

**Computational experience with interior point
column generation and cutting plane
methods.**

John E. Mitchell

Department of Mathematical Sciences

Rensselaer Polytechnic Institute

Troy, NY 12180 USA

mitchj@rpi.edu

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

SIAM Conference on Optimization,

Atlanta, GA,

Monday, 10 May 1999

Abstract

We describe computational experience with [interior point cutting plane algorithms](#) for linear ordering problems and MAXCUT problems, two [classical integer programming problems](#). We discuss warm starting an interior point method in such a setting, as well as other important features of our methods. We show that the use of an interior point method has enabled the solution of certain problems in far less time than that required by a simplex cutting plane algorithm.

1 Polyhedral Theory

- We have an [integer programming problem](#) of the form

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & x \quad \text{binary} \end{aligned}$$

Assume: c is integral.

- We can equivalently solve the [linear programming](#) problem

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \\ & Gx \leq h \\ & 0 \leq x \leq e \end{aligned}$$

where e denotes the vector of ones, for some appropriate set of constraints $Gx \leq h$.

- Drawback: the number of constraints will be [exponential](#) for any NP-Hard problem. The constraints may [not be known](#) explicitly.
- Even the number of constraints of the form $Ax \leq b$ may be exponential. Eg: Subtour elimination constraints for the traveling salesman problem.

2 Cutting plane algorithm

1. **Initialize** with an LP-relaxation of the integer program.
2. **Solve current relaxation**, using a primal-dual interior point method. This gives a lower bound on the optimal value of the integer program.
3. **Separation**: Look for violated cutting planes and add a subset to the relaxation. Drop any constraints that no longer appear important.
4. **Primal heuristic**: Try to round the fractional solution to the LP relaxation to give a good feasible integer solution. Store the resulting solution if it is better than the best solution found previously.
5. **Check for termination**: If the difference between the lower bound and the value of the best solution found so far is less than one, **STOP** with optimality. If no violated constraints are found and if the difference is greater than one, use branch and bound to complete the solution.
6. **Loop**: return to step 2

Refinements to the cutting plane method:

- Suffices to solve the relaxations **approximately**, gradually tightening the degree of accuracy.
- If primal and dual values are **sufficiently close** that it appears that solving this relaxation will solve the integer program, **do not look** for cutting planes.
- **Restart** after adding cutting planes by backing up the primal solution to a convex combination of $\frac{1}{2}e$ and the current point. This is feasible for the problems considered. Can be generalized. The dual solution is set to an earlier dual solution.
- For hard problems, it is useful to solve **every tenth relaxation** to an accuracy of 10^{-8} .
- The **sparsity** of the Cholesky factors of AA^T should be considered when deciding which cuts to add.
- Usually, proving optimality is harder than finding the optimal solution. For MAXCUT, we found that it was hard even to determine the optimal solution.

Adding a cut algebraically:

- The problem is formulated as the [primal](#) problem.
- Cutting planes are [rows](#) of A .
- Add constraints $Gx \leq g$.
- Modify relaxation with additional slack variables s_g :

$$\begin{aligned} \min \quad & c^T x \\ \text{subject to} \quad & Ax + s = b \\ & \mathbf{G}\mathbf{x} + \mathbf{s}_g = \mathbf{g} \\ & 0 \leq x \leq e, \quad 0 \leq s \leq u, \quad \mathbf{0} \leq \mathbf{s}_g \leq \mathbf{u}_g \end{aligned}$$

- [Restart](#): any point in the interior of the convex hull of feasible integer points is feasible in the new relaxation. For MAXCUT, can take $0.5e$. This point can be improved as the iterations proceed, or a sequence of possible restart points can be stored.

