Mixed-Integer Nonlinear Optimization: Applications, Algorithms, and Computation II

Sven Leyffer

Mathematics & Computer Science Division
Argonne National Laboratory

Graduate School in
Systems, Optimization, Control and Networks
Université catholique de Louvain
February 2013
Outline

1. Problem Definition and Assumptions
2. Nonlinear Branch-and-Bound
3. Advanced Nonlinear Branch-and-Bound
4. Multi-Tree Methods
5. Summary and Exercises
Mixed-Integer Nonlinear Optimization

Mixed-Integer Nonlinear Program (MINLP)

\[
\begin{align*}
\text{minimize} & \quad f(x) \\
\text{subject to} & \quad c(x) \leq 0 \\
& \quad x \in X \\
& \quad x_i \in \mathbb{Z} \text{ for all } i \in I
\end{align*}
\]

Assumptions

A1 $X$ is a bounded polyhedral set.

A2 $f$ and $c$ are twice continuously differentiable convex functions.

A3 MINLP satisfies a constraint qualification.

A2 (convexity) most restrictive (relaxed next week);

A3 is technical (MFCQ would have been sufficient);
Overview of Basic Methods

Two broad classes of method

1. Single-tree methods; e.g.
   - Nonlinear branch-and-bound
   - LP/NLP-based branch-and-bound
   - Nonlinear branch-and-cut

   ... build and search a single tree

2. Multi-tree methods; e.g.
   - Outer approximation
   - Benders decomposition
   - Extended cutting plane method

   ... alternate between NLP and MILP solves

Multi-tree methods only evaluate functions at integer points

Concentrate on methods for convex problems today.

Can mix different methods & techniques.
Outline

1. Problem Definition and Assumptions
2. Nonlinear Branch-and-Bound
3. Advanced Nonlinear Branch-and-Bound
4. Multi-Tree Methods
5. Summary and Exercises
Nonlinear Branch-and-Bound

Solve NLP relaxation \((x_i \text{ continuous, not integer})\)

\[
\begin{align*}
\text{minimize } & \quad f(x) \quad \text{subject to } c(x) \leq 0, \quad x \in X \\
\end{align*}
\]

- If \(x_i \in \mathbb{Z} \quad \forall \ i \in I\), then solved MINLP
- If relaxation is infeasible, then MINLP infeasible

... otherwise search tree whose nodes are NLPs:

\[
\begin{align*}
\begin{cases}
\text{minimize } & \quad f(x), \\
\text{subject to } & \quad c(x) \leq 0, \\
& \quad x \in X, \\
& \quad l_i \leq x_i \leq u_i, \quad \forall i \in I.
\end{cases} \\
\text{(NLP}(l, u))
\end{align*}
\]

NLP relaxation is NLP\((-\infty, \infty)\)
Nonlinear Branch-and-Bound

**Branching:** solution $x'$ of $(\text{NLP}(l, u))$ feasible but not integral:

- Find a nonintegral variable, say $x'_i$, $i \in I$.
- Introduce two child nodes with bounds
  \[(l^-, u^-) = (l^+, u^+) = (l, u)\]
  and setting:
  \[u_i^- := \lfloor x'_i \rfloor, \quad \text{and} \quad l_i^+ := \lceil x'_i \rceil\]

- Two new NLPs: $\text{NLP}(l^-, u^-)$ / $\text{NLP}(l^+, u^+)$
  ... corresponding to down/up branch

In practice, store problems on a heap $\mathcal{H}$

... pruning rules limit the tree $\Rightarrow$ no complete enumeration
Nonlinear Branch-and-Bound

Pruning Rules: Let \( U \) upper bound on solution

- **Infeasible**: (NLP\((l, u)\)) infeasible
  \[ \Rightarrow \text{any NLP in subtree is also infeasible.} \]

- **Integer feasible**: solution \( x^{(l,u)} \) of (NLP\((l, u)\)) integral
  - If \( f(x^{(l,u)}) < U \), then new \( x^* = x^{(l,u)} \) and \( U = f(l,u) \).
  - Otherwise, prune node: no better solution in subtree

- **Dominated by \( U \)**: optimal value of (NLP\((l, u)\)), \( f(x^{(l,u)}) \geq U \)
  \[ \Rightarrow \text{prune node: no better integer solution in subtree} \]
Nonlinear Branch-and-Bound

Solve relaxed NLP \((0 \leq y \leq 1)\) continuous relaxation
...solution value provides lower bound

- Branch on \(y_i\) non-integral
- Solve NLPs & branch until
  1. Node infeasible: \(\star\)
  2. Node integer feasible: \(\square\)
     \(\Rightarrow\) get upper bound \((U)\)
  3. Lower bound \(\geq U\): \(\triangleup\)

Search until no unexplored nodes

Software:
- GAMS-SBB, MINLPBB [L]
- BARON [Sahinidis] global
- Couenne [Belotti] global
Nonlinear Branch-and-Bound

Branch-and-bound for MINLP
Choose tol $\epsilon > 0$, set $U = \infty$, add $(NLP(-\infty, \infty))$ to heap $H$.

