

**The computational performance of
interior point cutting plane algorithms
for integer programming problems.**

John E. Mitchell¹

Department of Mathematical Sciences

Rensselaer Polytechnic Institute

Troy, NY 12180 USA

mitchj@rpi.edu

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

Visiting: TWI/SSOR

Delft University of Technology

The Netherlands

Brian Borchers¹

Mathematics Department

New Mexico Tech

Socorro, NM 87801 USA

borchers@nmt.edu

<http://www.nmt.edu/~borchers>

HPOPT98 Workshop, Rotterdam

Thursday, 18 June 1998

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

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 Overview

- The Ising spin glass problem:
 - Definition, polyhedral theory
 - Algorithm
 - Computational results

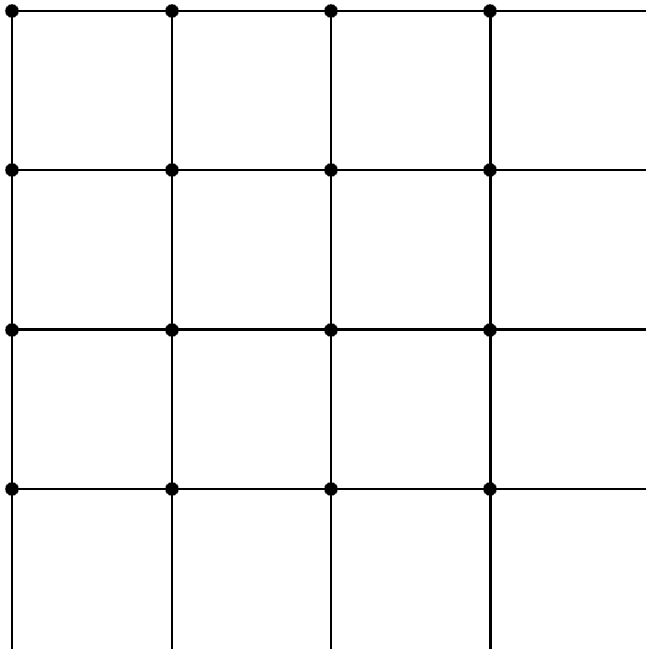
- The linear ordering problem:
 - Definition, polyhedral theory
 - Algorithm
 - Computational results

2 Finding the ground state of an Ising Spin Glass

- References: Grötschel *et al.*, 1985, De Simone *et al.*, 1996.
- Problem in glassy dynamics in statistical physics
- An *Ising spin glass* is a model of a magnetic material, and it consists of a grid of magnetic spins.
- Each spin S_i is in one of two states, which we call “up” and “down”; we assign S_i the value $+1$ if the spin is up and -1 if the spin is down.
- We assume the grid is an $L \times L$ square grid embedded on a torus.
- Further, we assume that the interactions between spins are restricted to neighbours, that is, we consider the short range model with Ising spins S_i .

Graph representation:

4×4 grid on a torus:



- Know **node interactions**, ± 1 . These are edge weights.
- Find **state** of each vertex.
- If edge between neighbouring vertices has value $J_{ij} = +1$, want vertices to be in *same* state.
- If edge between neighbouring vertices has value $J_{ij} = -1$, want vertices to be in *opposite* states.

3 Integer programming model

- Minimize the Hamiltonian of the energy:

$$H := - \sum_{\text{neighbours } i,j} J_{ij} S_i S_j$$

with $S_i = \pm 1$.

- Can be modelled as a **Max Cut** problem: all the *up* vertices on one side of the cut, and all the *down* vertices on the other side, with edge weights derived from J_{ij} .
- So solve

$$\min\{c^T x : x \text{ is the incidence vector of a cut}\}$$

Here, x has one component for each *edge*. The optimal value of this problem is *even*.

Polyhedral theory:

