Computational Experience of an Interior-Point Algorithm in a Parallel Branch-and-Cut Framework

Eva K. Lee* and John E. Mitchell†

Abstract

An interior-point algorithm within a branch-and-bound framework for solving nonlinear mixed integer programs is described. In contrast to solving the relaxation to optimality at each tree node, the relaxation is only solved to near-optimality. Analogous to using advanced bases for warmstart solutions in the case of linear MIP, a "dynamic" collection of warmstart vectors is kept. Computational results on various classes of nonlinear mixed integer programs are presented.

1 Introduction

Branch-and-bound is a classical approach for solving *linear* mixed integer programs. While the approach is applicable to nonlinear MIPs, there has been much less emphasis on the nonlinear case among the research community. Some work in this direction is described in Borchers and Mitchell [6, 7] and Sahinidis [21], and elsewhere. (See the survey paper by Hansen, Jaumard and Mathon [13].) Borchers and Mitchell [5] describe computational results with the use of an interior point branch and bound method for linear mixed integer programming problems, and Mitchell [18, 19] surveys the use of interior point methods to solve integer programming and combinatorial optimization problems. Other methods for mixed integer nonlinear programming include methods based on Bender's decomposition (e.g., see Floudas [11]), and the outer approximation method of Duran and Grossman [9]. Nevertheless, the coupling of nonlinear solvers within the branch-and-bound framework is relatively unexplored. The motivation for this work lies in combining the computational advances in nonlinear programming within the basic branch-and-bound framework to tackle 0/1 mixed integer programs with nonlinearities appearing in either the objective or constraints. Thus, the general problem considered is of the form

$$\begin{array}{lll} \text{minimize} & f(x) & \\ \text{subject to:} & g(x) & \leq 0 \\ & x_i & \in \{0,1\} \quad \forall\, i = 1,\dots,p, \end{array} \tag{NMIP}$$

where $x \in \Re^n$, $g : \Re^n \to \Re^m$ and $p \leq n$. Without loss of generality, bounds on the variables are integrated into the constraints $g(x) \leq 0$.

In Section 2 we describe the nonlinear interior-point based solver we developed to solve the branch-and-bound subproblems. The integration of the nonlinear solver within the

*Department of Industrial Engineering and Operations Research, Columbia University, New York, New York. Supported in part by NSF/NATO grant GER-9452935, and by NSF grant CCR-9501584. evakylee@columbia.edu

†Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, New York. Supported in part by ONR grant N00014–94–1–0391. mitchj@rpi.edu

branch-and-bound tree is described in Section 3. In Section 4, we briefly describe the parallel implementation, and preliminary numerical results are reported in Section 5.

2 Interior-point based Nonlinear Solver

At each node of the branch-and-bound tree a subproblem — obtained by fixing certain 0/1 variables to zero or one, and relaxing the integral restrictions on the remaining 0/1 variables — must be solved. Thus, each subproblem is a nonlinear programming problem with only continuous variables. As such, it can be written as:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to:} & g(x) \ \leq 0 \end{array} \tag{NLP}$$

There has been a tremendous amount of research among the nonlinear programming community on finding efficient algorithms for (NLP) [3, 4, 10, 17, 20]. Here, we employ the method of sequential quadratic programming (SQP) in solving (NLP). Suppose x^k is the current iterate. The direction d for computing the next iterate is obtained by first solving the quadratic problem:

$$\begin{array}{ll} \text{minimize} & \nabla f(x^k)^T d + \frac{1}{2} d^T B(x^k) d \\ \text{subject to:} & g(x^k) + \nabla g(x^k) d \ \leq 0, \end{array} \tag{QP}$$

where B is an approximation of the hessian of the Lagrangian for (NLP).

This quadratic problem is solved via an interior-point algorithm. The progress of the algorithm and the choice of a steplength is guided by a merit function. In our current implementation, we apply a simple L_1 merit function of the form:

$$f(x) + \rho ||g^+(x)||_1 \tag{MF}$$

where ρ is a constant chosen to be greater than $||y||_\infty$ with y the dual multiplier of (QP); and $g_i^+(x)$ denotes the violation of constraint g_i when evaluated at x. Below we describe the interior-point solver implemented to solve problem (QP) and some special features incorporated in our code.