Restarting the dual

- Dual problem is:

$$\begin{aligned}
 \max \quad & b^T y + \mathbf{g}^T \mathbf{y}_g - e^T w_x - u^T w_s - \mathbf{u}_g^T \mathbf{w}_g \\
 \text{subject to} \quad & A^T y + \mathbf{G}^T \mathbf{y}_g + z - w = c \\
 & y + z_s - w_s = 0 \\
 & \mathbf{y}_g + \mathbf{z}_g - \mathbf{w}_g = \mathbf{0} \\
 & z, z_s, \mathbf{z}_g \geq 0, \quad w, w_s, \mathbf{w}_g \geq 0.
 \end{aligned}$$

- Restart with $y_g = 0$, $z_g = w_g = \epsilon e$, say $\epsilon = 10^{-3}$.

3 Computational results for MAXCUT

- Given a graph with edge weights w , **partition** the vertices into two sets such that the sum of the edge weights for edges with one endpoint in each set is as large as possible.
- **Initial relaxation:**

$$\min\{c^T x : 0 \leq x_e \leq 1\} \quad (c = -w)$$

- **Cutting planes from cycles:** Every cycle and every cut intersect in an even number of edges. Gives the following facet defining inequality for every subset F of odd cardinality of every chordless cycle C :

$$x(F) - x(C \setminus F) \leq |F| - 1$$

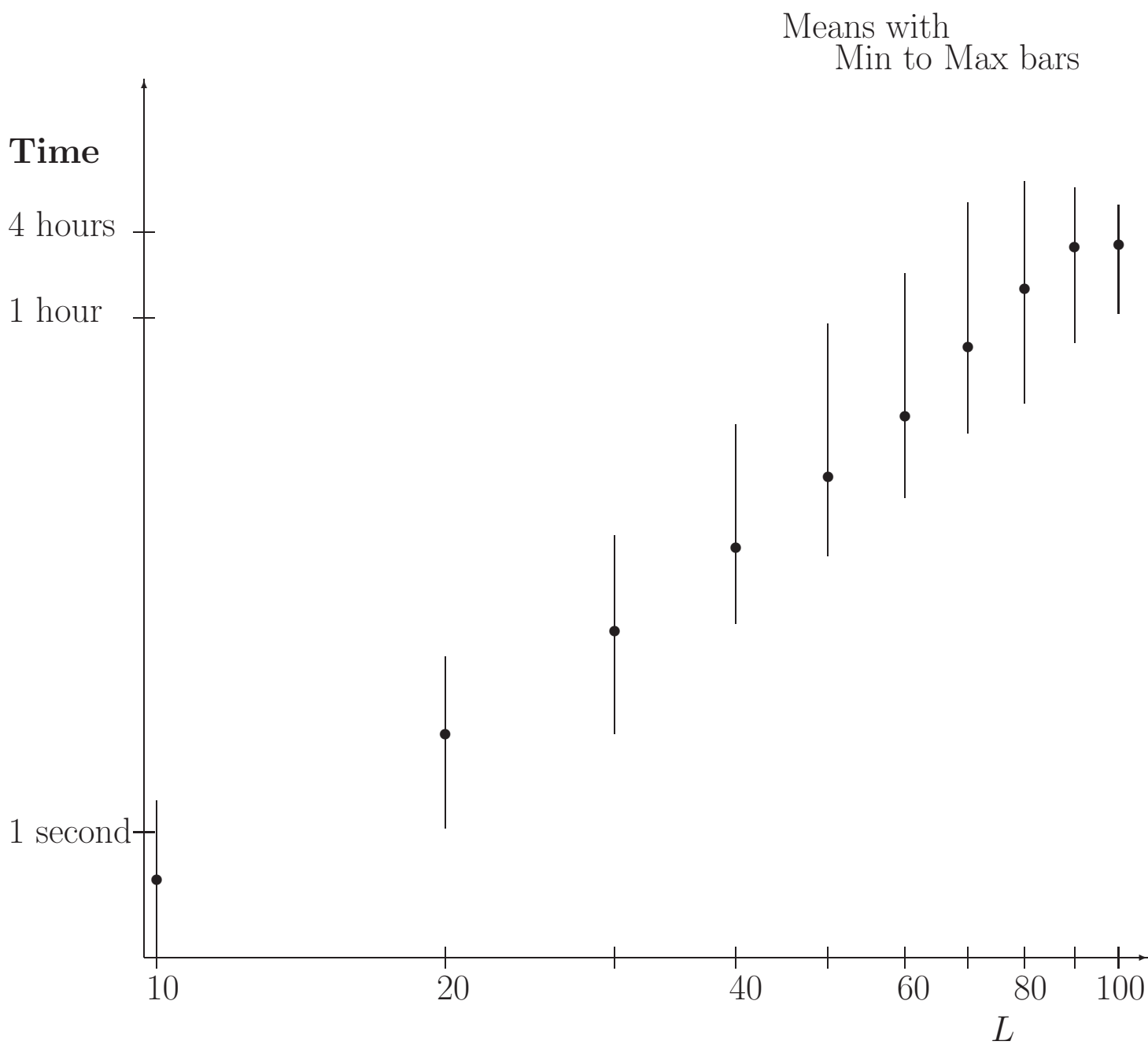
- Any integral vector that satisfies all the cycle inequalities must be the incidence vector of a cut.
- There are also other families of valid inequalities.
- Eg: finding the ground state of an **Ising spin glass**.