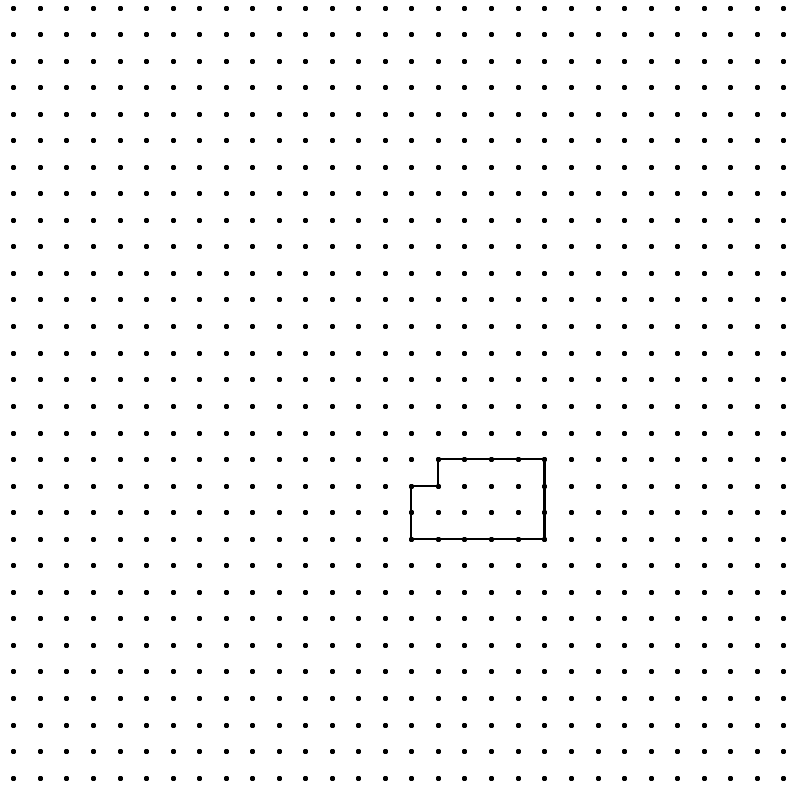
- **Initial relaxation:**

$$\min\{c^T x : 0 \leq x_e \leq 1\}$$

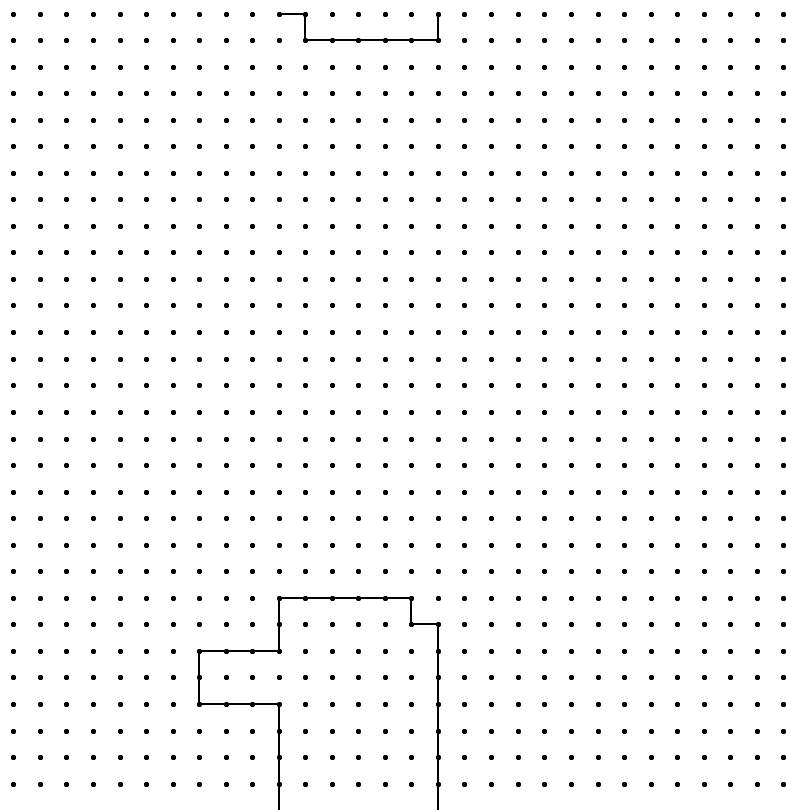
- **Cutting planes from cycles:** Every cycle and every cut intersect in an even number of edges. Gives the following facet defining inequality for every subset F of odd cardinality of every chordless cycle C :

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

- All the **squares** in the grid are chordless cycles. There are also many other chordless cycles.
- 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.



