Branch and Bound Methods

- basic ideas and attributes
- unconstrained nonconvex optimization
- mixed convex-Boolean optimization

Methods for nonconvex optimization problems

- **convex optimization methods** are (roughly) always global, always fast
- for general nonconvex problems, we have to give up one
- **local optimization methods** are fast, but need not find global solution (and even when they do, cannot certify it)
- **global optimization methods** find global solution (and certify it), but are not always fast (indeed, are often slow)

Branch and bound algorithms

- methods for **global** optimization for nonconvex problems
- nonheuristic
  - maintain provable lower and upper bounds on global objective value
  - terminate with certificate proving ϵ-suboptimality
- often slow; exponential worst case performance
- but (with luck) can (sometimes) work well

Basic idea

- rely on two subroutines that (efficiently) compute a lower and an upper bound on the optimal value over a given region
  - upper bound can be found by choosing any point in the region, or by a local optimization method
  - lower bound can be found from convex relaxation, duality, Lipschitz or other bounds, . . .

- basic idea:
  - partition feasible set into convex sets, and find lower/upper bounds for each
  - form global lower and upper bounds; quit if close enough
  - else, refine partition and repeat

Unconstrained nonconvex minimization

goal: find global minimum of function $f : \mathbf{R}^m \to \mathbf{R}$, over an m-dimensional rectangle $\mathcal{Q}_{\mathrm{init}}$, to within some prescribed accuracy ϵ

- for any rectangle $\mathcal{Q} \subseteq \mathcal{Q}_{\mathrm{init}}$, we define $\Phi_{\min}(\mathcal{Q}) = \inf_{x \in \mathcal{Q}} f(x)$
- global optimal value is $f^\star = \Phi_{\min}(\mathcal{Q}_{\mathrm{init}})$

Lower and upper bound functions

- we'll use lower and upper bound functions $\Phi_{\rm lb}$ and $\Phi_{\rm ub}$, that satisfy, for any rectangle $\mathcal{Q} \subseteq \mathcal{Q}_{\rm init}$,

$$\Phi_{\rm lb}(\mathcal{Q}) \leq \Phi_{\rm min}(\mathcal{Q}) \leq \Phi_{\rm ub}(\mathcal{Q})$$

- bounds must become tight as rectangles shrink:

$$\forall \epsilon > 0 \; \exists \delta > 0 \; \forall \mathcal{Q} \subseteq \mathcal{Q}_{\rm init}, \quad \mathbf{size}(\mathcal{Q}) \leq \delta \Longrightarrow \Phi_{\rm ub}(\mathcal{Q}) - \Phi_{\rm lb}(\mathcal{Q}) \leq \epsilon$$

  where $\mathbf{size}(\mathcal{Q})$ is diameter (length of longest edge of $\mathcal{Q}$)

- to be practical, $\Phi_{\rm ub}(\mathcal{Q})$ and $\Phi_{\rm lb}(\mathcal{Q})$ should be cheap to compute

Branch and bound algorithm

1. compute lower and upper bounds on $f^\star$
   - set $L_1 = \Phi_{\rm lb}(\mathcal{Q}_{\rm init})$ and $U_1 = \Phi_{\rm ub}(\mathcal{Q}_{\rm init})$
   - terminate if $U_1 - L_1 \leq \epsilon$
2. partition (split) $\mathcal{Q}_{\rm init}$ into two rectangles $\mathcal{Q}_{\rm init} = \mathcal{Q}_1 \cup \mathcal{Q}_2$
3. compute $\Phi_{\rm lb}(\mathcal{Q}_i)$ and $\Phi_{\rm ub}(\mathcal{Q}_i)$, $i = 1, 2$
4. update lower and upper bounds on $f^\star$
   - update lower bound: $L_2 = \min\{\Phi_{\rm lb}(\mathcal{Q}_1), \Phi_{\rm lb}(\mathcal{Q}_2)\}$
   - update upper bound: $U_2 = \min\{\Phi_{\rm ub}(\mathcal{Q}_1), \Phi_{\rm ub}(\mathcal{Q}_2)\}$
   - terminate if $U_2 - L_2 \leq \epsilon$
5. refine partition, by splitting $\mathcal{Q}_1$ or $\mathcal{Q}_2$, and repeat steps 3 and 4

- can assume w.l.o.g. U_i is nonincreasing, L_i is nondecreasing

- at each step we have a partially developed binary tree; children correspond to the subrectangles formed by splitting the parent rectangle