while $H \neq \emptyset$ do

  Remove $(NLP(l, u))$ from heap: $H = H - \{ NLP(l, u) \}$.
  Solve $(NLP(l, u)) \Rightarrow$ solution $x^{(l, u)}$

  if $(NLP(l, u))$ is infeasible then
  Prune node: infeasible

  else if $f(x^{(l, u)}) > U$ then
  Prune node; dominated by bound $U$

  else if $x^{(l, u)}_i$ integral then
  Update incumbent: $U = f(x^{(l, u)})$, $x^* = x^{(l, u)}$

  else
  BranchOnVariable($x^{(l, u)}_i, l, u, H$)
Nonlinear Branch-and-Bound

BnB is finite, provided $X$ is bounded polyhedron:

**Theorem (Finiteness of Nonlinear BnB)**

*Solve MINLP by nonlinear branch-and-bound, and assume that A1-A3 hold. Then BnB terminates at optimal solution (or indication of infeasibility) after a finite number of nodes.*

**Proof.**

- (A1-A3) $\Rightarrow$ every NLP solved globally
- Boundedness of $X$ $\Rightarrow$ tree is finite
$\Rightarrow$ convergence, see e.g. Theorem 24.1 of [?].
Nonlinear Branch-and-Bound

BnB trees can get pretty large ...

Synthesis MINLP B&B Tree: 10000+ nodes after 360s

... be smart about solving NLPs & searching tree!
Outline

1. Problem Definition and Assumptions
2. Nonlinear Branch-and-Bound
3. Advanced Nonlinear Branch-and-Bound
4. Multi-Tree Methods
5. Summary and Exercises
Advanced Nonlinear BnB

Basic BnB will work, but needs improvements:

- Selection of branching variables
- Node selection strategies
- Inexact NLP solves & hot-starts
- Cutting planes & branch-and-cut
- Software design & modern solvers, e.g. MINOTAUR

... critical for efficient implementation
Advanced Nonlinear BnB: Variable Selection

Ideally choose branching sequence to minimize tree size
... impossible in practice; sequence not known a priori
⇒ choose variable that maximizes increase in lower bound

Let $I_c \subset I$ set of fractional integer variables
... in practice choose subset of important variables (priorities)

**Maximum Fractional Branching**

Branch on variable $i_0$ with largest integer violation:

$$i_0 = \arg \max_{i \in I_c} \left\{ \min \left( x_i - \lfloor x_i \rfloor, \lceil x_i \rceil - x_i \right) \right\},$$

... as bad as random branching [?]
Advanced Nonlinear BnB: Variable Selection

Successful rules estimate change in lower bound after branching

- Increasing lower bound improves pruning
- For $x_i, i \in I$, define degradation estimates $D_i^+$ and $D_i^-$ for increase in lower bound
- Goal: make both $D_i^+$ and $D_i^-$ large!
- Combine $D_i^+$ and $D_i^-$ into single score:

$$s_i := \mu \min(D_i^+, D_i^-) + (1 - \mu) \max(D_i^+, D_i^-),$$

where parameter $\mu \in [0, 1]$ close to 1.

Degradation-Based Branching

Branch on variable $i_0$ with largest integer violation:

$$i_0 = \arg\max_{i \in I_c} \{s_i\}$$

... methods differ by how $D_i^+$ and $D_i^-$ computed
The first approach for computing degradations is ...

**Strong Branching**

Solve $2 \times |I_c|$ NLPs for every potential child node:

- Solution at current (parent) node $(\text{NLP}(l, u))$ is $f_p := f^{(l, u)}$
- $\forall x_i, i \in I_c$ create two temporary NLPs: $\text{NLP}_i(l^-, u^-)$ and $\text{NLP}_i(l^+, u^+)$
- Solve both NLPs ...
  ... if both infeasible, then prune $(\text{NLP}(l, u))$
  ... if one infeasible, then fix integer in parent $(\text{NLP}(l, u))$
  ... otherwise, let solutions be $f_i^+$ and $f_i^-$ and compute

$$D_i^+ = f_i^+ - f_p, \quad \text{and} \quad D_i^- = f_i^- - f_p.$$
Advanced Nonlinear BnB: Variable Selection

Advantage/Disadvantage of strong branching:

- **Good**: Reduce the number of nodes in tree
- **Bad**: Slow overall, because too many NLPs solved
- Solving NLPs approximately does not help

**Fact**: MILP $\neq$ MINLP

LPs hot-start efficiently (re-use basis factors), but NLPs cannot be warm-started (neither IPM nor SQP)!

**Reason** (NLPs are, well ... nonlinear):

- NLP methods are iterative: generate sequence $\{x^{(k)}\}$
- At solution, $x^{(l)}$, have factors from $x^{(l-1)}$ ... out-of-date
Approximate Strong Branching

Simple idea: Use QP (LP) approximation [?]

