

Branch-and-bound for convex mixed-integer program

Summer School on Nonlinear Optimization with Mixed Integer and Complementarity Constraints

Moritz Diehl, Andrea Ghezzi, Sven Leyffer and Armin Nurkanović

Argonne National Laboratory, University of Freiburg, and Mosek

October 1, 2026

Branch-and-bound for convex mixed-integer program

- 1 Fundamental Building Blocks
- 2 Nonlinear Branch-and-Bound
- 3 Generic Nonlinear Branch-and-Cut
- 4 LP/NLP-Based Branch-and-Bound
- 5 Summary and Take-Away Points: Branch-and-Cut

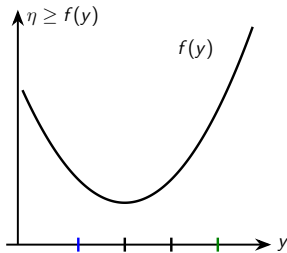
Fundamental Building Blocks of Algorithms for MINLP

Mixed-Integer Nonlinear Programming (MINLP) problem

$$\underset{x,y}{\text{minimize}} \quad f(x,y) \quad \text{subject to} \quad g(x,y) \leq 0, \quad y \in \mathbb{Z}^p$$

... related linearized objective: η -MINLP

$$\underset{x,y,\eta}{\text{minimize}} \quad \eta \quad \text{subject to} \quad \eta \geq f(x,y), \quad 0 \geq g(x,y), \quad y \in \mathbb{Z}^p$$



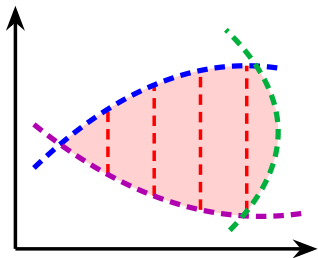
Fundamental Building Blocks of Algorithms

- 1 Relaxation: generate a simpler problem, easy-to-solve
- 2 Constraint enforcement: modify simpler problem to get solution
- 3 Upper Bounds: pruning search space for efficiency

Relaxation and Constraint Enforcement

Relaxation

- Used to compute a lower bound on the optimum
- Obtained by enlarging feasible set; e.g. ignore $y \in \mathbb{Z}$
- Typically much easier to solve than MINLP

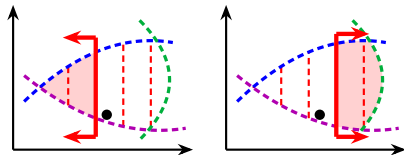


Constraint Enforcement

- Exclude (infeasible) relaxations from solutions
- Refine/tighten relaxation; e.g. add valid cuts

Upper Bounds

- Obtained from any feasible point; e.g. solve NLP for fixed y



Relaxations of Integrality

Definition (Relaxation)

Optimization problem $\min\{\check{f}(x, y) : (x, y) \in \mathcal{R}\}$ is a **relaxation** of $\min\{f(x, y) : (x, y) \in \mathcal{F}\}$, iff $\mathcal{R} \supset \mathcal{F}$ and $\check{f}(x, y) \leq f(x, y)$ for all $(x, y) \in \mathcal{F}$.

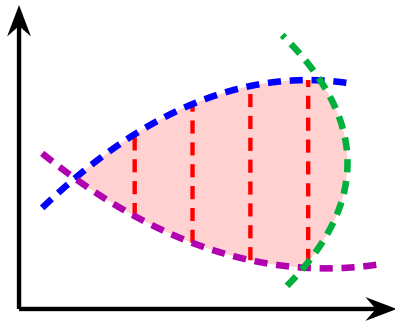
Goal: relaxation **easy to solve globally**, e.g. MILP or NLP

Relaxing Integrality

- Relax Integrality $y_i \in \mathbb{Z}$ to $y_i \in \mathbb{R}$
- Gives *nonlinear relaxation* of MINLP, or NLP:

$$\begin{cases} \underset{x}{\text{minimize}} & f(x, y), \\ \text{subject to} & g(x, y) \leq 0, \\ & (x, y) \text{ continuous} \end{cases}$$

- Used in branch-and-bound algorithms



Relaxations of Nonlinearity

Definition (Relaxation)

Optimization problem $\min\{\check{f}(x, y) : (x, y) \in \mathcal{R}\}$ is a **relaxation** of $\min\{f(x, y) : (x, y) \in \mathcal{F}\}$, iff $\mathcal{R} \supset \mathcal{F}$ and $\check{f}(x, y) \leq f(x, y)$ for all $(x, y) \in \mathcal{F}$.