2.1 Overview of the Interior-point Solver.

Our interior point solver is based on the one developed by Boggs *et al.* [3]. The interior-point solver is developed to solve the quadratic problem (QP). For convenience of notation, we rewrite (QP) as

$$\begin{array}{ll} \text{minimize} & c^T d + \frac{1}{2} d^T Q d \\ \text{subject to:} & Ad + S \ = b \end{array} \tag{QP}$$

where S is the diagonal matrix formed by the slack variables. The first step of the solver is to preprocess the augmented matrix $[A, b]$ in order to remove redundant rows and columns, dominated rows and columns, as well as columns associated with fixed variables. If necessary we modify (QP) to include an artificial variable with a dynamically altered cost. The iterates d^i are always strictly feasible in this modified problem, and the artificial variable is driven to zero as optimality is approached, provided the original (QP) is feasible. At the ith iterate, d^i, the next iterate d^{i+1} is determined by an (O3D) step as well as a Newton centering step. The (O3D) directions are given by the dual affine direction: $s^1 = -(A^TD^2A + \frac{1}{r_0}Q)^{-1}(c + Qx)$;

the centering direction: $s^2 = (A^TD^2A + \frac{1}{r_0}Q)^{-1}A^TDe$;

the order three correction direction: $s^3 = (A^TD^2A + \frac{1}{r_0}Q)^{-1}A^T(As^1 \circ As^1 \circ D^3e)$,

where D denotes the inverse of S, e denotes the vector of 1's, $\circ$ denotes the Hadamard product (i.e., the element-wise product) [14], and

$$r_0 = \frac{-(c + Qd^i)^T(c + Qd^i)}{(c + Qd^i)^T(A^TDe)}.$$

As optimality is approached, the direction s^2 is replaced by a direction that moves the iterate away from the constraint which is closest in the direction s^1. The steplengths taken in each of these directions are obtained by solving the three-dimensional subproblem:

$$\begin{array}{ll} \text{minimize} & \tilde{c}^T\tilde{x} + \frac{1}{2}\tilde{x}^T\tilde{Q}\tilde{x} \\ \text{subject to:} & \tilde{A}\tilde{x} \le \tilde{b} \end{array} \tag{O3D}$$

where $R = [s^1, s^2, s^3]$, $\tilde{c} = R^T(c + Qd^i)$, $\tilde{Q} = R^TQR$, $\tilde{A} = AR$, $\tilde{b} = b - Ad^i$, and $\tilde{x} \in \Re^3$.

Clearly, the origin is feasible for (O3D). However, since there is no guarantee that an optimal solution exists, the O3D solver returns a vector $\tilde{x}^*$ that is either declared to be optimal, or tends to be very large in magnitude due to unboundedness of (O3D). In the case of optimality, we increase the direction d^i by $.95\gamma R\tilde{x}^*$, where γ is the safe steplength (i.e., the largest step possible without violating the constraints in (QP)). Otherwise, $\tilde{x}^*$ is scaled (to avoid taking too large of a step) and d^i is updated as above. We use a Newton centering step after each (O3D) step to center the iterate and to improve the rate of convergence. This centering routine also involves solving a three dimensional subproblem, and it is based on the one described in [3]. The dual variables y are estimated by D^2As^1, with the negative components replaced by zero in order to satisfy the nonnegativity restrictions. The stopping criteria for the interior-point solver are when the dual solution is feasible and the relative duality gap is less than or equal to 10^{-8}; or when the relative improvement in the primal objective is less than 10^{-10}; or when there are no good steps obtained from the (O3D) routine and the Newton recentering step.

2.2 Special Features within the interior-point solver

Scaling of input matrix. Before any numerical calculation, the interior-point solver first performs iterative scaling of the rows (columns) of the constraint matrix of (QP). In particular, each row (column) is scaled by the geometric mean of the maximum and minimum of the absolute values of the nonzero entries of the row (column). We perform this iterative procedure until the ratio the absolute values of the maximum and minimum of the nonzeros in each row is within a certain pre-specified range. We also apply scaling within each O3D step.

Heuristic jumpstart of (QP). The interior-point solver includes a Phase I procedure to obtain a feasible interior starting point. Since Phase I involves calculations with large values, we also include an inexpensive heuristic for finding an initial interior starting point. The heuristic first sets the primal solution to zero. Then it determines the smallest right-hand-side coefficient. If this coefficient is positive, we scale it by a factor between 0 and 1. Otherwise we set the value to -1.0. After that, each non-slack variable is assigned this value, scaled by the number of columns and the norm of the corresponding column in the constraint matrix. The slacks are then assigned values according to the current slack values. If the resulting vector is feasible and is in the interior of the constraint matrix, this point will be used as the warmstart for the interior-point solver. Otherwise Phase I will be called.