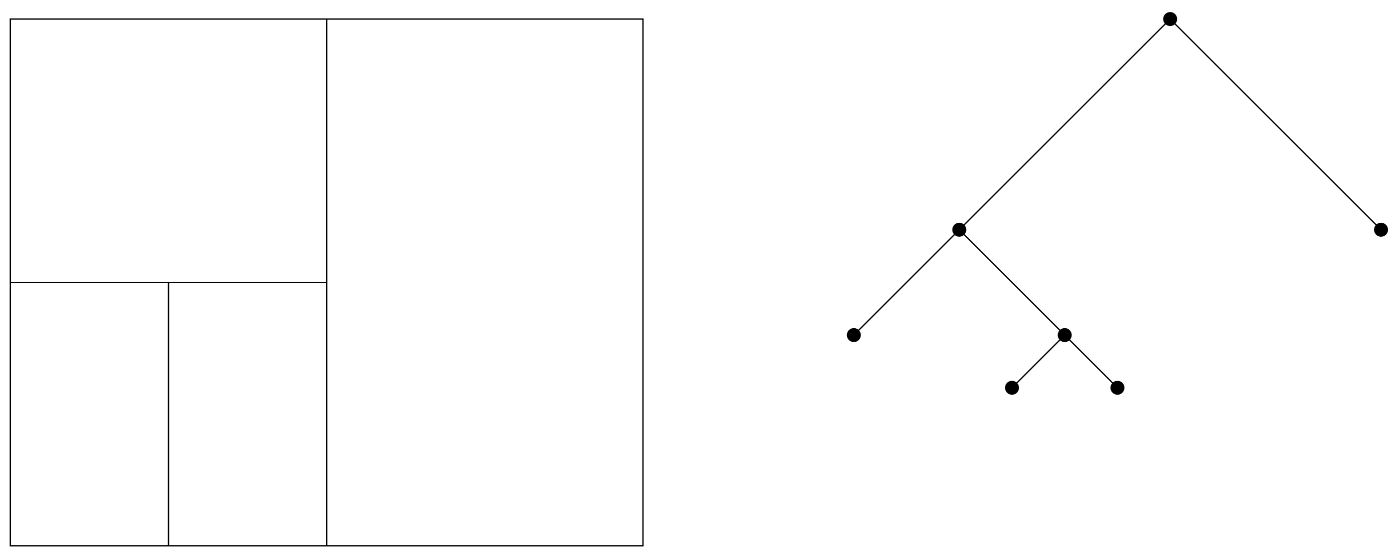
- leaves give the current partition of $\mathcal{Q}_{\rm init}$

- need rules for choosing, at each step
  - which rectangle to split
  - which edge (variable) to split
  - where to split (what value of variable)

- some good rules: split rectangle with smallest lower bound, along longest edge, in half

Example

partitioned rectangle in $\mathbf{R}^2$, and associated binary tree, after 3 iterations



Pruning

- can eliminate or **prune** any rectangle $\mathcal{Q}$ in tree with $\Phi_{\rm lb}(\mathcal{Q}) > U_k$
  - every point in rectangle is worse than current upper bound
  - hence cannot be optimal
- does not affect algorithm, but does reduce storage requirements
- can track progress of algorithm via
  - total pruned (or unpruned) volume
  - number of pruned (or unpruned) leaves in partition

Convergence analysis

- number of rectangles in partition $\mathcal{L}_k$ is k (without pruning)
- total volume of these rectangles is $\mathrm{vol}(\mathcal{Q}_{\mathrm{init}})$, so

$$\min_{\mathcal{Q}\in\mathcal{L}_k} \mathrm{vol}(\mathcal{Q}) \leq \frac{\mathrm{vol}(\mathcal{Q}_{\mathrm{init}})}{k}$$

- so for k large, at least one rectangle has small volume
- need to show that small volume implies small size
- this will imply that one rectangle has $U - L$ small
- hence $U_k - L_k$ is small

Bounding condition number

- **condition number** of rectangle $\mathcal{Q} = [l_1, u_1] \times \cdots \times [l_n, u_n]$ is

$$\mathrm{cond}(\mathcal{Q}) = \frac{\max_i (u_i - l_i)}{\min_i (u_i - l_i)}$$

- if we split rectangle along longest edge, we have

$$\mathrm{cond}(\mathcal{Q}) \leq \max\{\mathrm{cond}(\mathcal{Q}_{\mathrm{init}}), 2\}$$

  for any rectangle in partition

- other rules ($e.g.$, cycling over variables) also guarantee bound on $\mathrm{cond}(\mathcal{Q})$

Small volume implies small size

$$
\begin{aligned}
\mathrm{vol}(\mathcal{Q}) &= \prod_i (u_i - l_i) \geq \max_i (u_i - l_i) \left(\min_i (u_i - l_i) \right)^{m-1} \\
&= \frac{(2\,\mathsf{size}(\mathcal{Q}))^m}{\mathrm{cond}(\mathcal{Q})^{m-1}} \geq \left(\frac{2\,\mathsf{size}(\mathcal{Q})}{\mathrm{cond}(\mathcal{Q})} \right)^m
\end{aligned}
$$

and so size$(\mathcal{Q}) \leq (1/2)\mathrm{vol}(\mathcal{Q})^{1/m}\mathrm{cond}(\mathcal{Q})$

therefore if $\mathrm{cond}(\mathcal{Q})$ is bounded and $\mathrm{vol}(\mathcal{Q})$ is small, size$(\mathcal{Q})$ is small

Mixed Boolean-convex problem

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i=1,\ldots,m \\ & z_j \in \{0,1\}, \quad j=1,\ldots,n \end{array}$$

- $x \in \mathbf{R}^p$ is called *continuous variable*
- $z \in \{0,1\}^n$ is called *Boolean variable*
- $f_0,\ldots,f_n$ are convex in x and z
- optimal value denoted $p^\star$
- for each fixed $z \in \{0,1\}^n$, reduced problem (with variable x) is convex

Solution methods

- *brute force*: solve convex problem for each of the 2^n possible values of $z \in \{0,1\}^n$
  - possible for $n \leq 15$ or so, but not $n \geq 20$
- *branch and bound*
  - in worst case, we end up solving all 2^n convex problems
  - hope that branch and bound will actually work much better

Lower bound via convex relaxation

convex relaxation

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i=1,\ldots,m \\ & 0 \leq z_j \leq 1, \quad j=1,\ldots,n \end{array}$$

- convex with (continuous) variables x and z, so easily solved
- optimal value (denoted L_1) is lower bound on $p^\star$, optimal value of original problem
- L_1 can be $+\infty$ (which implies original problem infeasible)