Goal: relaxation **easy to solve globally**, e.g. MILP or NLP

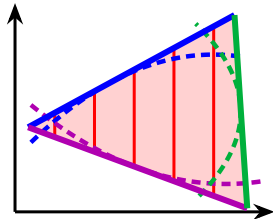
Relaxing Nonlinearity

- Relax nonlinearity
- Gives *linear relaxation* of MINLP, or MILP:

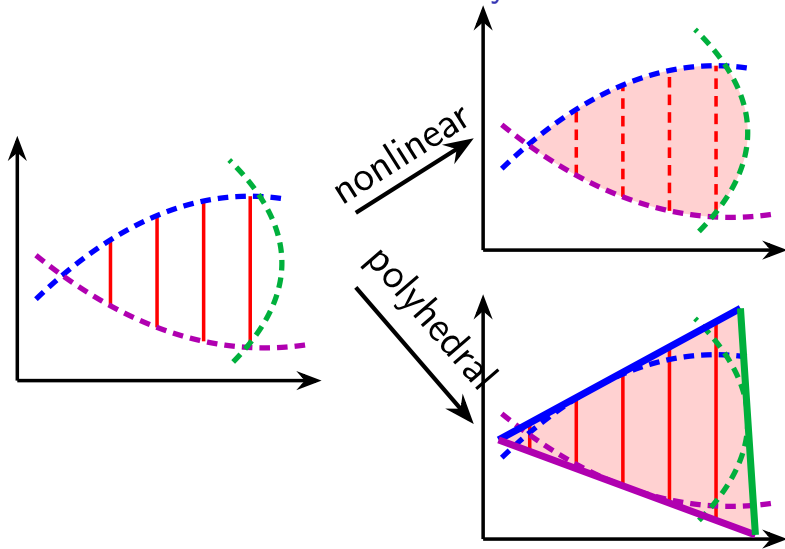
$$\begin{cases} \underset{x}{\text{minimize}} & c_x^T x + c_y^T y, \\ \text{subject to} & A_x^T x + A_y^T y + b \leq 0, \\ & y \text{ integer} \end{cases}$$

where $c_x^T x + c_y^T y \leq f(x, y)$ & $A_x^T x + A_y^T y + b \leq c(x, y)$

- Used in cutting plane algorithms



Two Types of Relaxation: Nonlinear & Polyhedral



Nonlinear and polyhedral relaxation

Relaxations

Relaxations can be combined to produce better algorithms

- Relax convex underestimators via supporting hyperplanes.
- Relax integrality of polyhedral relaxation to obtain an LP.

Relaxations are useful because we have following result:

Theorem (Relaxation Property)

If the solution of the relaxation of the η -MINLP is feasible in the η -MINLP, then it solves the MINLP.

... but if solution of relaxation is not feasible, then need ...

Constraint Enforcement

Goal: Given solution of relaxation, $(\hat{x}, \hat{y})$, not feasible in MINLP, exclude it from further consideration to ensure convergence

Three constraint enforcement strategies

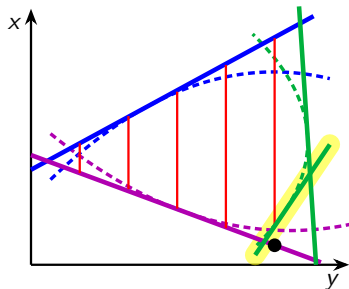
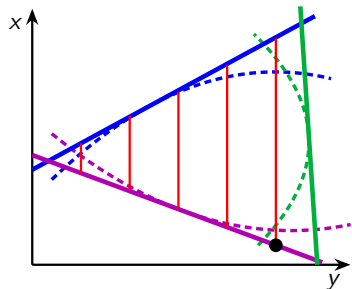
- ① Relaxation refinement: tighten the relaxation
- ② Branching: disjunction to exclude set of non-integer points
- ③ Spatial branching: divide region into sub-regions later

Strategies can be combined ...