- Randomly generated problems
 - Grid sizes up to 100×100 , on a torus. So up to 10000 vertices, 20000 edges.
 - Costs c_{ij} equally likely to be $+1$ or -1 .
- Computational environment
 - All runs performed on a Sun SPARC 20/71 using SunOS. All runtimes will be quoted in [seconds](#).
 - Interior point code written in Fortran. Fortran command **ETIME** used for timings.

L	N_L	Mean	Std Dev	Minimum	Maximum
10	5100	.47	.24	.15	1.68
20	3100	4.84	2.11	1.07	16.55
30	2100	24.49	11.67	4.80	114.80
40	1200	93.08	54.34	27.57	650.43
50	720	290.75	201.46	82.23	3370.88
60	440	772.37	707.68	207.42	7450.18
70	384	2245.92	2277.75	580.68	22768.40
80	310	5787.28	5369.17	935.38	32470.40
90	280	11320.24	6254.31	2474.20	28785.30
100	229	11873.59	3609.73	3855.02	21922.60

Table 1: Time (seconds) to solve Ising spin glass problems

Run times



4 Run times

- Problems of size 100×100 are solved in an average of **3hrs, 20 minutes**. By comparison, a simplex cutting plane code using CPLEX 3.0 on a Sun SPARCstation 10 required up to a **day** to solve problems of size 70×70 . (De Simone *et al*, 1996.)
- Fitting $\log(\text{Time})$ versus $\log(L)$ gives a slope of **approximately 4**. Thus, runtime only grows at rate L^4 . Note that the number of vertices increases at a rate of L^2 . The simplex runtime appeared to increase at a rate of approximately L^6 .
- The number of **iterations** per linear program averages out to around 8. Fewer iterations are required in earlier relaxations and more iterations for later relaxations.
- One possible way to improve the algorithm would be to **crossover** from an interior point method to a simplex method after a certain number of stages.

L	N_L	Stages		Iterations		Cutting Planes	
		Mean	Range	Mean	Range	Mean	Range
10	5100	4	2–10	8	5–19	75	50–200
20	3100	10	4–17	21	6–42	325	250–425
30	2100	16	7–32	39	13–89	800	625–1075
40	1200	24	15–45	64	29–130	1550	1325–1925
50	720	34	22–57	99	51–183	2475	2100–3000
60	440	49	30–117	154	70–549	3700	3075–5575
70	384	64	42–110	216	97–458	5225	4375–6750
80	310	86	55–167	310	154–780	7200	5950–10325
90	280	107	65–147	400	180–668	9400	7450–11150
100	229	110	67–132	391	167–527	11000	8875–12050