CPU[s] for root node and round (2 # ints) of strong branching:

<table>
<thead>
<tr>
<th>problem</th>
<th># ints</th>
<th>Full NLP</th>
<th>Cold QP</th>
<th>Hot QP</th>
</tr>
</thead>
<tbody>
<tr>
<td>stockcycle</td>
<td>480</td>
<td>4.08</td>
<td>3.32</td>
<td>0.532</td>
</tr>
<tr>
<td>RSyn0805H</td>
<td>296</td>
<td>78.7</td>
<td>69.8</td>
<td>1.94</td>
</tr>
<tr>
<td>SLay10H</td>
<td>180</td>
<td>18.0</td>
<td>17.8</td>
<td>1.25</td>
</tr>
<tr>
<td>Syn30M03H</td>
<td>180</td>
<td>40.9</td>
<td>14.7</td>
<td>2.12</td>
</tr>
</tbody>
</table>

- Small savings from replacing NLP by QP solves.
- Order of magnitude saving from re-using factors.
Approximate Strong Branching

**Hot-QP Starts in BQPD [Fletcher]**

- parent node is **dual** feasible after branching
- perform steps of **dual** active-set method to get primal feasible
- re-use factors of basis $B = LU$
- re-use factors of dense reduced Hessian $Z^T H Z = L^T D L$
- use $LU$ and $L^T D L$ to factorize KKT system

\[
\begin{bmatrix} H & A^T \\ A & 0 \end{bmatrix}
\]

where $B^{-1} = [A : V]^{-1} = \begin{bmatrix} Y \\ Z \end{bmatrix}$

- 2-3 pivots to re-optimize independent of problem size
Approximate Strong Branching

Parametric QP solve
Performance Profiles [Dolan and More, 2002]

Performance profiles
Clever way to display a benchmark

$\forall \text{ solver } s \log_2 \left( \frac{\# \text{ iter}(s, p)}{\text{best_iter}(p)} \right)$

$p \in \text{ problem}$

- “probability distribution”: solver “A” is at most $x$-times slower than best.
- Origin shows percentage of problems where solver “A” is best.
- Asymptotics shows reliability of solver “A”.
Performance Profiles (Formal Definition)

Performance ratio of $t_{p,s}$ for $p \in \mathcal{I}$ of problems, $s \in \mathcal{S}$ of solvers:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,i} \mid i \in \mathcal{S}\}}$$

distribution function $\rho_s(\tau)$ for solver $s \in \mathcal{S}$

$$\rho_s(\tau) = \frac{\text{size}\{p \in \mathcal{I} \mid r_{p,s} \leq \tau\}}{|\mathcal{I}|}.$$  

$\rho_s(\tau)$ probability that solver $s$ is at most $\tau \times$ slower than best
Approximate Strong Branching

Performance (nodes) of NLP/QP/LP strong branching

Proportion of problems x times worse than best

- NLP-fullstrong
- QP-fullstrong
- LP-fullstrong

Performance (nodes) of NLP/QP/LP strong branching
Approximate Strong Branching

Performance (CPU time) of NLP/QP/LP strong branching
**Advanced Nonlinear BnB: Variable Selection**

**Pseudocost Branching**

Keep history of past branching to estimate degradations

- \( n_{i}^{+}, n_{i}^{-} \) number of times up/down node solved for variable \( i \)
- \( p_{i}^{+}, p_{i}^{-} \) pseudocosts updated when child solved:

\[
p_{i}^{+} = \frac{f_{i}^{+} - f_{p}}{\lceil x_{i} \rceil - x_{i}} + p_{i}^{+}, \quad n_{i}^{+} = n_{i}^{+} + 1 \quad \text{or} \quad p_{i}^{-} = \ldots \quad n_{i}^{-} = \ldots
\]

- Compute estimates of \( D_{i}^{+} \) and \( D_{i}^{-} \) or branching:

\[
D_{i}^{+} = (\lceil x_{i} \rceil - x_{i}) \frac{p_{i}^{+}}{n_{i}^{+}} \quad \text{and} \quad D_{i}^{-} = (x_{i} - \lfloor x_{i} \rfloor) \frac{p_{i}^{-}}{n_{i}^{-}}.
\]

- Initialize pseudocosts with strong branching
- Good estimates for MILP, [?]
- Not clear how to update, if NLP infeasible ... \( \ell_1 \) penalty?
Following approach combines strong branching and pseudocosts

**Reliability Branching**

Strong branching early, then pseudocost branching

- While \( n_i^+ \) or \( n_i^- \) \( \leq \tau \) (= 5) do strong branching on \( x_i \)
- Once \( n_i^+ \) or \( n_i^- \) > \( \tau \) switch to pseudocost

Important alternatives to variables branching:

- SOS branching, see [?]
- Branching on split disjunctions

\[
\left( a^T x_I \leq b \right) \lor \left( a^T x_I \geq b + 1 \right)
\]

where \( a \in \mathbb{Z}^p \) and \( b \in \mathbb{Z} \) ... conceptually like conjugate directions
Advanced Nonlinear BnB: Node Selection

Strategic decision on which node to solve next.