Cycle for added inequality 673
for a 30×30 grid



**Cycle for added inequality 674
for a 30×30 grid**

4 Cutting plane algorithm

1. **Initialize**
2. **Solve current relaxation**, using a primal-dual interior point method. This gives a lower bound on the optimal value of the Ising spin glass problem.
3. **Separation:** Check all cycle inequalities corresponding to squares. Bucket sort resulting violated inequalities by violation and add a subset of constraints to the relaxation. If want more constraints, look at longer cycles. Drop any constraints that no longer appear important.
4. **Primal heuristic:** Look for the incidence vector of a cut close to the solution to the current relaxation. Store the resulting solution if it is better than the best ordering found previously.
5. **Check for termination:** If the difference between the lower bound and the value of the best ordering found so far is less than two, STOP with optimality. If no violated cycle inequalities are found and if the difference is greater than two, 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 MaxCut problem, do not look for cutting planes.
- **Restart** after adding cutting planes by backing up the primal solution to a convex combination of the vector of halves and the current point. The dual solution is set to an earlier dual solution.
- About 40% of the runtime is spent on the primal heuristic. This heuristic looks for chains of vertices which can be switched.
- Every tenth relaxation is solved to an accuracy of 10^{-8} .

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

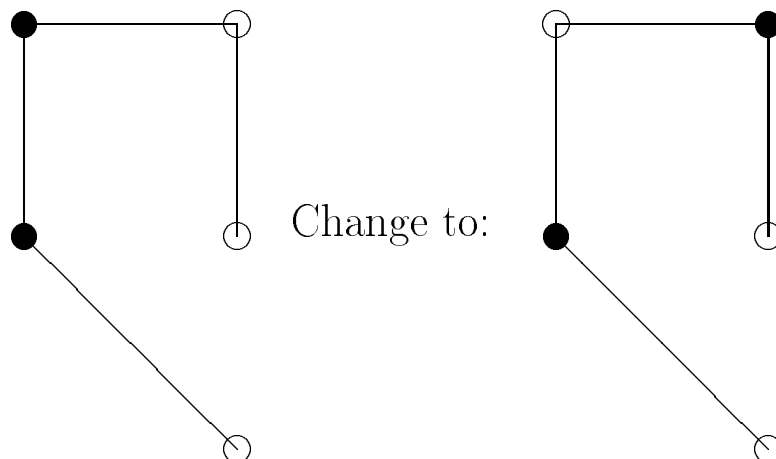
- Dual problem is:

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

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

Primal heuristic

- Construct a cut from the fractional solution x .
- Use local improvement to get a better solution:
 - 1-change: If moving one vertex to the other side of the cut improves the solution, make the change.
 - 2-change: If swapping two vertices on either side of the cut, make the change.
 - k -change: Look for chains of vertices where each vertex can be switched from one side to the other, and switching everything results in an improvement, even though individual switches do nothing. We look for chains of length up to 10 vertices.
 - Example:



Here, no 1-change gives improvement, need a 2-change.

