

**Solving Mixed Integer
Nonlinear Programming Problems
Using an Interior Point Method**

Eva K. Lee¹

Industrial and Systems Engineering

Georgia Tech

Atlanta, GA 30332

evakylee@isye.gatech.edu

<http://akula.isye.gatech.edu/~evakylee>

John E. Mitchell²

Department of Mathematical Sciences

Rensselaer Polytechnic Institute

Troy, NY 12180

mitchj@rpi.edu

<http://www.math.rpi.edu/~mitchj>

HPOPT Rotterdam

Thursday, 21 August 1997

¹Supported in part by NSF/NATO grant GER-9452935, and by NSF grant CCR-9501584.

²Supported in part by ONR grant N00014-94-1-0391.

Abstract

We discuss a parallel implementation of an interior point branch-and-cut method for solving mixed integer nonlinear programming problems. We consider warm starting the method at the nodes of the tree and also within the nodes of the tree. Computational results are presented.

1 Overview

- Formulation
- Solving the relaxations
- Branch and bound
- Parallel issues
- Computational results

2 Formulation

We are interested in mixed integer nonlinear programming problems (**MINLPs**) of the form

$$\begin{aligned}
 \min \quad & f(x) \\
 \text{subject to} \quad & g_i(x) \leq 0 \quad \text{for } i = 1, \dots, m \quad (MINLP) \\
 & x_j = 0 \text{ or } 1 \text{ for } j \in J \subseteq \{1, \dots, n\} \\
 & x \in \mathfrak{R}^n
 \end{aligned}$$

The set J contains the indices of variables constrained to be integer. The remaining variables are continuous.

Any **bounds** on variables are included in $g_i(x) \leq 0$.

This includes the bounds

$$0 \leq x_j \leq 1 \text{ for } j \in J$$

The problem is solved using **branch-and-cut**.

After some variables are fixed, we still have a problem of the form (*MINLP*), just with fewer variables.

Relaxation:

We relax each MINLP to

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0 \quad \text{for } i = 1, \dots, m \quad (NLP) \\ & x \in \mathfrak{R}^n \end{aligned}$$

This subproblem is solved using an **SQP** method. The SQP method uses an **interior point method** to solve the quadratic programming subproblems:

1. Find an initial iterate x^0 . Set $k = 0$.
2. Set up a quadratic approximation to (NLP) centered at x^k .
3. Approximately solve this quadratic approximation using an interior point method, finding a direction d^k .
4. Update $x^{k+1} = x^k + \beta d^k$ for some β , set $k = k + 1$.
5. If do not yet have a good enough solution to (NLP) , return to step 2.

A similar approach has been used by Boggs, Tolle, and Kearsley. Other interior point SQP approaches include those of Vanderbei and of Byrd, Hribar, and Nocedal.

3 Solving the relaxations

Direction finding subproblem:

Find a direction d by solving the quadratic program

$$\begin{aligned} \min \quad & c^T d + \frac{1}{2} d^T Q d \\ \text{subject to} \quad & A d + s = b \quad \text{for } i = 1, \dots, m \quad (QP) \\ & d \in \mathfrak{R}^n \\ & s \geq 0 \end{aligned}$$

where

- $c := \nabla f(\bar{x})$
- Q is an approximation to the Hessian of the Lagrangian at \bar{x} .
- $b_i := -g_i(\bar{x})$
- The i th row of A is $\nabla g_i(\bar{x})$.
- s is a vector of slacks.

Solving (QP)

- Use an **interior point method**.
- If necessary, use a dynamically altered **Big- M** method.
Thus, can assume $s > 0$.
- Given an iterate d^i , find a new iterate $d^i + \alpha_1 v^1 + \alpha_2 v^2 + \alpha_3 v^3$ by solving a **three-dimensional sub-problem** with the directions:

– *Dual affine direction:*

$$v^1 := -(A^T D^2 A + \frac{1}{r_0} Q)^{-1} (c + Qx)$$

– *Centering direction:*

$$v^2 := (A^T D^2 A + \frac{1}{r_0} Q)^{-1} A^T D e$$

– *Order three correction direction:*

$$v^3 := (A^T D^2 A + \frac{1}{r_0} Q)^{-1} A^T (A v^1 \circ A v^1 \circ D^3 e)$$

Here, e denotes vector of ones, \circ denotes Hadamard product, r_0 is an approximation to the residual on the objective suggested by Fiacco and McCormick,

$$\text{and } D_{ij} := \begin{cases} 1/s_i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Refinements

- Once we get close to optimality for (QP) , we *replace centering direction* v^2 by a direction moving us away from the closest constraint in the direction v^1 .
- After each descent step, solve a three dimensional subproblem to find a *centering step*.
- *Scaling*: The rows and columns of A are scaled to improve stability.
- *Heuristic jumpstart of (QP)* : Try to avoid Big- M procedure by heuristically looking for a somewhat centered starting iterate for (QP) .
- *Finding the multipliers α_i* : This requires solving a three-variable quadratic programming problem with m constraints. We use an interior point method.
- Only solve (QP) *approximately*, solve more accurately further down in tree.

