Discrete Optimization

Rahul Mazumder
Columbia Statistics

Computational Statistics

March 27, 2014

- convex optimization methods are (roughly) always global, always fast
- non-convex optimization is typically much harder — one needs a compromise
- Use local optimization methods — fast but not global, no certificate of global optimality
  (few exceptions)
- Global Optimization methods — get global solutions and certify it. Often obtaining a global solution is fast. But certifying that it is a global solution takes time...

Branch and Bound

- A non heuristic method for finding global solutions for non-convex problems.
- Basic idea is "divide and conquer"
- The original problem is prohibitively large, so we divide it into smaller sub-problems, which are "solved" (conquered) — this is repeated till all sub-problems have been "conquered"
- Dividing:
  Partition the feasible solutions into smaller and smaller subsets
- Conquering:
  — provide a bound for the best solution in the subset
  — discard the subset if you can prove that the subset cannot obtain an optimal solution
- Let us consider a general framework...

Unconstrained non-convex minimization

Task Find global minimum of a function $f : \Re^m \mapsto \Re$ over a m-dimensional rectangle Q_{IN} to some prescribed accuracy.

- For any rectangle $Q \subset Q_{IN}$ we define $\Phi_{\min}(Q) = \inf_{x \in Q} f(x)$
- global optimum value is $f^* = \Phi_{\min}(Q_{IN})$

Lower and Upper bound functions

- We will use lower and upper bound functions Φ_{LB} and Φ_{UB} that satisfy for any rectangle, $Q \subset Q_{IN}$

$$\Phi_{LB}(Q) \leq \Phi_{\min}(Q) \leq \Phi_{UB}(Q)$$

- bounds must become tight as rectangles shrink, i,e.,

$$\forall \epsilon > 0 \exists \delta > 0 \forall Q \subset Q_{IN}, \text{size}(Q) \leq \delta \implies \Phi_{UB}(Q) - \Phi_{LB}(Q) \leq \epsilon$$

  where, size(Q) is the diameter (length of the longest edge of Q)

- to be practical, $\Phi_{UB}(Q), \Phi_{LB}(Q)$ should be easy to compute.

Branch and bound algorithm

1. compute lower and upper bounds on f^*
   — set $L_1 = \Phi_{lb}(Q_{IN})$ and $U_1 = \Phi_{UB}(Q_{IN})$
   — terminate if $U_1 - L_1 \leq \epsilon$
2. partition (split) Q_{IN} into two rectangles $Q_{IN} = Q_1 \cup Q_2$
3. compute $\Phi_{UB}(Q_i)$ and $\Phi_{LB}(Q_i)$
4. update lower and upper bounds on f^*
   — update lower bound as $L_2 = \min\{\Phi_{LB}(Q_1), \Phi_{LB}(Q_2)\}$
   — update upper bound as $U_2 = \min\{\Phi_{UB}(Q_1), \Phi_{UB}(Q_2)\}$
   — terminate if $U_2 - L_2 \leq \epsilon$
5. refine partition by splitting Q_1 or Q_2 and repeat steps 3,4
   Note: At stage k we have:
   $L_k = \min_{i=1}^{k} \Phi_{LB}(Q_i) \qquad U_k = \min_{i=1}^{k} \Phi_{UB}(Q_i)$

Convergence analysis of branch and bound

- number of rectangles in partition L_k is k (without pruning)
- total volume of these rectangles is $vol(Q_{IN})$ so:

$$\min_{Q \in L_k} Vol(Q) \leq \frac{Vol(Q_{IN})}{k}$$

- 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

Convergence analysis of branch and bound

- condition number of rectangle $Q = [l_1, u_1] \times \ldots \times [l_n, u_n]$ is:

$$cond(Q) = \frac{\max_i(u_i - l_1)}{\min_i(u_i - l_i)}$$

- if we split rectangle along longest edge, we have

$$cond(Q) \leq \max\{cond(Q_{IN}), 2\}$$

- If Q is rectangle. then:

$$Vol(Q) \geq \left(\frac{2\text{size}(Q)}{cond(Q)}\right)^m$$

and so, size $(Q) \leq \frac{1}{2} Vol(Q)^{\frac{1}{m}} cond(Q)$
therefore if $cond(Q)$ is bounded and $vol(Q)$ is small then $size(Q)$ is small.