5 Computational results

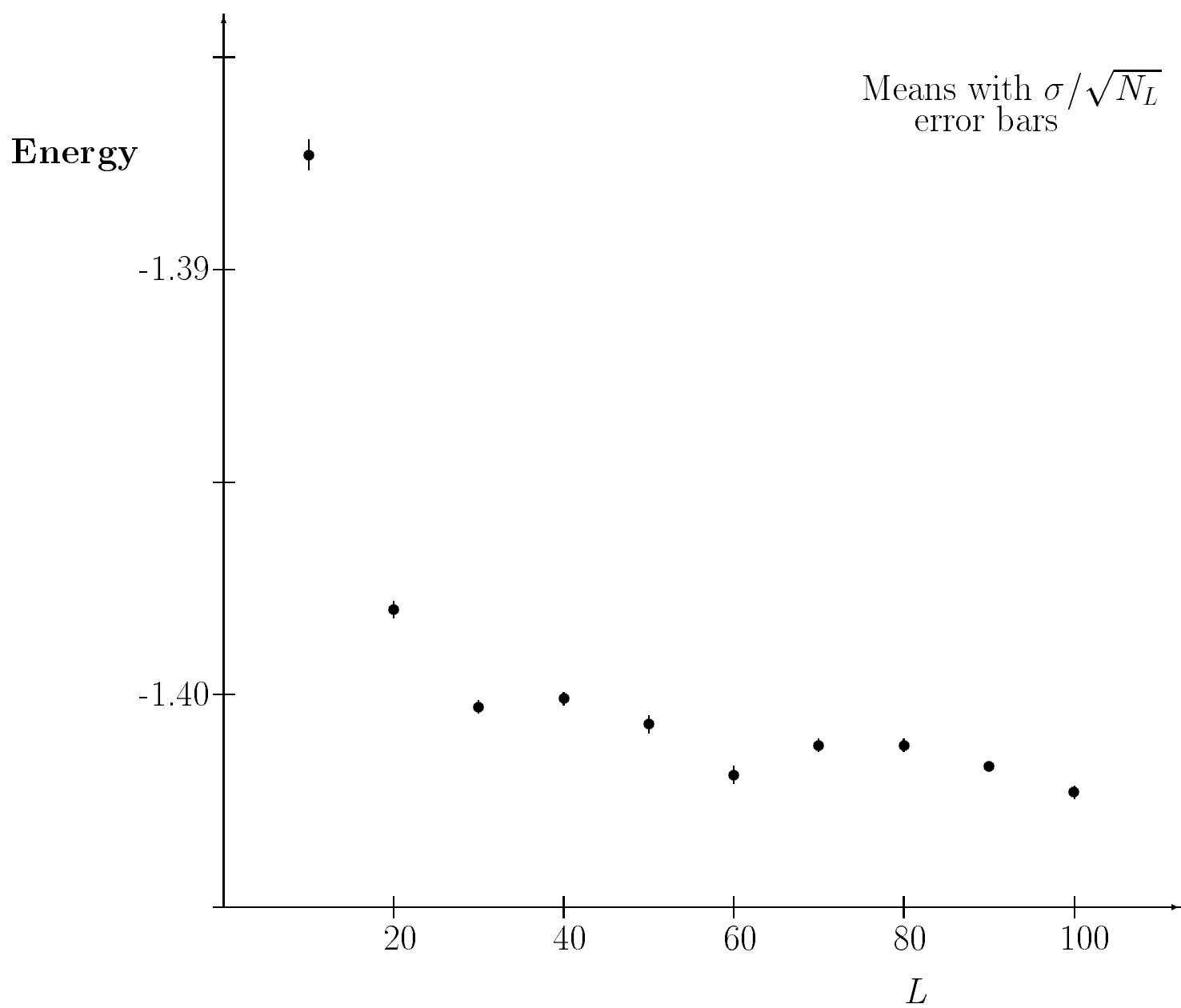
- **Randomly generated problems**

- Grid sizes up to 100×100 .
- Interactions J_{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.

Ground state energy



6 Ground state energy

- **Exponential model:**

$$E_e(L) = E_e^\infty + ce^{-aL}$$

Best fit: $E_e^\infty = -1.3997$.

With just $L \geq 70$: Best fit: $E_e^\infty = -1.4017$.

(De Simone *et al.* (1996) give -1.4007 ± 0.0003 .)

- **Quadratic model:**

$$E_q(L) = E_q^\infty + cL^{-2}$$

Best fit: $E_q^\infty = -1.4016$.

With just $L \geq 70$: Best fit: $E_q^\infty = -1.4031$.