Upper bounds

- can find an upper bound (denoted U_1) on $p^\star$ several ways
- simplest method: round each relaxed Boolean variable $z_i^\star$ to 0 or 1
- more sophisticated method: round each Boolean variable, then solve the resulting convex problem in x
- randomized method:
  - generate random $z_i \in \{0, 1\}$, with $\mathbf{Prob}(z_i = 1) = z_i^\star$
  - (optionally, solve for x again)
  - take best result out of some number of samples
- upper bound can be $+\infty$ (method failed to produce a feasible point)
- if $U_1 - L_1 \leq \epsilon$ we can quit

Branching

- pick any index k, and form two subproblems

- first problem:

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i=1,\ldots,m \\ & z_j \in \{0,1\}, \quad j=1,\ldots,n \\ & z_k = 0 \end{array}$$

- second problem:

$$\begin{array}{ll} \text{minimize} & f_0(x,z) \\ \text{subject to} & f_i(x,z) \leq 0, \quad i=1,\ldots,m \\ & z_j \in \{0,1\}, \quad j=1,\ldots,n \\ & z_k = 1 \end{array}$$

- each of these is a Boolean-convex problem, with $n - 1$ Boolean variables
- optimal value of original problem is the smaller of the optimal values of the two subproblems
- can solve convex relaxations of subproblems to obtain lower and upper bounds on optimal values

New bounds from subproblems

- let $\tilde{L}, \tilde{U}$ be lower, upper bounds for $z_k = 0$
- let $\bar{L}, \bar{U}$ be lower, upper bounds for $z_k = 1$
- $\min\{\tilde{L}, \bar{L}\} \geq L_1$
- can assume w.l.o.g. that $\min\{\tilde{U}, \bar{U}\} \leq U_1$
- thus, we have new bounds on $p^\star$:

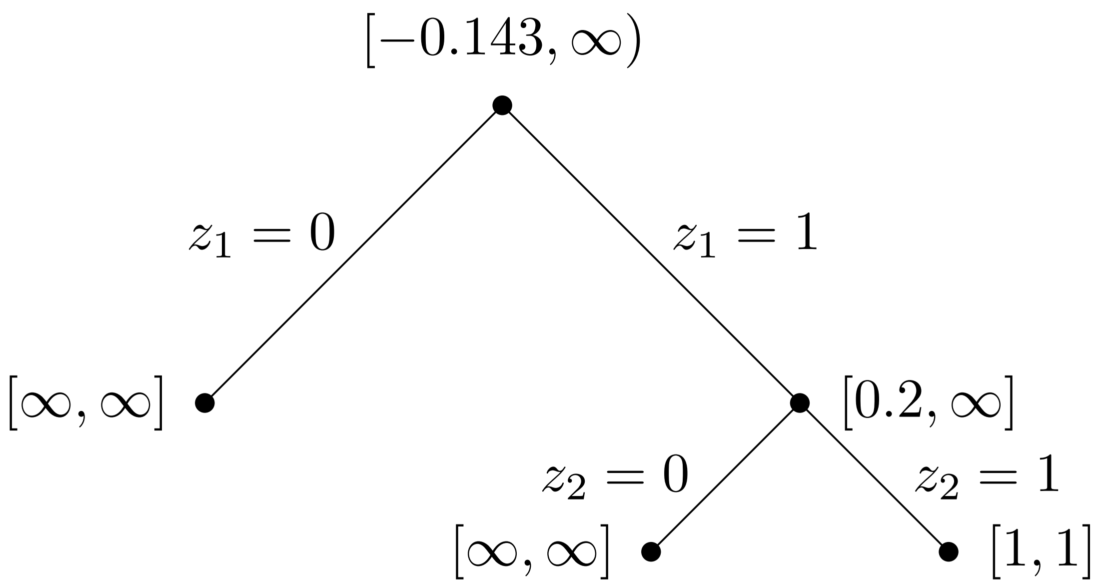
$$L_2 = \min\{\tilde{L}, \bar{L}\} \leq p^\star \leq U_2 = \min\{\tilde{U}, \bar{U}\}$$

Branch and bound algorithm

- continue to form binary tree by splitting, relaxing, calculating bounds on subproblems
- convergence proof is trivial: cannot go more than 2^n steps before $U = L$
- can prune nodes with L excceding current U_k
- common strategy is to pick a node with smallest L
- can pick variable to split several ways
  - 'least ambivalent': choose k for which $z^\star = 0$ or 1, with largest Lagrange multiplier
  - 'most ambivalent': choose k for which $|z_k^\star - 1/2|$ is minimum

Small example

nodes show lower and upper bounds for three-variable Boolean LP



Minimum cardinality example

find sparsest x satisfying linear inequalities

$$\begin{array}{ll} \text{minimize} & \mathbf{card}(x) \\ \text{subject to} & Ax \preceq b \end{array}$$

equivalent to mixed Boolean-LP:

$$\begin{array}{ll} \text{minimize} & \mathbf{1}^T z \\ \text{subject to} & L_i z_i \leq x_i \leq U_i z_i, \quad i = 1, \ldots, n \\ & Ax \preceq b \\ & z_i \in \{0, 1\}, \quad i = 1, \ldots, n \end{array}$$

with variables x and z and lower and upper bounds on x, L and U

Bounding x

- L_i is optimal value of LP

$$\begin{array}{ll} \text{minimize} & x_i \\ \text{subject to} & Ax \preceq b \end{array}$$

- U_i is optimal value of LP

$$\begin{array}{ll} \text{maximize} & x_i \\ \text{subject to} & Ax \preceq b \end{array}$$