Goals of node selection

- Find good feasible solution quickly to reduce upper bound, $U$
- Prove optimality of incumbent $x^*$ by increasing lower bound

Popular strategies:

1. Depth-first search
2. Best-bound search
3. Hybrid schemes
Advanced Nonlinear BnB: Depth-First Search

**Depth-First Search**
Select deepest node in tree (or last node added to heap $\mathcal{H}$)

**Advantages:**
- Easy to implement (Sven likes that ;-)  
- Keeps list of open nodes, $\mathcal{H}$, as small as possible  
- Minimizes the change to next NLP ($\text{NLP}(l, u)$):
  ... only single bound changes $\Rightarrow$ better hot-starts

**Disadvantages:**
- poor performance if no upper bound is found:  
  $\Rightarrow$ explores nodes with a lower bound larger than solution
**Best-Bound Search**

Select node with best lower bound

**Advantages:**
- Minimizes number of nodes for fixed sequence of branching decisions, because all explored nodes would have been explored independent of upper bound

**Disadvantages:**
- Requires more memory to store open problems
- Less opportunity for warm-starts of NLPs
- Tends to find integer solutions at the end
Advanced Nonlinear BnB: Best-Bound Search

1. **Best Expected Bound:** node with best bound after branching:

   \[ b^+_p = f_p + ([x_i] - x_i) \frac{p_i^+}{n_i^+} \text{ and } b^-_p = f_p + (x_i - [x_i]) \frac{p_i^-}{n_i^-} . \]

   Next node is \( \max_p \{ \min (b^+_p, b^-_p) \} \).

2. **Best Estimate:** node with best expected solution in subtree

   \[ e_p = f_p + \sum_{i: x_i \text{ fractional}} \min \left( ([x_i] - x_i) \frac{p_i^+}{n_i^+} , (x_i - [x_i]) \frac{p_i^-}{n_i^-} \right) , \]

   Next node is \( \max_p \{ e_p \} \).

... good search strategies combine depth-first and best-bound
Advanced Nonlinear BnB: Inexact NLP Solves

Role for inexact solves in MINLP

- Provide approximate values for strong branching
- Solve NLPs inexactly during tree-search:
  - consider single SQP iteration
    - perform early branching if limit seems non-integral
    - augmented Lagrangian dual for bounds
  - considers single SQP iteration
    - use outer approximation instead of dual
    - numerical results disappointing

  ... reduce solve time by factor 2-3 at best

- New idea: search QP tree & exploit hot-starts for QPs
  - QP-diving discussed next ...
Advanced Nonlinear BnB: QP-Diving

Branch-and-bound solves huge number of NLPs ⇒ bottleneck!

QP-Diving Tree-Search:
- solve root node & save factors from last QP solve
- same KKT for whole subtree
- perform MIQP tree-searches
  - depth-first search: ⇒ fast hot-starts
  - back-track:
    warm-starts

Need new fathoming rules ...

... alternative: change QP approximation after back-track
Advanced Nonlinear BnB: QP-Diving

Assume MINLP is convex

**QP-Diving Tree-Search:**
Solve QPs until

1. **QP infeasible:** ●
   ... QP is relaxation of NLP

2. **Node integer feasible:** □
   ⇒ NLP to get upper bnd ($U$)
   ... QP over-/under-estimates
   ⇒ resolve

3. **Infeasible O-cut $\eta < U$: ▲**
   Linear O-cut: $\eta \geq f_k + g^T_k d$

[Diagram of QP-Diving Tree-Search]

- Full NLP
- $y=0$, $y=1$
- $Q(P)$
- Infeasible
- Integer feasible
- Dominated linear objf
- Solve NLP
New Extended Performance Profiles

Performance ratio of $t_{p,s}$ for $p \in \mathcal{I}$ of problems, $s \in S$ of solvers:

$$\hat{r}_{p,s} = \frac{t_{p,s}}{\min\{t_{p,i} \mid i \in S, i \neq s\}}$$

distribution function $\rho_s(\tau)$ for solver $s \in S$

$$\hat{\rho}_s(\tau) = \frac{\text{size}\{p \in \mathcal{I} \mid \hat{r}_{p,s} \leq \tau\}}{|\mathcal{I}|}.$$ 

- $\hat{\rho}_s(\tau)$ probability that solver $s$ is at most $\tau \times$ slower than best
- For $\hat{r}_{p,s} \geq 1$ get standard performance profile
- Extension: $\hat{r}_{p,s} < 1$ if solver $s$ is fastest for instance $p$
- $\hat{\rho}_s(0.25)$ probability that solver $s$ is $4 \times$ faster than others
CPU-Times for MINOTAUR with Hot-Starts (IPOPT)

Hot-started QP give a huge improvement
CPU-Times for MINOTAUR with Hot-Starts (filterSQP)

Hot-started QP give a huge improvement
## Typical Results

### RSyn0840M02M

<table>
<thead>
<tr>
<th>Solver</th>
<th>CPU</th>
<th>NLPs</th>
<th>CPU/100NLPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPOPT</td>
<td>7184.91</td>
<td>69530</td>
<td>10.3335</td>
</tr>
<tr>
<td>filterSQP</td>
<td>7192.54</td>
<td>37799</td>
<td>19.0284</td>
</tr>
<tr>
<td>QP-Diving</td>
<td>5276.23</td>
<td>1387837</td>
<td>0.3802</td>
</tr>
</tbody>
</table>

⇒ many more nodes ... a little faster.