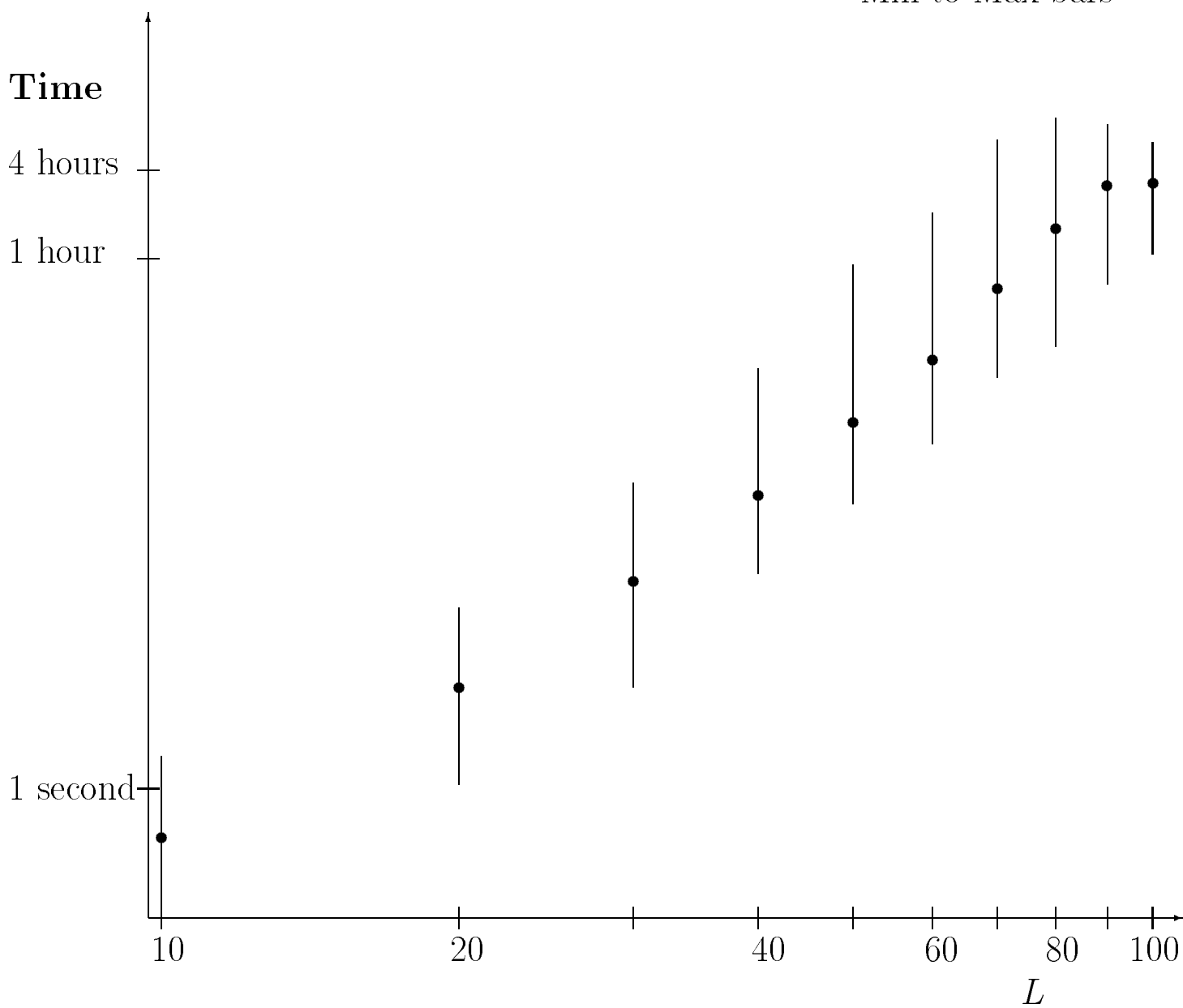
(De Simone *et al.* (1996) give -1.4022 ± 0.0003 .)

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 2: Time (seconds) to solve Ising spin glass problems

Run times

Means with
Min to Max bars



7 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 SPARC-station 10 required up to a **day** to solve problems of size 70×70 . (De Simone *et al*, 1996.)
- Fitting $\log(\textit{Time})$ 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.

8 The linear ordering problem

- Reference: Grötschel *et al.*, 1984.
- **Applications** include
 - Triangulation of input-output matrices in economics
 - Archeological seriation
 - Minimizing total weighted completion times in one-machine scheduling
 - Aggregation of individual preferences
- **Example:** comparing Dutch cheeses:
Each of a group of people perform *pairwise comparisons* between the cheeses. **Objective:** determine the overall preference order for the group.
- **In general:**
 - 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.

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:**

$$\begin{aligned} \min \quad & \sum_{i=1}^{p-1} \sum_{j=i+1}^p c(i, j)x(i, j) \\ \text{subject to} \quad & 0 \leq x(i, j) \leq 1, \quad 1 \leq i < j \leq p \end{aligned}$$

9 Cutting planes

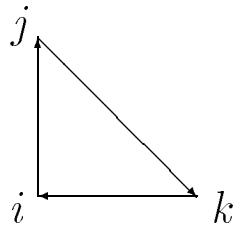
- Use **triangle inequalities** to prevent

i before j before k before i

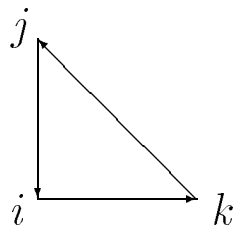
or: $x(i, j) = x(j, k) = x(k, i) = 1$

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

- For $1 \leq i < j < k \leq p$, get two forms:



$$x(i, j) + x(j, k) - x(i, k) \leq 1$$



$$-x(i, j) - x(j, k) + x(i, k) \leq 0$$

- If x is integral and satisfies all the triangle inequalities then it **solves the linear ordering problem**.
- Other inequalities exist. We only used triangle inequalities.