- solve $2n$ LPs to get all bounds

- if $L_i > 0$ or $U_i < 0$, we can just set $z_i = 1$

Relaxation problem

- relaxed problem is

$$
\begin{array}{ll}
\text{minimize} & \mathbf{1}^T z \\
\text{subject to} & L_i z_i \leq x_i \leq U_i z_i, \quad i = 1, \ldots, n \\
& Ax \preceq b \\
& 0 \leq z_i \leq 1, \quad i = 1, \ldots, n
\end{array}
$$

- (assuming $L_i < 0$, $U_i > 0$) equivalent to

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^n \left((1/U_i)(x_i)_+ + (-1/L_i)(x_i)_-\right) \\
\text{subject to} & Ax \preceq b
\end{array}
$$

- objective is asymmetric weighted ℓ_1-norm

A few more details

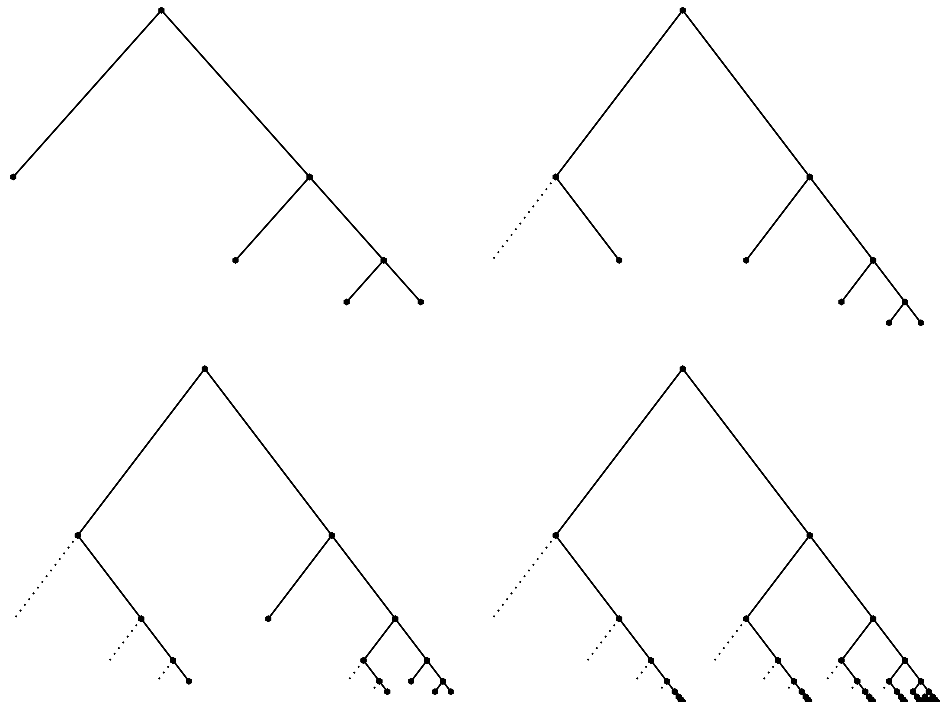
- relaxed problem is well known heuristic for finding a sparse solution, so we take $\mathbf{card}(x^\star)$ as our upper bound
- for lower bound, we can replace L from LP with $\lceil L \rceil$, since $\mathbf{card}(x)$ is integer valued
- at each iteration, split node with lowest lower bound
- split most ambivalent variable

Small example

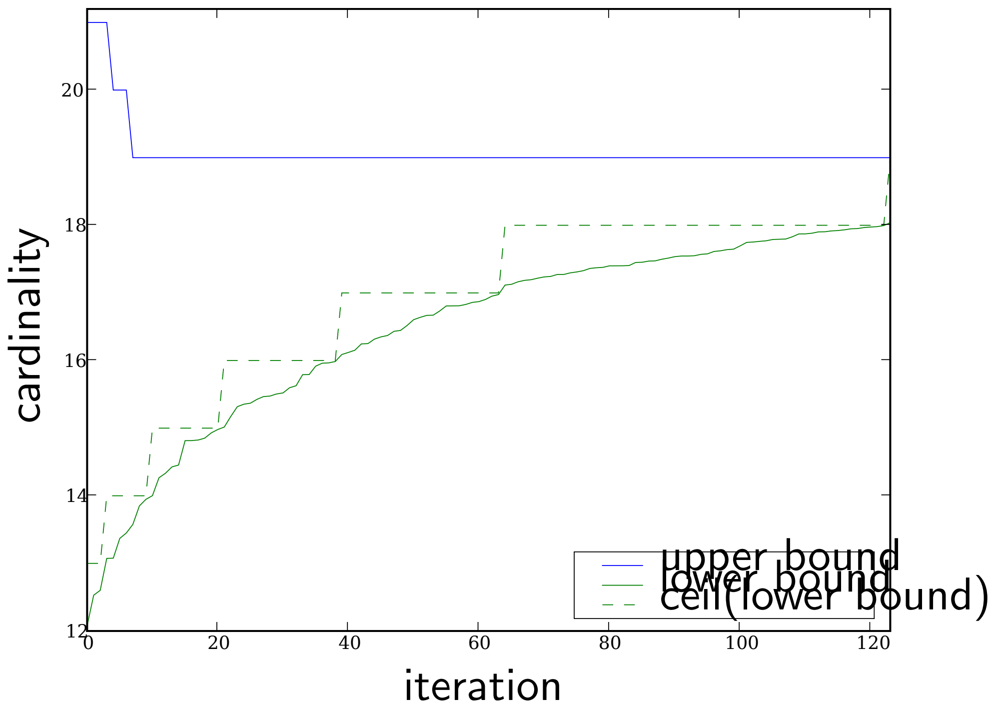
- random problem with 30 variables, 100 constraints
- $2^{30} \approx 10^9$
- takes 8 iterations to find a point with globally minimum cardinality (19)
- but, takes 124 iterations to **prove** minimum cardinality is 19
- requires 309 LP solves (including 60 to calculate lower and upper bounds on each variable)