Constraint Enforcement: Refinement

Tighten the relaxation to remove current solution $(\hat{x}, \hat{y})$ of relaxation

- Add a **valid inequality** to relaxation, i.e. an inequality that is satisfied by all feasible solutions of MINLP
- Valid inequality is called a **cut** if it excludes $(\hat{x}, \hat{y})$
- Used in Benders decomposition and outer approximation.
- MILP: cuts are basis for branch-and-cut techniques.



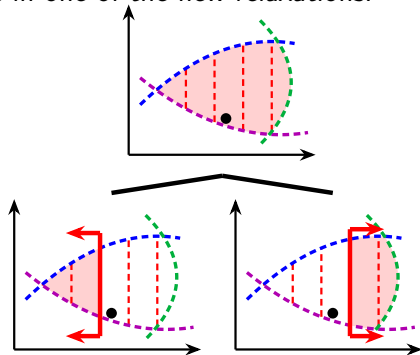
Constraint Enforcement: Branching

Eliminate current $\hat{x}$ solution by branch on integer variables:

- 1 Select fractional $\hat{y}_i$ for some i
- 2 Create two new relaxations by adding

$$y_i \leq \lfloor \hat{y}_i \rfloor \quad \text{and} \quad y_i \geq \lceil \hat{y}_i \rceil \quad \text{respectively}$$

... solution to MINLP lies in one of the new relaxations.



... creates branch-and-bound tree

Branch-and-bound for convex mixed-integer program

- 1 Fundamental Building Blocks
- 2 Nonlinear Branch-and-Bound**
- 3 Generic Nonlinear Branch-and-Cut
- 4 LP/NLP-Based Branch-and-Bound
- 5 Summary and Take-Away Points: Branch-and-Cut

Problem Statement and Assumptions

Mixed-Integer Nonlinear Optimization (MINLP) problem

$$\underset{x,y}{\text{minimize}} \ f(x,y) \quad \text{subject to} \ g(x,y) \leq 0, \quad y \in \mathbb{Z}^p$$

Basic Assumptions for **Convex** MINLP

A1 Feasible set contained in bounded polyhedral set

A2 f and g twice continuously differentiable **convex**

A3 MINLP satisfies a constraint qualification

A2 (convexity) **most restrictive** (show how to relax later)

A3 is technical (MFCQ would have been sufficient)

Nonlinear Branch-and-Bound

Solve NLP relaxation (y continuous, not integer)

$$\underset{x,y}{\text{minimize}} \quad f(x,y) \quad \text{subject to} \quad g(x,y) \leq 0$$

- If $y_i \in \mathbb{Z} \forall i$, then solved MINLP
- If relaxation is infeasible, then MINLP infeasible

... otherwise search tree whose nodes are NLPs:

$$\left\{ \begin{array}{l} \underset{x}{\text{minimize}} \quad f(x,y), \\ \text{subject to} \quad g(x,y) \leq 0, \\ \quad \quad \quad l_i \leq y_i \leq u_i, \quad \forall i \end{array} \right. \quad (\text{NLP}(l, u))$$

NLP relaxation is $\text{NLP}(-\infty, \infty)$

... search tree

Nonlinear Branch-and-Bound

Solve relaxed NLP ($0 \leq y \leq 1$ continuous relaxation)