Heuristic minimum ordering and diagonal augmentation within the Cholesky Factorization. At the beginning of the (QP) solve, one minimum ordering is performed on the matrix $A^TA + Q$. We include a minimum degree ordering implemented as described in [12], as well as a fast heuristic which orders the rows in nondecreasing order of the number of nonzero entries in each row. During numerical factorization, diagonal elements will be augmented whenever the values fall below a threshold zero tolerance. Within the O3D routines, the step $\tilde{x}$ is solved using dense-cholesky factorization with pivoting. We again augment diagonal elements when needed to ensure positive definiteness in the 3×3 matrix.

Iterative dual refinement. Two refinement steps for improving the dual values of (QP) are performed. The first works within each primal iterate in the interior-point solver. If the estimated dual variable y for a primal iterate is infeasible, its value will be iteratively refined by subtracting the value $D^2A(A^TD^2A + \frac{1}{r_0}Q)^{-1}(A^Ty + c + Qd)$, where $(A^Ty + c + Qd)$ is the amount of the dual violation at the current dual estimate. The second refinement occurs before the interior-point solver exits (QP). If the dual solution remains infeasible, it will be recalculated as $-D^2A(A^TD^2A)^{-1}(c + Qd)$.

Update dual variables of NLP. The dual solutions to (QP) can be used as dual solutions to (NLP). We use a slightly more conservative approach, and update the dual solution to (NLP) to be a weighted average of the former dual solution to (NLP) and the solution to (QP).

3 Nonlinear Mixed Integer Solver

The SQP solver described above is embedded within a branch-and-bound framework. At the root node of the branch-and-bound tree, the solver begins by solving the nonlinear programming relaxation of (NMIP). During this process, an interior feasible point will be saved for further use as a "warmstart." After the initial solve, the heuristic is called. If a feasible integer solution is found, the current upper bound is recorded. If the lower bound is lower than the upper bound, we proceed with the branch-and-bound tree search. The branching variables are selected based on either the smallest index among the fractional binary variable, or the most infeasible value (if there is a tie, the one with the smallest index is chosen). The nodes are processed using "best estimate" criteria. (Note that the bound on each node is only an estimate of its lower bound, due to our premature termination of the SQP solve.) For each node at the depth of multiple of 8, we perform a heuristic before branching. Below, we summarize some of the features in our solver.

Preprocessor. The first step in preprocessing involves removal of redundant rows, columns, dominated rows and columns as well as checks for infeasibility. After that, fixed columns are removed and the right-hand-side is updated appropriately. Next we scan for rows with exactly one nonzero entry. If the value for such a nonzero variable satisfies its bounds, the row is eliminated and the bounds for the variable are updated. Otherwise, the problem is infeasible.

Premature termination of SQP solver. Given feasible primal and dual solutions from the SQP solver, we terminate the iterative SQP procedure when the relative duality gap is less than a pre-specified tolerance. For accurate solutions, it is common to set this value to 10^{-8}. To reduce the iteration counts and avoid unnecessary computation, we set this value to 10^{-6} within each node subproblem.

Warmstart within branch-and-bound nodes and heuristics. At the root node of the branch-and-bound tree, an interior feasible point is reserved for warmstarts in other nodes of the tree. Unlike the linear mixed integer programming case, where the advanced basis at each node can be stored inexpensively, it is prohibitively expensive to store a double-precision vector of initial interior points within each tree node. Instead, we store several copies of interior vectors during the tree search. These vectors serve as warmstarts, and they are updated dynamically as the tree search proceeds.

Adaptive Heuristic Procedure. The heuristic employed here is a modification of that described in [1, 2]. The heuristic starts by using the warmstart associated with the current node. During successive solves within the heuristic, this vector is updated appropriately and, as such, can serve as a warmstart for the next solve. As is done when solving (SQP) within each branch-and-bound node, the heuristic is solved to a relative duality gap of 10^{-6} within each consecutive solve. However, a tighter tolerance is employed when more variables are fixed, and a higher degree of accuracy is desired.

4 Parallel Implementation

The initial parallel implementation is developed using TreadMarks[1]. TreadMarks [15] is a parallel programming system that allows distributed memory computing machines to be programmed as if they were shared memory machines. At the startup of the parallel code, one processor is responsible for reading the problem. That processor also solves the initial nonlinear programming relaxation. If the optimal solution is integral feasible, the algorithm is done; otherwise, the heuristic is called. After that, sequential branch-and-bound is performed until the number of active nodes accumulated exceeds a predetermined threshold.