Example of branch and bound

- Let us consider a simple example, illustrating a branch and bound algorithm in action.
- We will consider a Linear program where some of the variables are continuous and some are integers (MILP)
- We will see that MILPs have very nice structures...

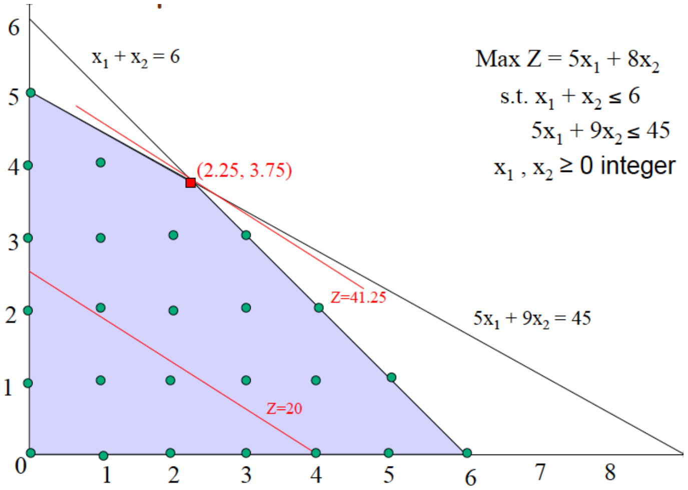
Example

Consider the following problem:

$$\begin{aligned} \text{maximize} \quad & 5x_1 + 8x_2 \\ \text{subject to} \quad & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \in \mathbb{Z}_+ \end{aligned}$$

where, optimization variables are x_1, x_2 and $\mathbb{Z}_+$ denotes the set of non-negative integers.

Example



Max Z = 5x1 + 8x2
s.t. x1 + x2 ≤ 6
5x1 + 9x2 ≤ 45
x1 , x2 ≥ 0 integer
x1 + x2 = 6
5x1 + 9x2 = 45
(2.25, 3.75)
Z=41.25
Z=20
0
1
2
3
4
5
6
7
8

Example

- If the LP relaxation has integral optimal solution x^*, then we are done
- In this case, $(x_1, x_2) = (2.25, 3.75)$ is the opt. soln for the LP relaxation — not integral.
- The opt. value of the relaxation is 41.25
- The opt. value of the LP relaxation is an upper bound for the opt. value of the integer program. Thus 41.25 is an upper bound

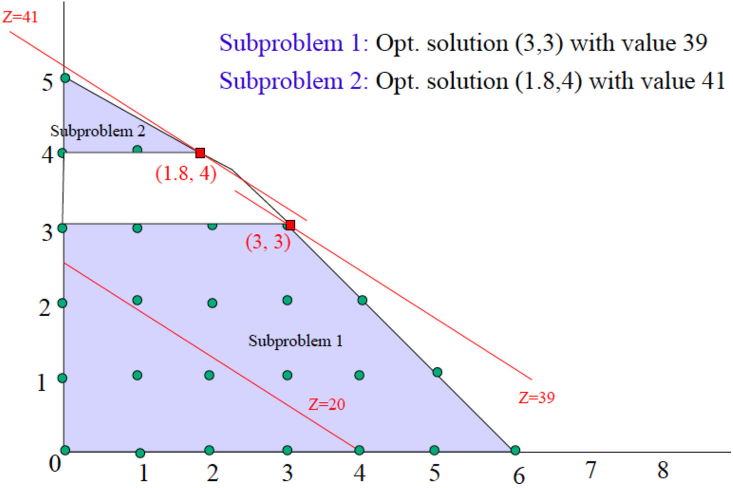
Example

- We will now branch (partition the feasible space), in an attempt to refine the solution.
- Choose a variable that is fractional in the optimal solution to the LP-relaxation say, x_2 . We must have either $x_2 \leq 3$ or $x_2 \geq 4$.
- Branch on x_2 to create two new subproblems:

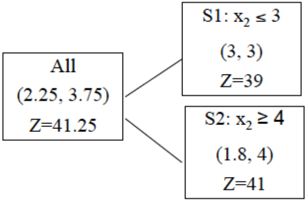
$$\underbrace{\begin{pmatrix} \text{maximize} & 5x_1 + 8x_2 \\ \text{subject to} & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \\ & x_2 \leq 3 \end{pmatrix}}_{\text{Subprob 1}} \qquad \underbrace{\begin{pmatrix} \text{maximize} & 5x_1 + 8x_2 \\ \text{subject to} & x_1 + x_2 \leq 6 \\ & 5x_1 + 9x_2 \leq 45 \\ & x_1, x_2 \geq 0 \\ & x_2 \geq 4 \end{pmatrix}}_{\text{Subprob 2}}$$

- Solve both the problems

Example

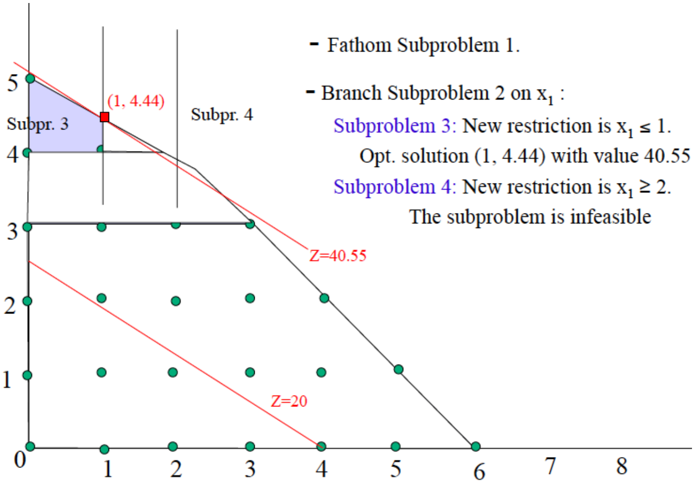


Z=41
Subproblem 1: Opt. solution (3,3) with value 39
Subproblem 2: Opt. solution (1.8,4) with value 41
5
Subproblem 2
4
(1.8, 4)
3
(3, 3)
2
Subproblem 1
1
Z=20
Z=39
0
1
2
3
4
5
6
7
8

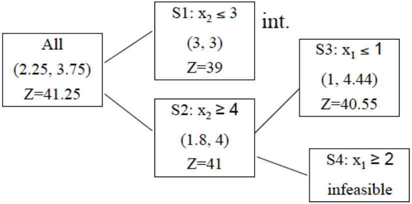


- The opt. for subproblem 1 is integral : $(3, 3)$
- If further branching on a subproblem will yield no useful information we say it is *fathomed* (We fathom subproblem 1)
- The best integer soln found so far is "incumbent", its value denoted by Z^*.

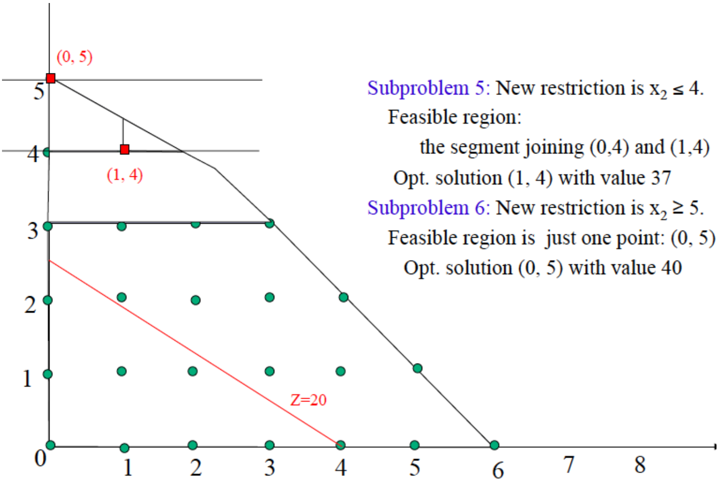
  [Here the incumbent is (3,3) and $Z^* = 39$]
- Z^* is a lower bound for the IP.