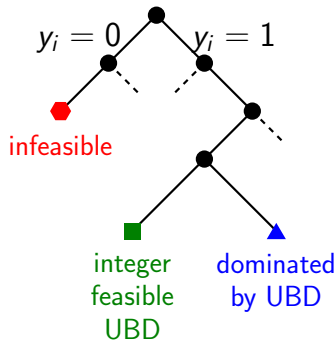
... solution value provides lower bound

- Branch on y_i non-integral
- Solve NLPs & branch until
 - 1 Node infeasible: ●
 - 2 Node integer feasible: □
⇒ get upper bound (U)
 - 3 Lower bound $\geq U$: ▲

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 $\text{tol } \epsilon > 0$, set $U = \infty$, add $(\text{NLP}(-\infty, \infty))$ to heap $\mathcal{H}$.

while $\mathcal{H} \neq \emptyset$ **do**

 Remove $(\text{NLP}(l, u))$ from heap: $\mathcal{H} = \mathcal{H} - \{ \text{NLP}(l, u) \}$.

 Solve $(\text{NLP}(l, u)) \Rightarrow \text{solution } (x, y)^{(l, u)}$

if $(\text{NLP}(l, u))$ is infeasible **then**

 | Prune node: infeasible

else if $f((x, y)^{(l, u)}) > U$ **then**

 | Prune node; dominated by bound U

else if $y^{(l, u)}$ integral **then**

 | Update incumbent : $U = f((x, y)^{(l, u)})$, $(x, y)^* = (x, y)^{(l, u)}$.

else

 | BranchOnVariable($y_i^{(l, u)}$, $l, u, \mathcal{H}$)

Nonlinear Branch-and-Bound

BnB is finite, provided feasible set is bounded polyhedron:

Theorem (Finiteness of Nonlinear Branch-and-Bound)

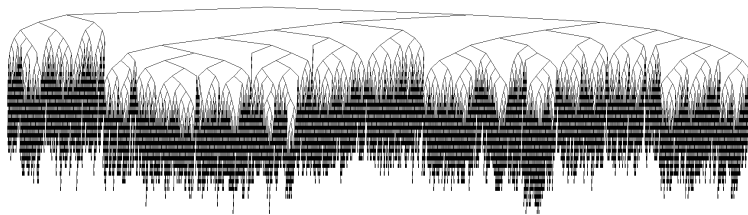
Solve MINLP by nonlinear branch-and-bound, and assume that A1-A3 hold. Then BnB terminates at optimal solution (or indication of infeasibility) after a finite number of nodes.

Proof.

- (A1-A3) $\Rightarrow$ every NLP solved globally (convex, MFCQ)
 - Boundedness of feasible set $\Rightarrow$ tree is finite
- $\Rightarrow$ convergence, see e.g. Theorem 24.1 of [Schrijver, 1986]. □

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!

Short Exercise

Exercise (Size of Branch-and-Bound Tree)

- ① Given $y_i \in \{0, 1\}^P$: what is the maximum depth of the branch-and-bound tree; what is the maximum size of the branch-and-bound tree?
- ② Given $y_i \in \{0, 1, \dots, K\}^P$: what is the maximum depth of the branch-and-bound tree; what is the maximum size of the branch-and-bound tree?
- ③ Suggest an alternative branching strategy for general integers $y_i \in \{0, 1, \dots, K\}^P$!

Branch-and-bound for convex mixed-integer program

- 1 Fundamental Building Blocks
- 2 Nonlinear Branch-and-Bound
- 3 Generic Nonlinear Branch-and-Cut**
- 4 LP/NLP-Based Branch-and-Bound
- 5 Summary and Take-Away Points: Branch-and-Cut

Generic Nonlinear Branch-and-Cut

Mixed-Integer Nonlinear Programming (MINLP) problem

$$\underset{x,y}{\text{minimize}} \ f(x,y) \quad \text{subject to} \ g(x,y) \leq 0, \quad y \in \mathbb{Z}^p$$

Next define generic branch-and-cut algorithm

- Combines branching and cut-generation
- Abstract framework ... will see detailed examples

Define NLP subproblem with added bounds ($l.u$)

$$(\text{NLP}(l.u)) \quad \left\{ \underset{x,y}{\text{minimize}} \ f(x,y) \quad \text{subject to} \ g(x,y) \leq 0, \quad l \leq y \leq u \right\}$$