The shared data in our implementation consists of the best lower bound (for maximization problems), its corresponding solution, and the global list of active nodes. For an individual processor, the initial setup consists of reading in a copy of the nonlinear programming relaxation, as well as all the modifications to it after performing preprocessing and scaling at the root. The processors then perform the following procedure repeatedly until the entire list of active nodes is exhausted and every processor is idle, signaling the completion of the parallel processing. Each idle processor fetches an active node from the global list, using best-estimate selection, and reads the current best lower bound. The nonlinear program is then solved. If an integral solution is obtained, this node is fathomed without further branching; otherwise, a local heuristic is called according to the heuristic interval setting. If the heuristic is performed and a better lower bound is obtained, the best lower bound and solution are "updated." If there is no gap between the nonlinear programming objective value and this lower bound, for the current node, the node is fathomed; otherwise, a branching variable is selected and two new nodes corresponding to the selected variable are added to the list of active nodes. Here, accessing to global information is done via the locking mechanism in TreadMarks.

5 Preliminary Results

The entire optimization code is built "in-house" in C. Preliminary tests are performed on five portfolio problems and 12 mixed 0/1 instances from the MIPLIB running on a SUN

[1]TreadMarks is a trademark of Parallel Tools, L.L.C.

Name	Rows	Cols	0/1 var.	Initial LP Obj.	Optimal MIP Obj.	BB nodes	CPU Time (secs)
port150	755	300	150	1.49695	1.496950	49	902.6
port100	505	200	100	1.3735	1.37357	29	495.3
port50	255	100	50	1.77815	1.83227	30	90.5
port50b	255	100	50	1.54860	1.548633	78	180.7
port10	55	20	10	2.89533	2.89533	12	19.2
air01	23	771	771	6743.0	6796	4	62.3
bm23	20	27	27	20.57	34	101	328.2
egout	98	141	55	149.589	568.1007	634	1290.2
misc01	54	83	82	57.0	563.5	49	235.5
misc02	39	59	58	1010.0	1690	14	13.0
mod008	6	319	319	290.931	307	194	576.2
mod013	62	96	48	256.016	280.95	120	305.7
p0033	16	33	33	2520.57	3089	62	85.2
p0040	23	40	40	61796.55	62027	12	20.3
rgn	24	180	100	48.799	82.199	15	139.7
stein15	13	9	15	7.0	9	23	34.2
stein27	118	27	27	13.0	18	620	543.2

TABLE 1
Test Problem Descriptions and Branch-and-Bound Statistics

workstation SPARC20/M61 (50MHZ). The portfolio problems are of the form

$$\begin{array}{llrll}
\text{minimize} & x^TQx & & & \\
\text{subject to:} & x_i & -y_i & \leq 0 & \forall\ i = 1,\ldots,n \\
 & & \sum_{i=1}^n y_i & \leq K & \\
 & \sum_{i=1}^n a_i x_i & & = M & \\
 & \sum_{i=1}^n x_i & & = 1 & \\
 & & y_i & \in \{0,1\} & \forall\ i = 1,\ldots,n \\
 & & x_i & \geq 0 & \forall\ i = 1,\ldots,n
\end{array}$$

where Q is a dense positive definite matrix which represents the covariance between different commodities.

The preliminary results on the sequential code are quite encouraging. In particular, the warmstarts within each branch-and-bound node, as well as the dual refinement prove to reduce iteration counts within the interior-point solver, by as much as three to four folds in the reduction of overall interior-point iterations. In the parallel runs, close to linear speedup is observed in problems which require over 100 CPU seconds. Present algorithmic work includes improving the robustness and capability of the solver (to handle nonconvex objective and constraint functions); as well as parallel implementation via MPI parallel interface. The solution of nonconvex problems will obviously require the use of a method for generating reliable lower bounds on the global optimum of the nonlinear programming relaxation of the problem. Further numerical tests will be conducted on different nonlinear mixed integer programs arising in real applications.