- *Dual variables:* Estimated by $y := D^2 A v^1$.
 - *Iterative dual refinement:* If y is dual infeasible at an iteration, can perform iterative refinement to attempt to reduce the infeasibility. If y is still dual infeasible when we terminate solution to (QP) , we recalculate y as $-D^2 A (A^T D^2 A)^{-1} (c + Qd)$.
 - *Dual solutions to (NLP) :* If y is dual feasible for (QP) then it can be used as the vector of dual multipliers for (NLP) . Instead, we use a weighted average of y and the old vector of dual multipliers for (NLP) .
- If (QP) is *unbounded*, best multipliers α_i may also be unbounded. In this case, we limit the length of the step.
- Once (QP) has been solved, the *steplength* β is found by using a linesearch on a merit function that considers both the objective function $f(x)$ and the constraints $g_i(x)$.

4 Branch and bound

At each node of branch and bound tree, solve the relaxation (NLP) of the current ($MINLP$). One of four possible **outcomes**:

1. (NLP) is **infeasible**, so fathom.
2. Optimal solution to (NLP) is **feasible** in ($MINLP$) so fathom.
3. Optimal solution to (NLP) has value worse than that of a known solution to ($MINLP$), so fathom by **bounds**.
4. Optimal solution to (NLP) is not feasible in ($MINLP$), but has value better than the best known solution to ($MINLP$): need to keep this node **active**.

Suffices to solve (NLP) **approximately** in every situation except 2.

Refinements:

- *Preprocessing:* Remove redundant rows and columns, dominated rows and columns, check for infeasibility. Remove fixed columns. Eliminate rows with exactly one nonzero entry.
- *Warm starts:* A warm start vector is stored at the root node. A list of interior vectors is developed and used for warm starting. It is impractical to store the optimal solution to every active node. (Contrasts with simplex, where a list of the basic variables can be stored at each node. Note that in simplex, the basis matrix still has to be factored afresh at each node if the parent was not solved very recently.)
- *Primal heuristic:* For each node at a depth of a multiple of 8, perform a heuristic search for a good feasible solution to (*MINLP*).

5 Parallel implementation:

- Use *TreadMarks*.
- At *startup*, one processor is responsible for reading in problem and solving initial relaxation. Sequential branch-and-bound is performed until number of active nodes is sufficiently large.
- *Subsequent nodes*: An idle processor fetches a node from the global list of active nodes and then solves the relaxation.
- *Shared data*: Best upper bound and corresponding integral solution. Global list of active nodes. Set of warmstart vectors.

6 Preliminary Computational Results

MIPLIB Problems:

These are linear problems, solved on a SUN 20/M61 workstation, using a parallel emulator.

Name	Rows	Cols	0/1 var.
air01	23	771	771
bm23	20	27	27
egout	98	141	55
misc01	54	83	82
misc02	39	59	58
mod008	6	319	319
mod013	62	96	48
p0033	16	33	33
p0040	23	40	40
rgn	24	180	100
stein15	13	9	15
stein27	118	27	27

Table 1: MIPLIB Problem Descriptions

Name	Initial	Optimal	BB nodes	CPU Time (secs)
	LP Obj.	MIP Obj.		
air01	6743.0	6796	4	62.3
bm23	20.57	34	101	328.2
egout	149.589	568.1007	634	1290.2
misc01	57.0	563.5	49	235.5
misc02	1010.0	1690	14	13.0
mod008	290.931	307	194	576.2
mod013	256.016	280.95	120	305.7
p0033	2520.57	3089	62	85.2
p0040	61796.55	62027	12	20.3
rgn	48.799	82.199	15	139.7
stein15	7.0	9	23	34.2
stein27	13.0	18	620	543.2

**Table 2: MIPLIB Branch-and-Bound
Statistics**

Portfolio problems (quadratic):

These are linearly constrained problems with a convex quadratic objective function, solved on a SUN 20/M61 workstation, using a parallel emulator.

Name	Rows	Cols	0/1 var.
port150	755	300	150
port100	505	200	100
port50	255	100	50
port50b	255	100	50
port10	55	20	10

Table 3: Portfolio Problem Descriptions

Name	Initial		Optimal	
	LP Obj.	MIP Obj.	BB nodes	CPU Time (secs)
port150	1.49695	1.496950	49	902.6
port100	1.3735	1.37357	29	495.3
port50	1.77815	1.83227	30	90.5
port50b	1.54860	1.548633	78	180.7
port10	2.89533	2.89533	12	19.2

Table 4: Portfolio Branch-and-Bound Statistics

Name	CPU Time (secs)	Aver. ip iters	No warm- start Time	Aver. ip iters
port150	902.6	7	2459.4	21
port100	495.3	6	1100.3	20
port50	90.5	6	350.7	18
port50b	180.7	5	310.2	12
port10	19.2	6	29.4	8
air01	62.3	17	189.2	42
bm23	328.2	7	510.5	15
egout	1290.2	7	3012.3	18
misc01	235.5	5	412.2	11
misc02	13.0	6	40.2	14
mod008	576.2	7	1320.1	16
mod013	305.7	5	810.2	11
p0033	85.2	7	211.2	15
p0040	20.3	6	31.2	8
rgn	139.7	7	402.1	21
stein15	34.2	4	40.3	5
stein27	543.2	5	792.2	8

Table 5: The effect of the warm start

Table 6: Speedup on 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

7 Conclusions and future work:

Preliminary results encouraging because:

- Iteration counts reduced by a factor of two to three at each node because of the efficient warm starts and, to a lesser extent, the dual refinement.
- In the parallel runs, linear speedup is observed in problems that take more than approximately 100 CPU seconds.

Future work:

- Make the code more robust and quicker.
- Extend the capability of the solver to handle nonconvex problems.
- Parallel implementation with message passing interface.