- Fathom Subproblem 1.
- Branch Subproblem 2 on x_1 :
  - Subproblem 3: New restriction is $x_1 \leq 1$.
    Opt. solution (1, 4.44) with value 40.55
  - Subproblem 4: New restriction is $x_1 \geq 2$.
    The subproblem is infeasible



- If a subproblem is infeasible then it is fathomed. Here, subproblem 4 is fathomed.
- The upper bound for the problem is updated:
  $39 \leq \text{OPT} \leq 40.55$
- Next branch on subproblem 3 on x_2



Subproblem 5: New restriction is $x_2 \leq 4$.

Feasible region:

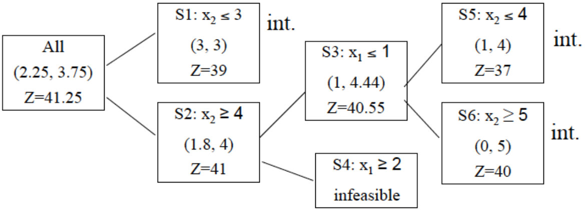
the segment joining (0,4) and (1,4)

Opt. solution (1, 4) with value 37

Subproblem 6: New restriction is $x_2 \geq 5$.

Feasible region is just one point: (0, 5)

Opt. solution (0, 5) with value 40



- If the optimal value of a subproblem is smaller than Z^* then it is fathomed. [here subproblem 5 is fathomed]
- If a subproblem has integral opt. solution x^* and its value is $> Z^*$ then x^* replaces the current incumbent.

  [Subproblem 5 has integral optimal solution, and its value $40 > 39 = Z^*$. Thus, (0,5) is the new incumbent, and new $Z^* = 40$.]
- If there are no unfathomed subproblems left then the current incumbent is an optimal solution for the IP.

  [In our case, (0, 5) is an optimal solution with optimal value 40.]

Branch and Bound and Beyond

- Note that the ordering of the branching is important and can influence algorithm run-time
- Worst case the algorithm can be NP, do complete (or near complete) enumeration
- MILP solvers/algorithms rely on combinations of branch and bound, branch and cuts, cutting plane methods, rounding and very clever mix of heuristics...for better performance in theory and practice.

Modern Integer Programming

- **Pessimistic Viewpoint:**
  — Integer programming is NP hard.
  — Dismiss problems as prohibitively expensive

Modern Integer Programming

- **Pessimistic Viewpoint:**
  — Integer programming is NP hard.
  — Dismiss problems as prohibitively expensive
  This is not quite true
- Thanks to powerful computers, skillful software engineering, tremendous developments in algorithms — a large class of these problems can be **solved** to global (or near global) accuracy within a very reasonable time frame.
- **Realistic Viewpoint:**
  A large class of integer programs (for example: Mixed Integer Linear Optimization) are tractable, i.e., they solve practical sized problems to provable optimality within a very reasonable time frame...

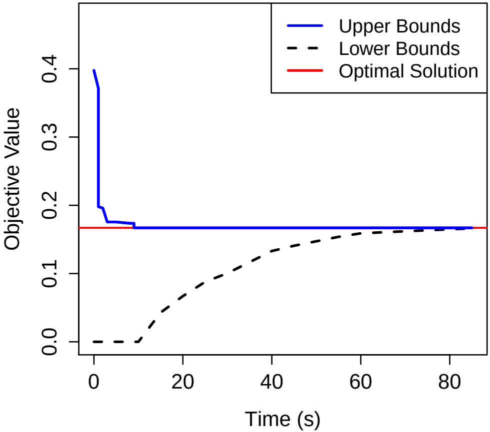
What is Mixed Integer Linear Optimization (MIO) ?