### CLay0305H

<table>
<thead>
<tr>
<th>Solver</th>
<th>CPU</th>
<th>NLPs</th>
<th>CPU/100NLPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPOPT</td>
<td>1951.1</td>
<td>16486</td>
<td>11.8349</td>
</tr>
<tr>
<td>filterSQP</td>
<td>849.74</td>
<td>16717</td>
<td>5.0831</td>
</tr>
<tr>
<td>QP-Diving</td>
<td>97.89</td>
<td>24029</td>
<td>0.4074</td>
</tr>
</tbody>
</table>

⇒ similar number of nodes ... much faster!
MINOTAUR: A New Software Framework for MINLP

**Mixed Integer Nonlinear Optimization Toolkit:**
Algorithms, Underestimators & Relaxations

Goal: Implement a Range of Algorithms in Common Framework

- Fast, usable MINLP solver.
- **Flexibility** for developing new algorithms.
- **Ease** of developing new algorithms.
MINOTAUR’s Four Main Components

Interfaces for reading input

- AMPL

Engines to solve LP/NLP/QP

- QP: BQPD
- NLP: FilterSQP/IPOPT
- LP: OSI-CLP

Algorithms to solve MINLP

- Branch-and-Bound
- Outer-Approximation
- Quesada-Grossmann
- Branch-and-Refine

Base

Data Structures:

- Problem
- Objective & Constraints
- Functions
- Modifications
- Gradient, Jacobian, Hessian

Tools for Search:

- Node Processors
- Node Relaxers
- Branchers
- Tree Manager

Utilities

- Loggers & Timers
- Options
MINOTAUR’s Four Main Components

**Interfaces** for reading input
- AMPL
- Your Interface Here

**Engines** to solve LP/NLP/QP
- QP: BQPD
- NLP: FilterSQP/IPOPT
- LP: OSI-CLP
- Your engine here

**Algorithms** to solve MINLP
- Branch-and-Bound
- Outer-Approximation
- Quesada-Grossmann
- Branch-and-Refine
- Your algorithm here

**Base**

- **Your Data Structures:**
  - Problem
  - Objective & Constraints
  - Functions
  - Modifications
  - Gradient, Jacobian, Hessian
- **Your Tools for Search:**
  - Node Processors
  - Node Relaxers
  - Branchers
  - Tree Manager
- **Utilities**
  - Loggers & Timers
  - Options

Highly Customizable
MINOTAUR Approach to MINLP: Handlers

- Branch-and-{Bound||Cut||Reduce} algorithms require methods to:
  - Relax a problem
  - Reformulate a problem
  - Presolve a problem
  - Check feasibility of a given point
  - Separate a given point
  - Find a branching candidate
  - Branch on a candidate

... methods depend on structure of constraints or objective
MINOTAUR Approach to MINLP: Handlers

- Branch-and-{Bound||Cut||Reduce} alg\textsuperscript{s} require methods to
  - Relax a problem
  - Reformulate a problem
  - Presolve a problem
  - Check feasibility of a given point
  - Separate a given point
  - Find a branching candidate
  - Branch on a candidate

... methods depend on structure of constraints or objective

- **Handlers** implement (type specific) above methods
  ... adopted from constraint-programming solver: SCIP

- MINOTAUR’s Handlers:
  - IntVarHandler
  - LinearHandler
  - BilinearHandler
  - MultilinearHandler, ...

- Branch-and-xxx components like Nodeprocessor, Brancher can be agnostic to type of objective and constraints.
MINLP Branch-and-Bound in MINOTaur

How to write an NLP Branch-and-Bound solver in MINOTaur:

Node Relaxer

Do nothing!

Node Processor

Solve Relaxn.

\[ lb \geq ub? \]

yes \rightarrow Return

no \rightarrow Is Feasible?

yes \rightarrow Update ub

no \rightarrow Branch

Brancher

Pick a fractional variable.