10 Cutting plane algorithm

1. **Initialize:** We assume the data is integral.
2. **Solve current relaxation,** using either a primal-dual interior point method, or the simplex method. This gives a lower bound on the optimal value of the linear ordering problem.
3. **Separation:** Check all triangle inequalities. Bucket sort resulting violated inequalities by violation and add a subset of *arc-disjoint* constraints to the relaxation. Drop any constraints that no longer appear important.
4. **Primal heuristic:** Look for the incidence vector of a linear ordering close to the solution to the current relaxation. Store the resulting solution if it is better than the best ordering found previously.
5. **Check for termination:** If the difference between the lower bound and the value of the best ordering found so far is less than one, STOP with optimality. If no violated triangle inequalities 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 with the interior point cutting plane method:

- Suffices to solve the relaxations **approximately**, gradually tightening the degree of accuracy.
- **Restart** after adding cutting planes by backing up the primal solution to a known feasible solution. Because all possible constraints are known, the vector of halves can be iteratively updated to give a reasonable restart point.
- Because the interior point algorithm terminates each stage early with an interior point, use different criteria in the **bucket sorts** for the two algorithms in deciding which cuts to add.
- It is relatively easy to **find** the optimal solution; most of the work is in **proving optimality**.

Can we use a dual formulation?

- With a primal formulation, a column will become dense if a variable appears in a lot of cutting planes. For the linear ordering problem, each cutting plane has only three nonzero entries, so consider using a dual formulation to attempt to improve sparsity of AA^T .

- Set up the LP relaxation as the **dual** problem:

$$\begin{aligned} \max \quad & b^T y \\ \text{subject to} \quad & A^T y \leq c \quad (DDF) \\ & 0 \leq y \leq e. \end{aligned}$$

- Primal problem is then

$$\begin{aligned} \min \quad & e^T r \quad + \quad c^T x \\ \text{subject to} \quad & r - t + Ax = b \quad (PDF) \\ & r, t, x \geq 0 \end{aligned}$$

with one component of x for each cutting plane.

- In experiments, this appears to be **worse** than the primal formulation, principally because A is tall and thin: for our largest problems, have 31,125 edges and only need approximately 10,000 triangle inequalities.

11 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.

12 Randomly generated problems

- Principal type:
 - Up to 250 objects.
 - * For $i < j$, generate $g(i, j)$ uniformly between 0 and 99.
 - * For $j < i$, generate $g(i, j)$ uniformly between 0 and 39.

Randomly permute so the primal heuristic is not at an advantage.

- Zero out a percentage of the entries.
- Generated six different classes of problems. For each problem within a particular class, we used the same crossover criterion.

- Type B: Divide the items into groups. Then

$$g(i, j) = \beta_1 U[-1, 1] + \beta_2 U[0, 1] \text{sign}(\text{group}(j) - \text{group}(i))$$

where β_1 and β_2 are positive parameters, and $U[., .]$ denotes a uniform distribution.

Note: If all $g(i, j)$ are i.i.d. $U[0, 1]$, then problems with more than about 20 sectors require more than just triangle inequalities. Thus, we take the group size to be 5 or 10.

- Type C: Generate k random permutations. Then $g(i, j)$ is a weighted combination of the number of times i appears before j in the permutations.

Note: If we give the permutations equal weight then combining just three permutations is already hard with as few as 100 sectors.

13 Computational results

- 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.
- CPLEX has an option of using the point provided by the interior point method as a warm start. We found that this was only a very marginal help, and occasionally led to failure to terminate. Therefore, we report results that do not use this feature.

Time to solve with interior point code vs time to solve with simplex code

(No crossover)