Algorithm progress

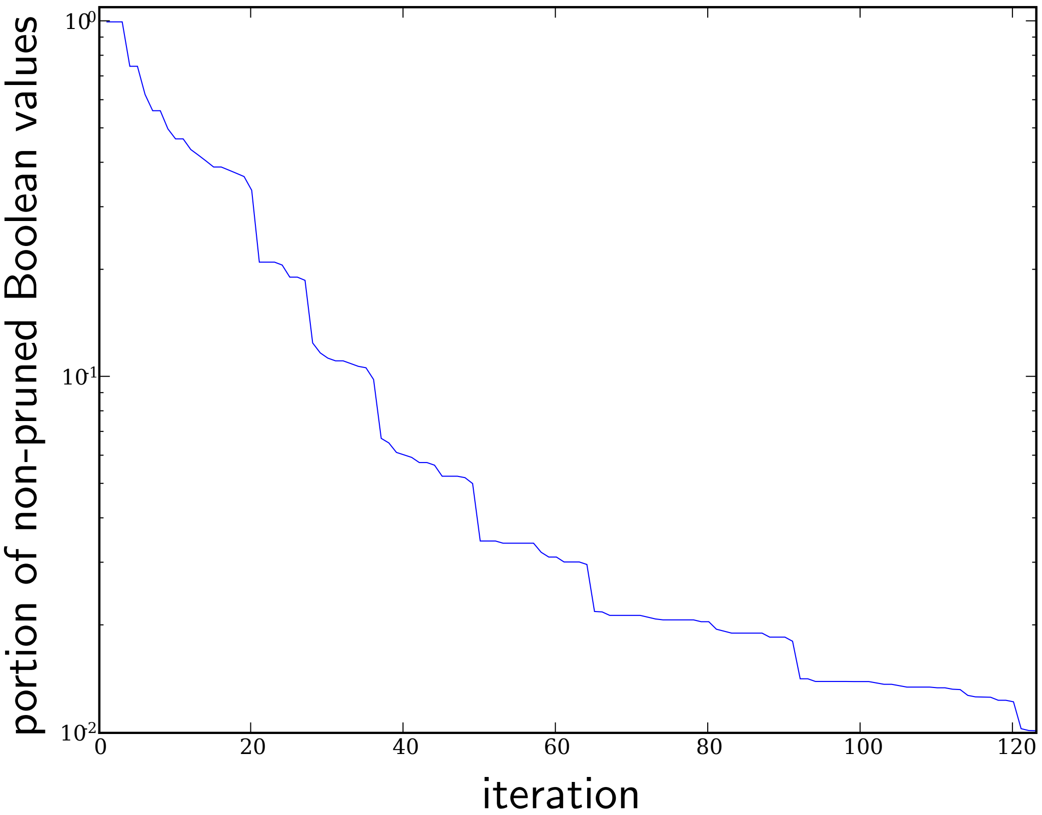
tree after 3 iterations (top left), 5 iterations (top right), 10 iterations (bottom left), and 124 iterations (bottom right)