Use Minotaur::IntVarHandler for all three
How to write an NLP Branch-and-Bound solver in MINOTAUR:

**Node Relaxer**
- Do nothing!

**Node Processor**
- Solve Relaxation
- \( \text{lb} \geq \text{ub} \) ?
- Is Feasible?
- \( \text{yes} \rightarrow \text{Return} \)
- \( \text{no} \rightarrow \text{Branch} \)
- \( \text{lb} \geq \text{ub} \)?
- Is Feasible?
- \( \text{yes} \rightarrow \text{Update ub Return} \)
- \( \text{no} \rightarrow \text{Branch} \)

**Brancher**
- Pick a fractional variable.

Use `Minotaur::IntVarHandler` for all three:

```cpp
relax() {
    // empty
}

bool isFeasible() {
    // test integrality
}

cand* findBrCandidates() {
    // return fractional vars
    branch(cand) {
    // return modifications
}
MINLP Branch-and-Bound in MINOTAUR

How to write an **NLP Branch-and-Bound** solver in MINOTAUR:

**Node Relaxer**

- Do nothing!

**Node Processor**

- Solve Relaxn.
- \( lb \geq ub? \)
  - yes: Return
  - no:
    - Is Feasible?
      - yes: Update ub Return
      - no: Branch

**Brancher**

- Pick a fractional variable.

Use Minotaur::IntVarHandler for all three

**Solver in < 200 lines:**

- Read instance. Load Engine.
- Create IntVarHandler.
- Load it to NodeProcessor, Brancher, NodeRelaxer.
- Solve
## Features

<table>
<thead>
<tr>
<th></th>
<th>Bonmin</th>
<th>FilMINT</th>
<th>BARON</th>
<th>Couenne</th>
<th>Minotaur</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algorithms:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NLP B&amp;B</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Branch &amp; Cut</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Branch &amp; Reduce</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Support for Nonlinear Functions:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comput. Graph</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Nonlin. Reform.</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>Native Derivat.</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Interfaces:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIMMS</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>AMPL</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GAMS</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Open Source</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
MINOTAUR Performance

Time taken for 463 MINLP Instances from GAMS, MacMINLP, CMU test-sets.
MINOTAUR’s Soft-Wear Stack

... available at www.mcs.anl.gov/minotaur
MINLP Trees are Huge

Synthesis MINLP B&B Tree: 10000+ nodes after 360s

⇒ use MILP solvers to search tree?
Multi-Tree Methods

MILP solvers much better developed than MINLP
- LPs are easy to hot-start
- Decades of investment into software
- MILPs much easier; e.g. no need for constraint qualifications
⇒ developed methods that exploit this technology

Multi-Tree Methods
- Outer approximation [?]
- Benders decomposition [?]
- Extended cutting plane method [?]
... solve a sequence of MILP (and NLP) problems

Multi-tree methods evaluate functions “only” at integer points!
Recall the $\eta$-MINLP formulation

$$\begin{aligned}
\text{minimize} & \quad \eta, \\
\text{subject to} & \quad f(x) \leq \eta, \\
& \quad c(x) \leq 0, \\
& \quad x \in X, \\
& \quad x_i \in \mathbb{Z}, \ \forall i \in I.
\end{aligned}$$

where we have “linearized” the objective: $\eta \geq f(x)$

Use $\eta$-MINLP in this section
Outer Approximation

Mixed-Integer Nonlinear Program (MINLP)

\[
\begin{align*}
\text{minimize} & \quad f(x) \quad \text{subject to} \quad c(x) \leq 0, \ x \in X, \ x_i \in \mathbb{Z} \ \forall \ i \in I
\end{align*}
\]

NLP subproblem for fixed integers \( x_I^{(j)} \):

\[
\begin{align*}
\text{NLP}(x_I^{(j)}) & \begin{cases} 
\text{minimize} \quad f(x) \\
\text{subject to} \quad c(x) \leq 0 \\
x \in X \quad \text{and} \quad x_I = x_I^{(j)},
\end{cases}
\end{align*}
\]

with solution \( x^{(j)} \).

If \( \text{NLP}(x_I^{(j)}) \) infeasible then solve feasibility problem ...
Outer Approximation

Mixed-Integer Nonlinear Program (MINLP)

\[
\begin{align*}
\text{minimize } & \quad f(x) \quad \text{subject to } c(x) \leq 0, \ x \in X, \ x_i \in \mathbb{Z} \ \forall \ i \in I
\end{align*}
\]

NLP feasibility problem for fixed integers \(x_i^{(j)}\):

\[
\begin{align*}
\text{minimize } & \quad \sum_{i \in J^+} w_i c_i^+(x) \\