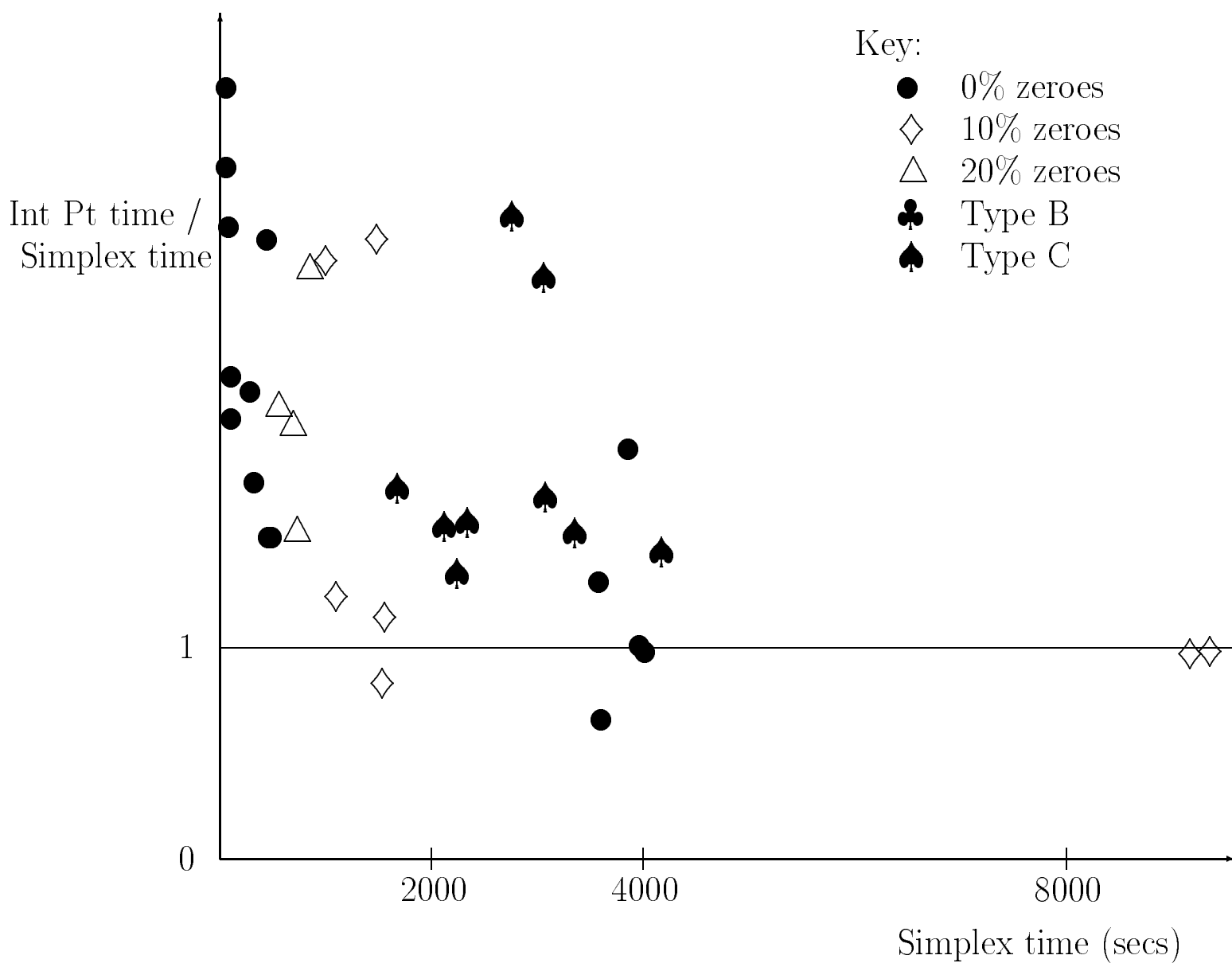


Table 9: Iterations on r150a1

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

Comments:

- Crossover points:
 - **r150.0**, **r200.0**, Type B and Type C: switch after two stages.
 - **r150.1** and **r250.0**: switch after three stages.
 - **r200.1**: switch after add < 600 constraints in a stage. (On average, after 7 stages.)
 - **r100.2**: switch after add < 100 constraints in a stage. (On average, after 7 stages.)
- Each of the criteria resulted in improvement over pure simplex for almost every set of problems.
- Final number of constraints for the pure interior code are approximately:
 - **r100.2**: 3000 constraints.
 - **r150.0**: 4000 constraints.
 - **r150.1**: 5000 constraints.
 - **r200.0**: 6500 constraints.
 - **r200.1**: 8000 constraints.
 - **r250.0**: 10000 constraints.
 - Type B: 9000 constraints.
 - Type C: 8000 constraints.

14 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.