Generic Nonlinear Branch-and-Cut

Choose a tol $\epsilon > 0$, set $U = \infty$, add $(\text{NLP}(-\infty, \infty))$ to heap $\mathcal{H}$.

while $\mathcal{H} \neq \emptyset$ **do**

 Remove $(\text{NLP}(l.u))$ from heap: $\mathcal{H} = \mathcal{H} - \{ (\text{NLP}(l.u)) \}$.

repeat

 Solve $(\text{NLP}(l.u)) \Rightarrow \text{solution } (x, y)^{(l,u)}$.

if $(\text{NLP}(l.u))$ is infeasible **then**

 | Prune node: infeasible

else if $f((x, y)^{(l,u)}) > U$ **then**

 | Prune node; dominated by bound U

else if $y^{(l,u)}$ integral **then**

 | Update incumbent: $U = f((x, y)^{(l,u)})$, $(x, y)^* = (x, y)^{(l,u)}$ & prune.

else $\text{GenerateCuts}((x, y)^{(l,u)}, j)$... details later ;

until *no new cuts generated or node pruned*;

if $(\text{NLP}(l.u))$ not pruned & not incumbent **then**

 | $\text{BranchOnVariable}(y^{(l,u)}, l, u, \mathcal{H})$

Branch-and-bound for convex mixed-integer program

- 1 Fundamental Building Blocks
- 2 Nonlinear Branch-and-Bound
- 3 Generic Nonlinear Branch-and-Cut
- 4 LP/NLP-Based Branch-and-Bound**
- 5 Summary and Take-Away Points: Branch-and-Cut

Solving convex MINLPs by using linear approximations

- Solving convex (nonlinear) optimization problems is expensive
- MILP-based branch-and-bound methods are super-fast because...
... It is very efficient to re-solve a Linear Programming (LP) problem after branching.

Is there any way to get the efficiency of MILP on convex MINLP?

Consider a slightly different definition of the problem:

$$\begin{array}{ll}\underset{x,y,\eta}{\text{minimize}} & \eta \\ \text{subject to} & \eta \geq f(x,y) \\ & 0 \geq g(x,y) \\ & y \in \mathbb{Z}^p\end{array}$$

Relaxations of Nonlinear Convex Constraints

Mixed-Integer Nonlinear Optimization (MINLP) problem

$$\underset{x,y,\eta}{\text{minimize}} \quad \eta \quad \text{subject to} \quad \eta \geq f(x,y), \quad 0 \geq g(x,y), \quad y \in \mathbb{Z}^p$$

Relaxing Convex Constraints

- Convex $0 \geq g(x,y)$ and $\eta \geq f(x,y)$ relaxed by supporting hyperplanes

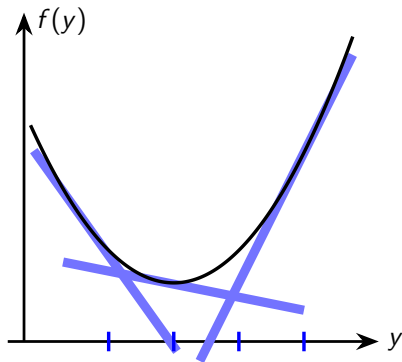
$$\eta \geq \hat{f} + \nabla \hat{f}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right)$$

$$0 \geq \hat{g} + \nabla \hat{g}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right)$$

for a set of point $(\hat{x}, \hat{y}) \in \mathcal{P}$

- Polyhedral relaxation of convex constraints.

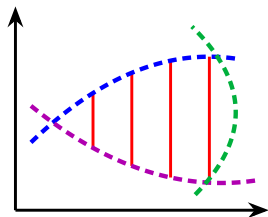
Cuts form the so-called *Outer Approximation* [Duran and Grossmann, 1986].