\text{subject to } & \quad c_i(x) \leq 0, \ i \in J \\
& \quad x \in X \quad \text{and } x_I = x_i^{(j)}
\end{align*}
\]

where \(w_i > 0\) are weights and solution is \(x^{(j)}\).

\(F(x_i^{(j)})\) generalize minimum norm solution

... provides certificate that \(\text{(NLP}(x_i^{(j)}))\) infeasible
Convexity of $f$ and $c$ implies that

**Lemma (Supporting Hyperplane)**

Linearization about solution $x^{(j)}$ of $(NLP(x^{(j)}_i))$ or $(F(x^{(j)}_i))$, 

\[(OA) \quad \eta \geq f^{(j)} + \nabla f^{(j)^T} (x - x^{(j)}) \quad \text{and} \quad 0 \geq c^{(j)} + \nabla c^{(j)^T} (x - x^{(j)}) ,\]

are outer approximations of the feasible set of $\eta$-MINLP.

**Lemma (Feasibility Cuts)**

If $(NLP(x^{(j)}_i))$ infeasible, then $(OA)$ cuts off $x_i = x^{(j)}_i$. 


Outer Approximation

Mixed-Integer Nonlinear Program (η-MINLP)

\[
\min_{x} \eta \quad \text{s.t.} \quad \eta \geq f(x), \quad c(x) \leq 0, \quad x \in X, \quad x_i \in \mathbb{Z} \quad \forall \ i \in I
\]

Define index set of all possible feasible integers, \(\mathcal{X}\)

\[
\mathcal{X} := \left\{ x^{(j)} \in X : x^{(j)} \text{ solves } (\text{NLP}(x^{(j)})) \right\} \quad \text{or} \quad (F(x^{(j)})) \right\}.
\]

... boundedness of \(X\) implies \(|\mathcal{X}| < \infty\)

Construct equivalent OA-MILP (outer approximation MILP)

\[
\begin{aligned}
&\text{minimize} \quad \eta, \\
&\text{subject to} \quad \eta \geq f^{(j)} + \nabla f^{(j)}^T (x - x^{(j)}), \quad \forall x^{(j)} \in \mathcal{X} \\
&\quad 0 \geq c^{(j)} + \nabla c^{(j)}^T (x - x^{(j)}), \quad \forall x^{(j)} \in \mathcal{X} \\
&\quad x \in X, \\
&\quad x_i \in \mathbb{Z}, \quad \forall i \in I.
\end{aligned}
\]
Outer Approximation in Less Than 1000 Words
Outer Approximation

Theorem (Equivalence of OA-MILP and MINLP)

Let assumptions A1-A3 hold

- If $x^*$ solves MINLP, then it also solves OA-MILP
- If $(\eta^*, x^*)$ solves OA-MILP, then $\eta^*$ is optimal value of MINLP, and $x_i^*$ is an optimal integer.

MILP and MINLP are not quite equivalent

Example where OA-MILP not equivalent to MINLP

\[
\begin{align*}
\text{minimize} & \quad x_3 \\
\text{subject to} & \quad (x_1 - \frac{1}{2})^2 + x_2^2 + x_3^3 \leq 1, \quad x_1 \in \mathbb{Z} \cap [-1, 2].
\end{align*}
\]

... OA-MILP has no coefficients for $x_2$ ... undefined
Outer Approximation Algorithm

Solving OA-MILP clearly not sensible; define upper bound as

\[ U^k := \min_{j \leq k} \left\{ f(j) \mid \text{(NLP}(x^{(j)})\text{ is feasible}) \right\}. \]

Define relaxation of OA-MILP, using \( \mathcal{X}^k \subset \mathcal{X} \), with \( \mathcal{X}^0 = \{0\} \)

\[
M(\mathcal{X}^k) \begin{cases} 
\text{minimize} & \eta, \\
\text{subject to} & \eta \leq U^k - \epsilon \\
& \eta \geq f(j) + \nabla f(j)^T (x - x^{(j)}), \forall x^{(j)} \in \mathcal{X}^k \\
& 0 \geq c(j) + \nabla c(j)^T (x - x^{(j)}), \forall x^{(j)} \in \mathcal{X}^k \\
& x \in X, \\
& x_i \in \mathbb{Z}, \forall i \in I.
\end{cases}
\]

... build up better OA \( \mathcal{X}^k \) iteratively for \( k = 0, 1, \ldots \)
Outer Approximation Algorithm

Outer approximation
Given $x^{(0)}$, choose tol $\epsilon > 0$, set $U^{-1} = \infty$, set $k = 0$, and $\mathcal{X}^{-1} = \emptyset$.

repeat

Solve (NLP($x^{(j)}_I$)) or (F($x^{(j)}_I$)); solution $x^{(j)}$.

if (NLP($x^{(j)}_I$)) feasible $\& f^{(j)} < U^{k-1}$ then

Update best point: $x^* = x^{(j)}$ and $U^k = f^{(j)}$.

else

Set $U^k = U^{k-1}$.

Linearize $f$ and $c$ about $x^{(j)}$ and set $\mathcal{X}^k = \mathcal{X}^{k-1} \cup \{j\}$.

Solve ($M(\mathcal{X}^k)$), let solution be $x^{(k+1)}$ $\&$ set $k = k + 1$.

until MILP ($M(\mathcal{X}^k)$) is infeasible
Outer Approximation Algorithm

Alternate between solve $\text{NLP}(y_j)$ and MILP relaxation

\[ \text{MILP} \Rightarrow \text{lower bound}; \quad \text{NLP} \Rightarrow \text{upper bound} \]

\[ \text{... convergence follows from convexity & finiteness} \]
Outer Approximation Algorithm

Alternate between solve $\text{NLP}(y_j)$ and MILP relaxation

MILP $\Rightarrow$ lower bound; NLP $\Rightarrow$ upper bound

... convergence follows from convexity & finiteness
Outer Approximation Algorithm

Theorem (Convergence of Outer Approximation)

Let Assumptions A1-A3 hold, then outer approximation terminates finitely at optimal solution of MINLP or indicates it is infeasible.

Outline of Proof.