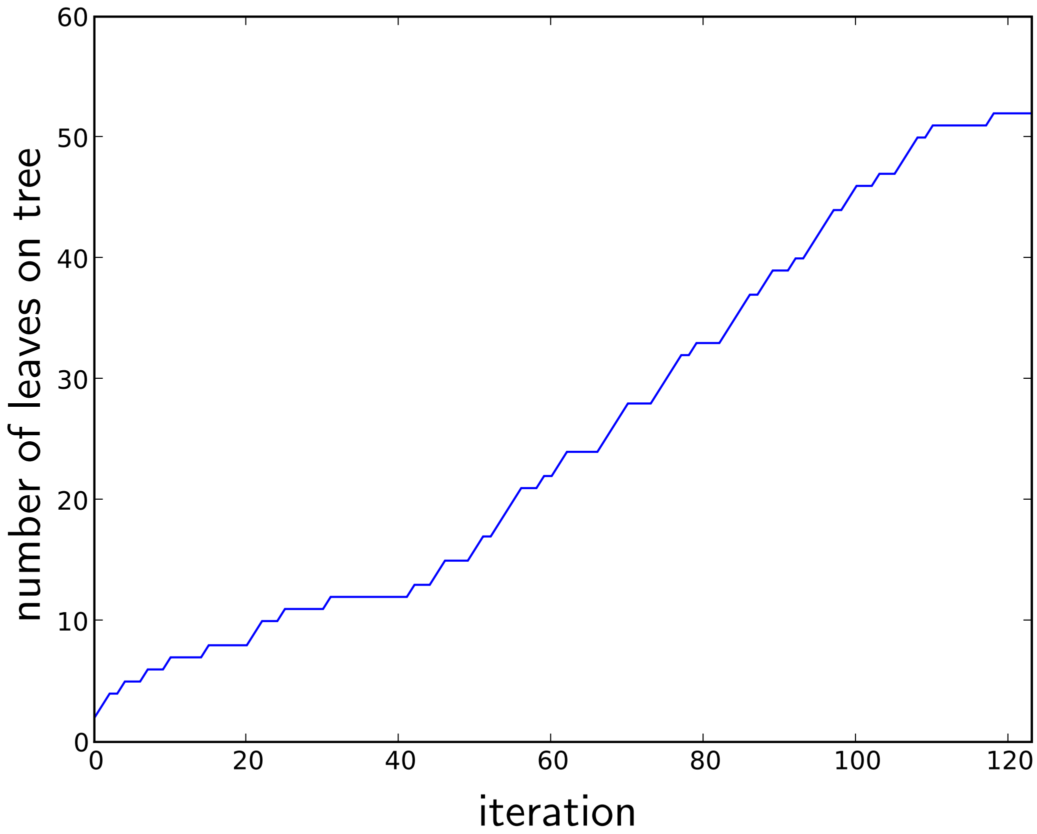
Global lower and upper bounds



Portion of non-pruned sparsity patterns



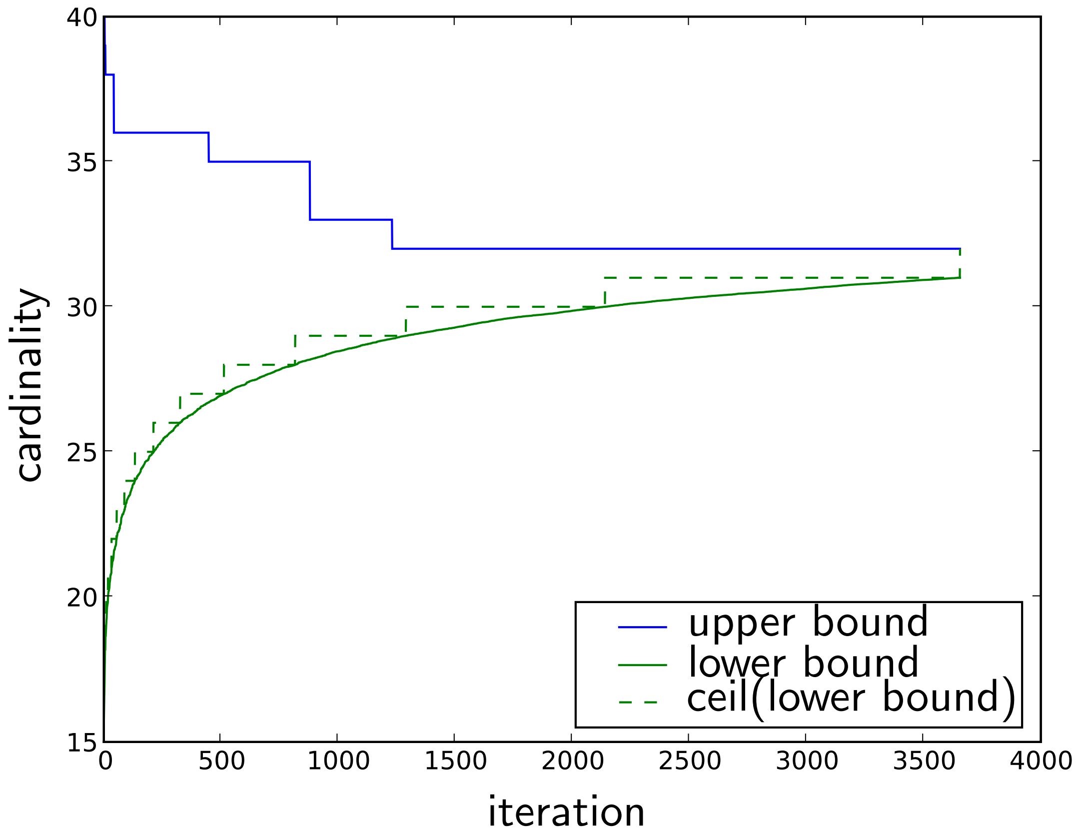
Number of active leaves in tree



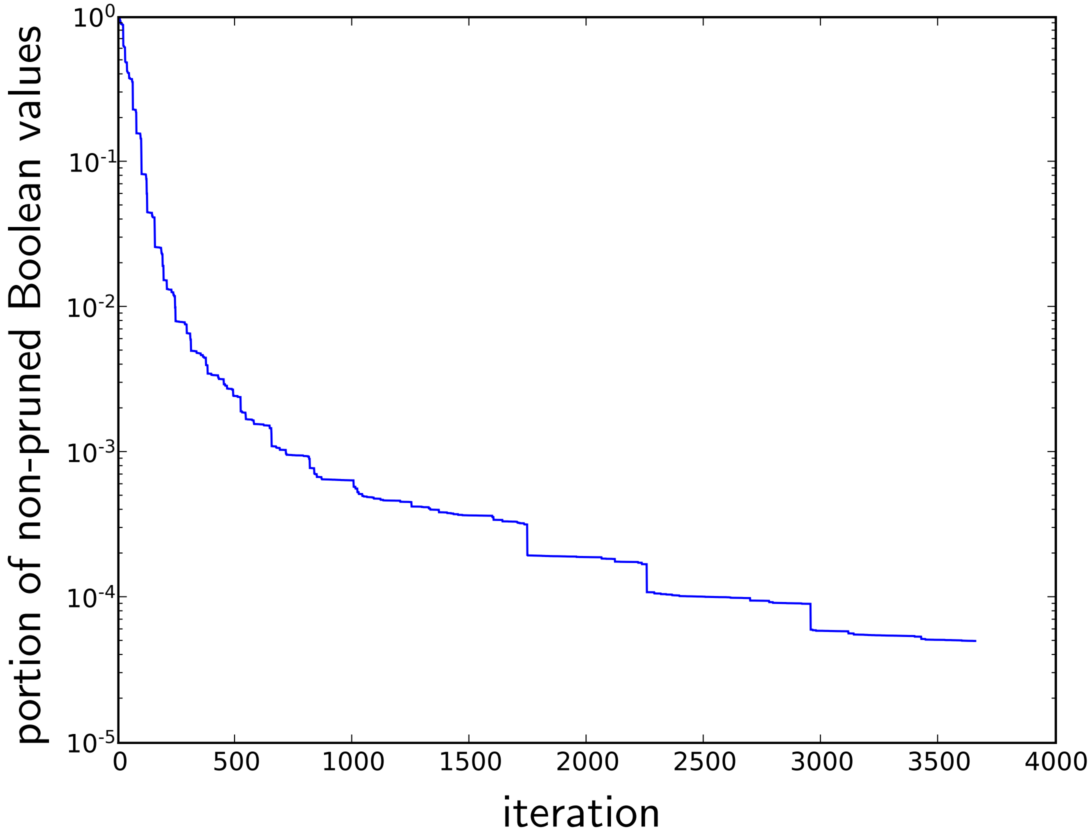
Larger example

- random problem with 50 variables, 100 constraints
- $2^{50} \approx 10^{15}$
- took 3665 iterations (1300 to find an optimal point)
- minimum cardinality 31
- same example as used in ℓ_1-norm methods lecture
  - basic ℓ_1-norm relaxation (1 LP) gives x with $\mathbf{card}(x) = 44$
  - iterated weighted ℓ_1-norm heuristic (4 LPs) gives x with $\mathbf{card}(x) = 36$