L	N	Need B&B	Mean	Std Dev	Minimum	Maximum
10	5100	0	-1.3873	.0466	-1.5600	-1.2400
20	3100	0	-1.3980	.0228	-1.4700	-1.3200
30	2100	0	-1.4003	.0151	-1.4444	-1.3467
40	1200	0	-1.4001	.0109	-1.4325	-1.3700
50	720	0	-1.4005	.0098	-1.4368	-1.3744
60	440	0	-1.4019	.0077	-1.4233	-1.3783
70	386	2	-1.4012	.0064	-1.4196	-1.3829
80	328	18	-1.4012	.0059	-1.4163	-1.3869
90	290	10	-1.4017	.0043	-1.4128	-1.3916
100	236	7	-1.4023	.0047	-1.4134	-1.3864

Table 1: Ground state energy of Ising spin glass problems

Table 3: Times for the 3 algorithms

Objects	% zeroes	Name	Interior	Simplex	Crossover
150	0	r150a0	185	89	70
		r150b0	201	55	66
		r150c0	219	96	72
		r150d0	224	75	66
		r150e0	203	62	65
		Mean	206	75	68
200	0	r200a0	565	318	200
		r200b0	1196	408	245
		r200c0	610	276	193
		r200d0	713	468	198
		r200e0	690	455	209
		Mean	755	385	209
250	0	r250a0	2382	3593	683
		r250b0	4685	3574	548
		r250c0	3924	4013	574
		r250d0	7486	3860	524
		r250e0	3983	3947	631
		Mean	4492	3797	592

Table 4: Times for the 3 algorithms

Objects	% zeroes	Name	Interior	Simplex	Crossover
100	20	r100a2	1043	487	185
		r100b2	frac	3886	426
		r100c2	2216	794	214
		r100d2	1040	669	132
		r100e2	1322	645	192
		Mean	1405	1296	230
150	10	r150a1	1249	1496	218
		r150b1	1717	1511	165
		r150c1	2723	963	221
		r150d1	4229	1441	237
		r150e1	1316	1058	200
		Mean	2247	1294	208
200	10	r200a1	dnf	10791	1101
		r200b1	9167	9317	886
		r200c1	8856	9142	692
		r200d1	frac	33038	1667
		r200e1	dnf	10687	836
		Mean ²	–	9984	879

² Means for r200.1 omit r200d1.

Table 5: Times for the 3 algorithms

Type	Objects	Name	Interior	Simplex	Crossover
B*	238	group250a	7626	638	383
		group250b	2869	395	249
		group250c	5118	538	287
		group250d	—	526	276
		group250e	1417	355	229
		Mean	4258	490	299
C	250	perm250a3	4894	2951	756
		perm250b3	2661	1566	425
		perm250c3	4800	3224	810
		perm250d3	3006	1977	365
		perm250e3	3397	2200	360
		Mean	3752	2384	543
C	250	perm250a5	—	3821	558
		perm250b5	2751	2112	413
		perm250c5	7926	2934	568
		perm250d5	7856	2630	596
		perm250e5	5665	4047	591
		Mean	6049	3108	545

*: Time to **prove** optimality only

Table 6: Breakdown of Crossover Times

Objects	% zeroes	Name	Total time	% Interior
150	0	r150a0	70	56.2
		r150b0	66	58.9
		r150c0	72	54.4
		r150d0	66	56.6
		r150e0	65	58.1
		Mean	68	56.9
200	0	r200a0	200	51.1
		r200b0	245	36.9
		r200c0	193	45.9
		r200d0	198	47.5
		r200e0	209	44.3
		Mean	208	45.1
250	0	r250a0	683	38.3
		r250b0	548	46.1
		r250c0	574	44.4
		r250d0	524	51.0
		r250e0	631	41.0
		Mean	592	44.2

Table 7: Breakdown of Crossover Times

Objects	% zeroes	Name	Total Time	% Interior
100	20	r100a2	185	89.2
		r100b2	426	37.0
		r100c2	214	83.8
		r100d2	132	72.5
		r100e2	192	86.6
		Mean	230	73.8
150	10	r150a1	218	29.5
		r150b1	165	39.2
		r150c1	221	29.9
		r150d1	237	27.7
		r150e1	200	32.0
		Mean	208	31.6
200	10	r200a1	1100	34.1
		r200b1	886	73.4
		r200c1	692	47.1
		r200d1	1666	35.8
		r200e1	836	52.3
		Mean	1036	48.6

Table 8: Breakdown of Crossover Times

Type	% zeroes	Name	Total Time	% Interior
B	100	group250a	383	30
		group250b	249	47
		group250c	287	41
		group250d	276	42
		group250e	229	52
		Mean	285	41
C	250	perm250a3	756	20
		perm250b3	425	38
		perm250c3	810	20
		perm250d3	365	55
		perm250e3	360	58
		Mean	543	32
C	250	perm250a5	558	42
		perm250b5	413	55
		perm250c5	568	39
		perm250d5	596	36
		perm250e5	591	40
		Mean	545	42