- Optimality of $x^{(j)}$ in $(\text{NLP}(x^{(j)}))$
  \[ \Rightarrow \eta \geq f^{(j)} \] for feasible point of $(M(X^k))$
  ... ensures finiteness, since $X$ compact

- Convexity $\Rightarrow$ linearizations are supporting hyperplanes
  ... ensures optimality
Worst Case Example of Outer Approximation

[?] construct infeasible MINLP:

\[
\begin{align*}
\text{minimize} & \quad 0 \\
\text{subject to} & \quad \sum_{i=1}^{n} \left( y_i - \frac{1}{2} \right)^2 \leq \frac{n-1}{4} \\
& \quad y \in \{0, 1\}^n
\end{align*}
\]

Intersection of ball of radius \( \frac{\sqrt{n-1}}{2} \) with unit hypercube.

**Lemma**

*OA cannot cut more than one vertex of the hypercube. MILP master problem feasible for any \( k < 2^n \) OA cuts*

**Theorem**

*OA visits all \( 2^n \) vertices*
Benders Decomposition

Can derive Benders cut from outer approximation:

- Take optimal multipliers $\lambda^{(j)}$ of $\text{(NLP}(x^{(j)}_I))$
- Sum outer approximations

\[
\eta \geq f^{(j)} + \nabla f^{(j)T} (x - x^{(j)}) + \lambda^{(j)T} (0 \geq c^{(j)} + \nabla c^{(j)T} (x - x^{(j)}))
\]

Using KKT conditions wrt continuous variables $x_C$:
\[
0 = \nabla_C \mathcal{L}^{(j)} = \nabla_C f + \nabla_C c \lambda^{(j)} \& \lambda^{(j)T} c^{(j)} = 0
\]
... eliminates continuous variables, $x_C$

Benders cut only involves integer variables $x_I$.
Can write cut as $\eta \geq f^{(j)} + \mu^{(j)T} (x_I - x^{(j)}_I)$,
where $\mu^{(j)}$ multiplier of $x = x^{(j)}_I$ in $\text{(NLP}(x^{(j)}_I))$
Benders Decomposition

For MINLPs with convex problems functions $f$, $c$, we can show:

1. **Benders cuts are weaker than outer approximation**
   - Benders cuts are linear combination of OA

2. **Outer Approximation & Benders converge finitely**
   - Functions $f$, $c$ convex $\Rightarrow$ OA cuts are outer approximations
   - OA cut derived at optimal solution to NLP subproblem
     $\Rightarrow$ feasable descend directions
     ... every OA cut corresponds to first-order condition
   - Cannot visit same integer $x^{(j)}_I$ more than once
     $\Rightarrow$ terminate finitely at optimal solution

Readily extended to situations where \((\text{NLP}(x^{(j)}_I))\) not feasible.
Extended Cutting Plane (ECP) Method

ECP is variation of OA
- Does not solve any NLPs
- Linearize $f$, $c$ around solution of MILP, $x^{(k)}$:
  - If $x^{(k)}$ feasible in linearization, then solved MINLP
  - Otherwise, pick linearization violated by $x^{(k)}$ and add to MILP

Properties of ECP
- Convergence follows from OA & Kelley’s cutting plane method
- NLP convergence rate is linear
- Can visit same integer more than once ...
  ... single-tree methods use ECP cuts to speed up convergence
Summary of Multi-Tree Methods

Three Classes of Multi-Tree Methods

1. Outer approximation based on first-order expansion
2. Benders decomposition linear combination of OA cuts
3. Extended cutting plane method: avoids NLP solves

Common Properties of Multi-Tree Methods

- Only need to solve final MILP to optimality
  ... can terminate MILP early ... adding more NLPs
- Can add cuts from incomplete NLP solves
- Worst-case example for OA also applies for Benders and ECP
- No warm-starts for MILP ... expensive tree-search

... motivates single-tree methods next ...
Outline

1. Problem Definition and Assumptions
2. Nonlinear Branch-and-Bound
3. Advanced Nonlinear Branch-and-Bound
4. Multi-Tree Methods
5. Summary and Exercises
Summary and Exercises

Key points
- Single and multi-tree methods have advantages
- Exploit linearity (or QP) as much as possible
- Implementation matters ... many modern solvers

Exercise for Credit
Prepare a 2-5 minute short chat about why you are interested in MINLP. Present on Monday after the lecture. Email Sven to let him know that you will talk. You can (but do not have to) use slides.


An outer-approximation algorithm for a class of mixed-integer nonlinear programs.

Generalized Benders decomposition.

On the existence of convex decompositions of partially separable functions.

An outer-inner approximation for separable MINLPs.
Technical report, LIF, Faculté des Sciences de Luminy, Université de Marseille.

Integrating SQP and branch-and-bound for mixed integer nonlinear programming.

A computational study of search strategies in mixed integer programming.

Preprocessing and probing techniques for mixed integer programming problems.