The generic MIO framework concerns the following optimization problem:

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}'\boldsymbol{\alpha} + \mathbf{d}'\boldsymbol{\theta} \\ & A\boldsymbol{\alpha} + B\boldsymbol{\theta} \geq \mathbf{b} \\ & \boldsymbol{\alpha} \in \Re^n_+ \\ & \boldsymbol{\theta} \in \{0,1\}^m, \end{aligned} \tag{1}$$

where, $\mathbf{c} \in \Re^n, \mathbf{d} \in \Re^m, A \in \Re^{k\times n}, B \in \Re^{k\times m}, \mathbf{b} \in \Re^k$ are the given parameters of the problem; we optimize over both continuous ($\boldsymbol{\alpha}$) and discrete ($\boldsymbol{\theta}$) variables.

Typical Evolution of MIO



Modern solvers for MIO use combinations of branch and bound , branch and cuts, cutting plane methods, rounding and very clever mix of heuristics...

Statistical Applications

Application 1: Clustered Regression

Bertsimas and Shioda 2007

Application 1

- Consider the problem of simultaneously fitting K linear regression functions to a set of N data points $(y_i, \mathbf{x}_i), i = 1, \ldots, N$ with $\mathbf{x}_i \in \Re^p$.
- In Clustered Regression, we want the linear regression lines to be:

$$y_i = \mathbf{x}_i' \boldsymbol{\beta}_{k_i} + \epsilon_i, i = 1, \ldots, N$$

  and $k_1, k_2, \ldots, k_N$ take K different values, i.e., $|\{k_1, k_2, \ldots, k_N\}| = K$.
- Question: How does one do this ?

Application 1

- Consider the problem of simultaneously fitting K linear regression functions to a set of N data points $(y_i, \mathbf{x}_i), i = 1, \ldots, N$ with $\mathbf{x}_i \in \Re^p$.
- In Clustered Regression, we want the linear regression lines to be:

$$y_i = \mathbf{x}_i' \boldsymbol{\beta}_{k_i} + \epsilon_i, i = 1, \ldots, N$$

  and $k_1, k_2, \ldots, k_N$ take K different values, i.e., $|\{k_1, k_2, \ldots, k_N\}| = K$.
- Question: How does one do this ?
- Task:

$$\text{minimize} \quad \sum_{i=1}^{N} |y_i - \mathbf{x}_i' \boldsymbol{\beta}_{k_i}|$$

  subject to $|\{k_1, k_2, \ldots, k_N\}| = K$.

Application 1

The problem can be cast as a MIO.
Let the groups be $\overline{K} := \{1, 2, \ldots, K\}$.
Denote binary variables $a_{k,i}$ such that:

$$a_{k,i} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ is in group } k \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The mixed integer linear optimization problem for Clustered Regression is:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \delta_i \\ \text{subject to} \quad & \delta_i \geq (y_i - \mathbf{x}_i' \boldsymbol{\beta}_k) - M(1 - a_{k,i}), k \in \overline{K}; i \in 1, .., N \\ & \delta_i \geq -(y_i - \mathbf{x}_i' \boldsymbol{\beta}_k) - M(1 - a_{k,i}), k \in \overline{K}; i \in 1, .., N \\ & \sum_{k} a_{k,i} = 1, i \in N \\ & a_{k,i} \in \{0, 1\}, \delta_i \geq 0. \end{aligned}$$

Application 2: Least Quantile of Squares

Bertsimas and Mazumder 2013

Review of Robust Statistics

Usual linear model

$$\mathbf{y}_{n\times 1} = \mathbf{X}_{n\times p}\boldsymbol{\beta}_{p\times 1} + \boldsymbol{\epsilon}_{n\times 1}$$

Assume $\mathbf{X}$ contains a column of ones.

- Least Squares (LS) estimator

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LS})} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} r_i^2$$

  where $r_i = y_i - \mathbf{x}_i'\boldsymbol{\beta}$ for $i = 1, \ldots, n$

- The LS estimator is adversely affected by a single outlier and has a limiting Breakdown point of 0 (Dohono & Huber '83; Hampel '75). ($n \to \infty$, and p fixed)

Review of Robust Statistics

- The Least Absolute Deviation (LAD) estimator:

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LAD})} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} |r_i|,$$

  is not good either.

  LAD has 0 breakdown point.