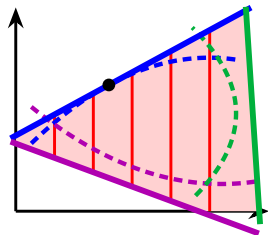
LP/NLP Branch and Bound

LP/NLP-based branch-and-bound

- Branch-and-cut algorithm with cuts from NLP solves
- Create MILP relaxation of MINLP



$$0 \geq g(x)$$



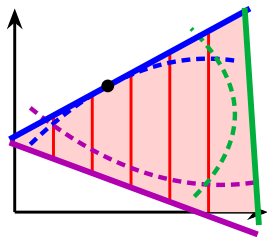
$$0 \geq g^{(k)} + \nabla g^{(k)T} (x - x^{(k)})$$

- Search MILP-tree $\Rightarrow$ faster re-solves
- Interrupt MILP tree-search to create new linearizations

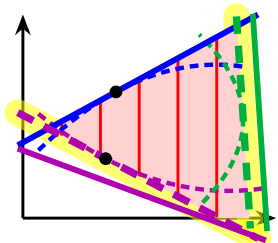
LP/NLP Branch and Bound

LP/NLP-based branch-and-bound

- Branch-and-cut algorithm with cuts from NLP solves
- Create MILP relaxation of MINLP & refine linearizations



$$0 \geq g(x)$$



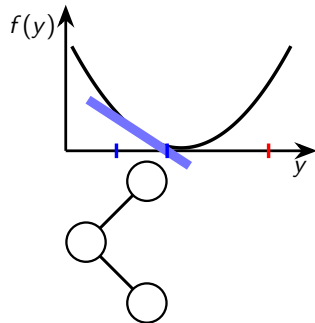
$$0 \geq g^{(k)} + \nabla g^{(k)T} (x - x^{(k)})$$

- Search MILP-tree $\Rightarrow$ faster re-solves
- Interrupt MILP tree-search to create new linearizations

LP/NLP-Based Branch-and-Bound

Aim: avoid **solving expensive MILPs**

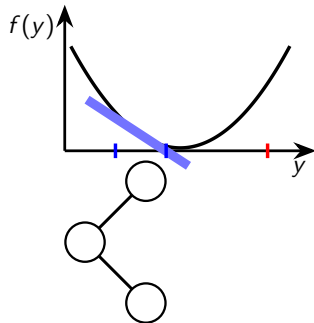
- Form MILP outer approximation



LP/NLP-Based Branch-and-Bound

Aim: avoid **solving expensive MILPs**

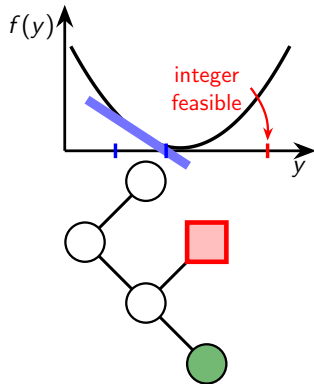
- Form MILP outer approximation
- Take initial MILP tree



LP/NLP-Based Branch-and-Bound

Aim: avoid **solving expensive MILPs**

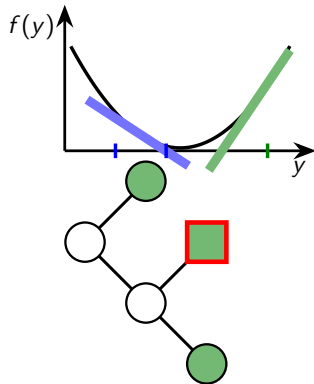
- Form MILP outer approximation
- Take initial MILP tree
- **interrupt MILP**, when new integral $x_i^{(j)}$ found
 $\Rightarrow$ solve $\text{NLP}(x_i^{(j)})$ get $x^{(j)}$



LP/NLP-Based Branch-and-Bound

Aim: avoid **solving expensive MILPs**

- Form MILP outer approximation
- Take initial MILP tree
- **interrupt MILP**, when new integral $x_i^{(j)}$ found
 $\Rightarrow$ solve $\text{NLP}(x_i^{(j)})$ get $x^{(j)}$
- linearize f, c about $x^{(j)}$
 $\Rightarrow$ **add linearization to tree**



