinear programming*, Computational Optimization and Applications 13 (1999), pp. 231–252.
- [22] J. Viswanathan and I. Grossman, *A combined penalty function and outer approximation method for MINLP optimization*, Computers and Chemical Engineering 14 (1990), pp. 769–782.
- [23] A. Wächter and L.T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Tech. Rep. RC 23149, IBM T. J. Watson Research Center, Yorktown, USA (2004).
- [24] R.A. Waltz, *Algorithms for large-scale nonlinear optimization* (2002), PhD. dissertation. Dept. of Electrical and Computer Engineering, Northwestern University, Evanston, IL, USA.
- [25] T. Westerlund and K. Lundqvist, *Alpha-ecp version 5.01: An interactive minlp-solver based on the extended cutting plane method*, Tech. Rep. 01-178-A, Process Design Laboratory at Abo Akademii University (2001).