Table 2: Results for Ising spin glass problems

5 The linear ordering problem

- Have p objects to place in order.
- If place i before j , pay cost $g(i, j)$.
- Conversely, if place i after j , pay cost $g(j, i)$.
- Choose ordering to minimize the cost.
- [Applications](#) include
 - Triangulation of input-output matrices in economics
 - Archeological seriation
 - Minimizing total weighted completion times in one-machine scheduling
 - Aggregation of individual preferences

Modelling the problem:

- **Variables:** Define

$$x(i, j) = \begin{cases} 1 & \text{if } i \text{ before } j \\ 0 & \text{otherwise} \end{cases}$$

- **Eliminate variables:**

Must have $x(i, j) + x(j, i) = 1$ for each pair $1 \leq i < j \leq p$. Use this to eliminate variables $x(j, i)$, $j > i$.

- **Objective:**

$x(i, j)$, $i < j$, has cost coefficient

$$c(i, j) := g(i, j) - g(j, i).$$

- **Initial relaxation:**

$$\min \quad \sum_{i=1}^{p-1} \sum_{j=i+1}^p c(i, j)x(i, j)$$

$$\text{subject to } 0 \leq x(i, j) \leq 1, \quad 1 \leq i < j \leq p$$

- Use **triangle inequalities** to prevent

i before j before k before i

$$\text{or: } x(i, j) = x(j, k) = x(k, i) = 1$$

Enforced by $x(i, j) + x(j, k) + x(k, i) \leq 2$.

- If the solution to an LP relaxation is integral and satisfies all the triangle inequalities then it **solves the linear ordering problem**.
- Other inequalities exist. We only used triangle inequalities.

6 Crossover

Three different algorithms:

1. Use **interior point method exclusively** to solve the relaxations.
2. Use the **simplex method exclusively** to solve the relaxations.
3. **Combine** the two methods: use the interior point method to solve the first few relaxations and use the simplex method to solve the remaining relaxations.

Experimented with different points to perform the crossover.

7 Computational results

- Randomly generated problems with up to 250 sectors.
- All runs performed on a Sun SPARC 20/71. All runtimes will be quoted in seconds.
- Interior point code written in Fortran. Fortran command **ETIME** used for timings.
- Simplex code written in C. Uses CPLEX 4.0 to solve the relaxations. UNIX command **time** used for timings.
- For the crossover runs, the interior point code wrote the problem out to files and the simplex code read from the files. The times to write out and to read in the problem are included in the runtimes we give.
- The final LP relaxations have between 3000 and 10000 constraints.

Table 3: Iterations on r150a1

Stages	Interior		Simplex		Crossover	
	Cuts	Itns	Cuts	Itns	Cuts	Itns
1	1000	3	2151	2100	1000	3
2	1948	3	1365	977	1948	3
3	1001	3	942	718	1001	3
4	288	3	740	897	288	3
5	248	3	726	1703	248	3
6	194	4	1070	3285	194	4
7	145	3	1157	2363	145	6041
8	445	4	203	3569	143	104
9	79	4	1118	2074	121	132
10	38	3	132	2385	126	98
11	40	4	98	2762	97	131
12	30	4	540	903	110	130
13	34	4	26	880	104	147
14	23	4	16	345	38	126
15	23	4	9	236	97	76
16	55	3	10	127	18	36
17	9	5	1	58	7	1
18	17	5	4	75	8	1

Time for crossover code: 162 seconds for interior
50 seconds for simplex

8 Conclusions

- **Ising spin glass problems:**
 - The interior point code is able to solve **far larger** instances than those previously reported.
 - The ground state energy estimates provided by this model using just the data for large L are lower than previous estimates.

- **Linear ordering problems:**
 - For sufficiently hard problems, combining the two codes performs **significantly better** than either code individually.
 - For larger problems, the interior point and simplex codes require **comparable** time. The interior point solver is a research code. For example, it does not use supernodes when calculating the Cholesky factorization. We believe that current high quality interior point solvers are at least 2–3 times faster than our code for linear programming problems.