Table 2
Speedup on 4 SPARC20/M61

Name	2	3	4
port150	1.94	2.76	3.60
port100	1.90	2.80	3.41
port50	1.67	2.41	2.23
port50b	1.78	2.45	3.22
port10	1.34	1.21	1.90
air01	1.12	1.23	1.02
bm23	1.87	2.87	3.56
egout	2.02	2.97	3.87
misc01	1.56	1.98	2.33
misc02	1.23	1.20	1.01
mod008	1.98	2.67	3.76
mod013	1.78	2.77	3.65
p0033	1.77	2.56	3.41
p0040	1.43	1.21	1.12
rgn	1.57	1.54	1.24
stein15	1.34	1.17	1.23
stein27	1.90	2.78	3.56

References

[1] Bixby, R.E., W. Cook, A. Cox, and E.K. Lee, “Computational experience with parallel mixed integer programming in a distributed environment”, Research Monograph CRPC95-5554, Center for Research on Parallel Computation, Rice University, Houston, Texas 1995.

[2] Bixby, R.E. and E.K. Lee, “Solving a truck dispatching scheduling problem using branch-and-cut”, (1994), to appear *Operations Research*

[3] Boggs, P.T., P.D. Domich, J.E. Rogers, and C. Witzgall, “An interior-point method for general large scale quadratic programming”, *Annals of Operations Research*, 62, 419–538, 1996.

[4] Boggs, P.T., J. W. Tolle, and A.J. Kearsley, “A practical algorithm for general large scale nonlinear optimization problems”, Internal Report (1994), National Institute of Standards and Technology.

[5] Borchers B. and J.E. Mitchell, “Using an interior point method in a branch and bound algorithm for integer programming” Technical Report 195, (1991; revised July 1992), Mathematical Sciences, Rensselaer Polytechnic Institute.

[6] Borchers B. and J.E. Mitchell, “An improved branch-and-bound algorithm for mixed integer nonlinear programs”, *Computers and Operations Research*, 21, 359–367, 1994.

[7] Borchers B. and J.E. Mitchell, “A comparison of branch and bound and outer approximation methods for 0-1 MINLPs”, (1996), to appear in *Computers and Operations Research.*

[8] Domich, P.D., P.T. Boggs, J.E. Rogers, and C. Witzgall, “Optimizing over three-dimensional subspaces in an interior-point method for linear programming”, *Linear Algebra and Its Applications*, 152, 315–342, 1991.

[9] Duran, M.A., I. E. Grossman, “An outer-approximation algorithm for a class of mixed-integer nonlinear programs”, *Mathematical Programming*, 36, 307–339, 1986.

[10] El-Bakry, A.S., R.A. Tapia, T. Tsuchiya, and Y. Zhang, "On the formulation and theory of the newton interior-point method for nonlinear programming", Technical Report (1994). Computational and Applied Mathematics, Rice University, Houston, Texas.

[11] Floudas, C.A., *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press, New York, 1995.

[12] George, A. and J. Liu, *Computer Solution to Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1981.

[13] Hansen, P., B. Jaumard, and V. Mathon, "State-of-the-art survey: constrained nonlinear programming", *ORSA Journal on Computing*, 5, 97–119, 1993.

[14] Horn, R.A. and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, New York, 1985.

[15] P. Keleher, A. Cox, S. Dwarkadas and W. Zwaenepoel, "TreadMarks: Distributed Memory on Standard Workstations and Operating Systems," *Proceedings of the 1994 Winter Usenix Conference*, 115–131, 1994.

[16] Lustig, I., R.E. Marsten, and D.F. Shanno, "On implementing Mehrotra's predictor-corrector interior point method for linear programming", *SIAM Journal on Optimization*, 2, 435–449, 1992.

[17] McCormick, G.P., "The superlinear convergence of a nonlinear primal-dual algorithm", Technical Report T-550/91, School of Engineering and Applied Science, George Washington University, Washington, D.C, 1991.

[18] Mitchell, J.E., "Interior point algorithms for integer programming", in J. Beasley, editor, *Advances in Linear and Integer Programming*, Oxford University Press, 1996.

[19] Mitchell, J.E., "Interior point methods for combinatorial optimization", in T. Terlaky, editor, *Interior Point Methods in Mathematical Programming*, Kluwer Academic Publishers, 1996.

[20] Monteiro, R.C. and S.J. Wright, "A globally and superlinearly convergent potential reduction interior point method for convex programming," Manuscript, 1992.

[21] Sahinidis, N, "BARON: An all-purpose global optimization software package", Report number UILU–ENG–95–4002, Department of Mechanical and Industrial Engineering, University of Illinois, Urbana, Illinois, 1995.