- M-Estimators (Huber 1973), came to a partial rescue by minimizing

$$\sum_{i=1}^{n} \rho(r_i),$$

  where, $\rho(r)$ is a symmetric function, with min. at zero; by slightly improving the breakdown point.

Least Median of Squares

- Rousseew, 1984 introduced the Least Median of Squares (LMS)

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LMS})} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \left(\underset{i=1,\ldots,n}{\operatorname{median}} \; |r_i| \right). \tag{3}$$

- LMS has a finite sample breakdown point of almost 50 %.
- Historically, first equivariant estimator with highest possible breakdown point.
- Theoretical properties are well understood (?)
- *Robust methods have important applications — computer vision, chemometrics, health-care, others...*

Least Quantile of Squares

- More generally, Least Quantile of Squares (LQS) estimator:

$$\widehat{\boldsymbol{\beta}}^{(\mathrm{LQS})} \in \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; |r_{(q)}|, \tag{4}$$

where, $r_{(q)}$ is the qth ordered absolute residual:

$$|r_{(1)}| \leq |r_{(2)}| \leq \ldots \leq |r_{(n)}|. \tag{5}$$

LMS Computation : State of the art

- LMS problem is NP hard (Bernholt '05).
- **Exact Algorithms**
  - Enumeration based, branch and bound, theory CS algorithms with $O(n^p)$.
  - Clever *Exact* algorithms scale upto $n = 50, p = 5$.
- **Approximate Algorithms**
  - Based on heuristic subsampling / local searches.
  - Scale better, but no guarantees
- Almost all methods use *special* geometric properties of the LMS solution.
  Do not generalize to account for shrinkage in β (say).

What we do

Solve the following problem:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad |r_{(q)}|,$$

where, $r_i = y_i - \mathbf{x}_i' \boldsymbol{\beta}$, q is a quantile.

More generally, framework can solve:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad |r_{(q)}|, \;\; \text{subject to} \;\; \mathbf{A}\boldsymbol{\beta} \leq \mathbf{b} \;\; (\text{and/or} \;\; \|\boldsymbol{\beta}\|_2^2 \leq \delta)$$

Overview of our approach

- Write the LMS problem as a Mixed Integer Optimization (MIO) problem.
- Use techniques in MIO to do **global** optimization.

Formulation

Consider the ordered residuals, and assume n odd

$$|r_{(1)}| \leq |r_{(2)}| \leq \ldots \leq |r_{(n)}|.$$

- We need to write $r_{(q)}$, where $q = (n+1)/2$ as a MIO.

Formulation

Step 1: Introduce binary variables $z_i, i = 1, \ldots, n$ such that:

$$z_i = \begin{cases} 1, & \text{if } |r_i| \leq |r_{(q)}|, \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

Step 2: Use auxiliary continuous variables $\mu_i, \overline{\mu}_i \geq 0$ such that:

$$|r_i| - \mu_i \leq |r_{(q)}| \leq |r_i| + \overline{\mu}_i, i = 1, \ldots, n, \tag{7}$$

with the conditions:

$$\begin{aligned} &\text{If} \quad |r_i| \geq |r_{(q)}|, \quad \text{then} \quad \overline{\mu}_i = 0, \mu_i \geq 0, \\ \text{and} \quad &\text{if} \quad |r_i| \leq |r_{(q)}|, \quad \text{then} \quad \mu_i = 0, \overline{\mu}_i \geq 0. \end{aligned} \tag{8}$$

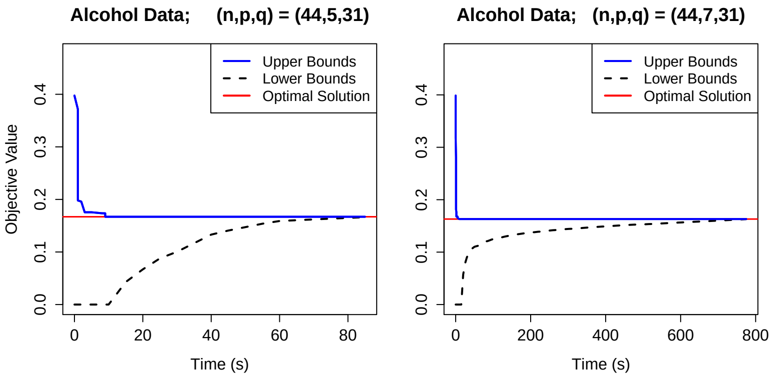
Formulation

The following MIO formulation:

$$
\begin{aligned}
\text{minimize} \quad & \gamma & & \\
\text{subject to} \quad & |r_i| + \overline{\mu}_i \geq \gamma, & & i = 1 \ldots, n \\
& \gamma \geq |r_i| - \mu_i, & & i = 1 \ldots, n \\
& M_u z_i \geq \overline{\mu}_i, & & i = 1, \ldots, n \\
& M_\ell (1 - z_i) \geq \mu_i, & & i = 1, \ldots, n \\
& \sum_{i=1}^{n} z_i = q & & \\
& \mu_i \geq 0, & & i = 1, \ldots, n \\
& \overline{\mu}_i \geq 0, & & i = 1, \ldots, n \\
& z_i \in \{0, 1\}, & & i = 1, \ldots, n,
\end{aligned} \tag{9}
$$

where, $\gamma, z_i, \mu_i, \overline{\mu}_i, i = 1, \ldots, n$ are the optimization variables, characterizes the LMS solution.

Typical Evolution of MIO



A good MIO formulation needs work

- The formulation, just described is one of many MIO formulations for the LMS problem.
- We can improve upon (9) using more sophisticated modeling tools in Integer Optimization, for example, Specially Ordered Sets.

Boosting the performance of MIO

- MIO formulations can tackle problems of small to moderate size quite efficiently. Significantly better than existing methods.
- In general, they are found to benefit significantly from advanced warm-starts.
- Algorithms based on continuous optimization methods (Non-Linear programming), can be used for this purpose.

Continuous Optimization – Algorithm I

- Based on Sequential Linear optimization. Relies on:
  - the decomposition:

$$|y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}| = \underbrace{\sum_{i=1}^{q+1} |y_{(i)} - \mathbf{x}'_{(i)}\boldsymbol{\beta}|}_{H_{q+1}(\boldsymbol{\beta})} - \underbrace{\sum_{i=1}^{q} |y_{(i)} - \mathbf{x}'_{(i)}\boldsymbol{\beta}|}_{H_q(\boldsymbol{\beta})}, \qquad (10)$$

    where $r_{(q)} = y_{(q)} - \mathbf{x}'_{(q)}\boldsymbol{\beta}$.

  - and observe that the function:

$$\boldsymbol{\beta} \mapsto H_q(\boldsymbol{\beta})$$

    is convex in $\boldsymbol{\beta}$

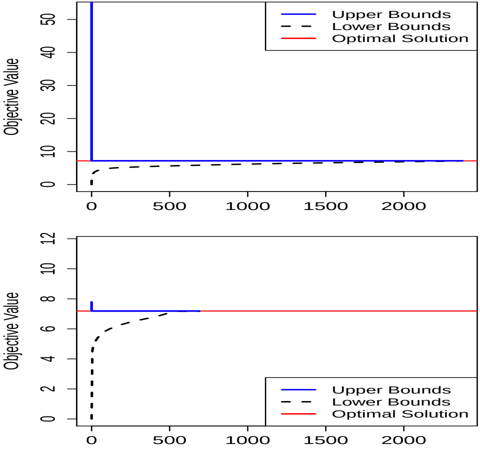
Continuous Optimization – Algorithm II

- Uses a sub-differential based method on

$$|y_{(q)} - \mathbf{x}'_{(q)}\beta|$$

Impact of Warm-Starts

Evolution of MIO (cold-start) [top] vs (warm-start) [bottom]



$n = 501, p = 5$, synthetic example

Acknowledgements

Notes based on:

- Stephen Boyd's EE 364B lecture notes (Stanford Univ.)
- Jim Burke's class notes (Washington Univ.)
- Bertsimas and Shioda "Classication and Regression via Integer Optimization", 2007
- Bertsimas and Mazumder "Least Quantile of Squares via Modern Optimization", 2013