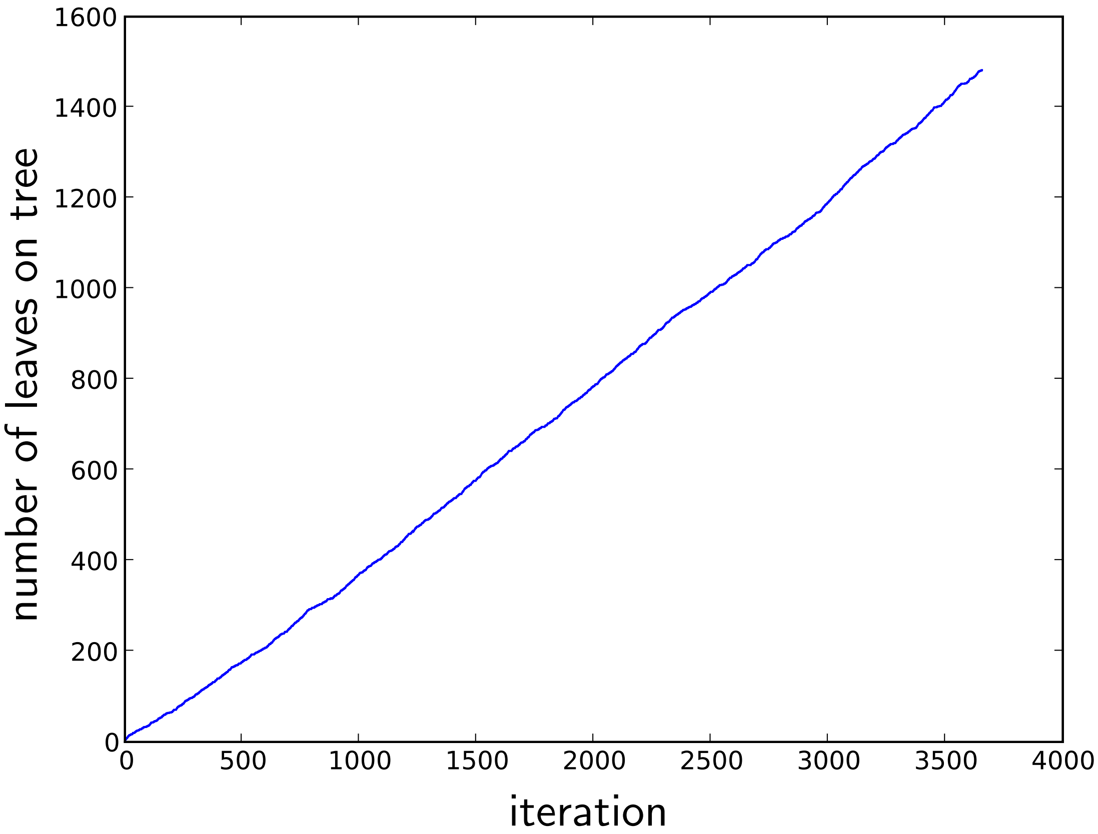
Global lower and upper bounds



Portion of non-pruned sparsity patterns



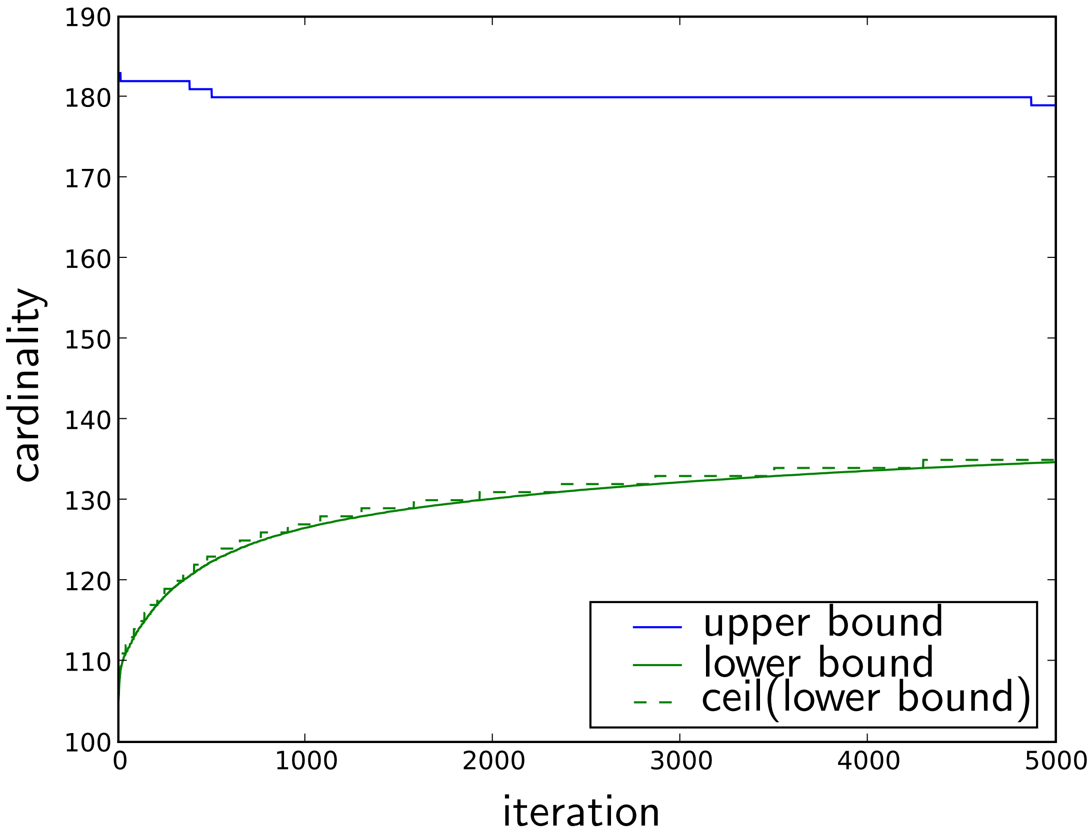
Number of active leaves in tree



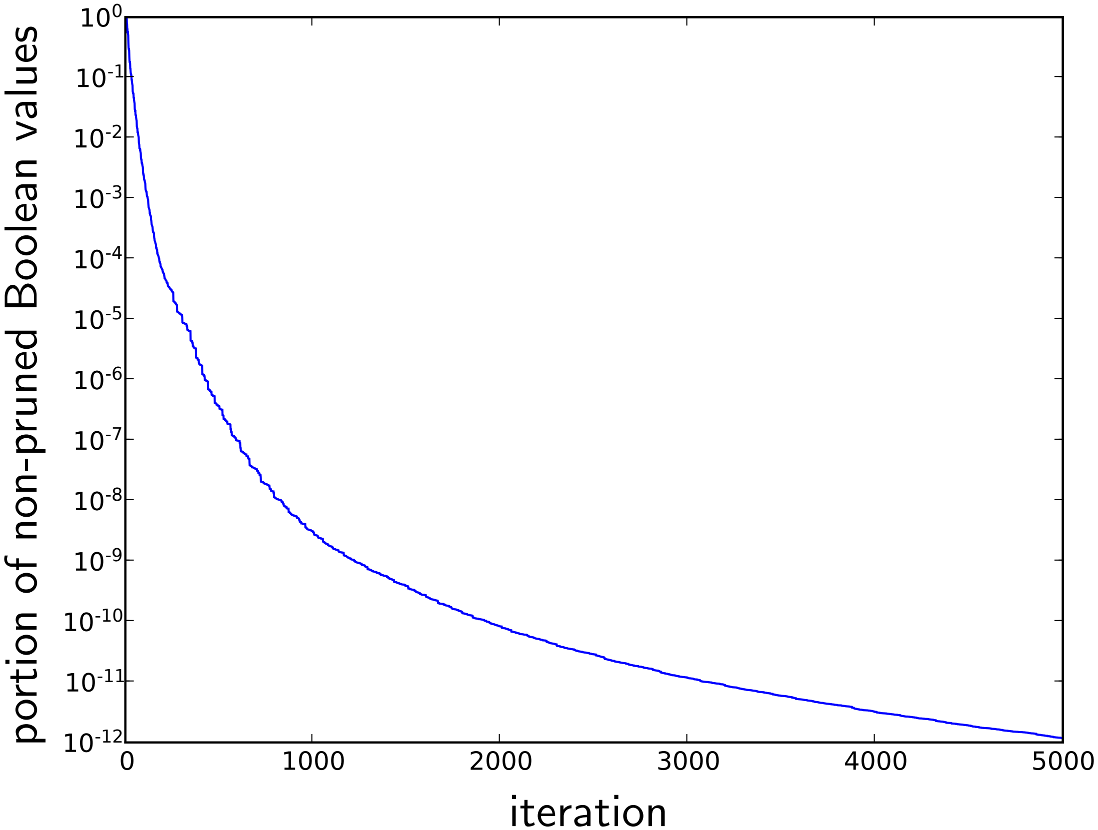
Even larger example

- random problem with 200 variables, 400 constraints
- $2^{200} \approx 1.6 \cdot 10^{60}$
- we quit after 10000 iterations (50 hours on a single processor machine with 1 GB of RAM)
- only know that optimal cardinality is between 135 and 179
- but have reduced number of possible sparsity patterns by factor of 10^{12}

Global lower and upper bounds



Portion of non-pruned sparsity patterns



Number of active leaves in tree