LP/NLP-Based Branch-and-Bound

Aim: avoid **solving expensive MILPs**

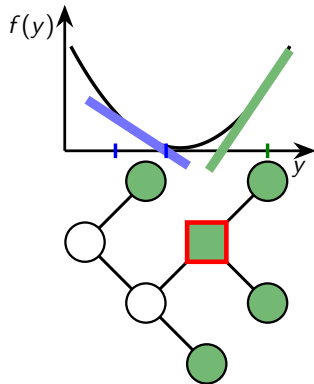
- Form MILP outer approximation
- Take initial MILP tree
- **interrupt MILP**, when new integral $x_I^{(j)}$ found
⇒ solve NLP($x_I^{(j)}$) get $x^{(j)}$
- linearize f, c about $x^{(j)}$
⇒ **add linearization to tree**
- **continue MILP** tree-search

... until lower bound $\geq$ upper bound

MINOTAUR: FilterSQP + own tree search [Mahajan et al., 2021]

FiLMINT: FilterSQP + MINTO [Abhishek et al., 2010]

BONMIN: IPOPT + CBC [IBM/CMU] also BB, OA [Bonami et al., 2008]



Useful notation to define LP/NLP-Based BnB

- $\mathcal{P}$ set of points $(\hat{x}, \hat{y})$ at which we have optimality or feasibility cuts:

$$\eta \geq \hat{f} + \nabla \hat{f}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right)$$

$$0 \geq \hat{g} + \nabla \hat{g}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right)$$

- $\mathcal{H}$ heap of open LP subproblems on tree
- R-MINLP: MINLP's continuous relaxation

LP/NLP-Based Branch-and-Bound

Subproblems solved in LP/NLP-Based BnB

- NLP($\hat{y}$) NLP with fixed integers

$$\text{NLP}(\hat{y}) \left\{ \begin{array}{ll} \underset{x}{\text{minimize}} & f(x, \hat{y}) \\ \text{subject to} & g(x, \hat{y}) \leq 0 \end{array} \right.$$

- LP($\mathcal{P}, l, u$) LP node on MILP tree

$$\text{LP}(\mathcal{P}, l, u) \left\{ \begin{array}{ll} \underset{x, y, \eta}{\text{minimize}} & \eta \\ \text{subject to} & \eta \geq \hat{f} + \nabla \hat{f}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right), \forall \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \in \mathcal{P} \\ & 0 \geq \hat{g} + \nabla \hat{g}^T \left(\begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \right), \forall \begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} \in \mathcal{P} \\ & l \leq y \leq u \end{array} \right.$$

LP/NLP-Based Branch-and-Bound

Set $U = \infty$, set $\mathcal{P} = \emptyset$, solve R-MINLP Add solution of R-MINLP $(\hat{x}, \hat{y})$ to $\mathcal{P} = \mathcal{P} \cup \{(\hat{x}, \hat{y})\}$ & add $(LP(\mathcal{P}, -\infty, \infty))$ to heap $\mathcal{H}$.

while $\mathcal{H} \neq \emptyset$ **do**

Remove $LP(\mathcal{P}, l, u)$ from heap: $\mathcal{H} = \mathcal{H} \setminus \{ LP(\mathcal{P}, l, u) \}$.

repeat

Solve $LP(\mathcal{P}, l, u)$ and let solution be $(\hat{\eta}, \hat{x}, \hat{y})$

if $LP(\mathcal{P}, l, u)$ infeasible or $\hat{\eta} > U$ **then**

 Prune node: infeasible or objective dominated

else if $\hat{y}$ integer feasible **then**

 Solve NLP($\hat{y}$) (or NLFO) $(x', y' = \hat{y})$ & add OA cuts: $\mathcal{P} = \mathcal{P} \cup \{(x', y')\}$

if NLP($\hat{y}$) feasible and $f' < U$ **then**

 New incumbent $(x^*, y^*) = (x', y')$

until no new OA cuts generated or node pruned;

if $LP(\mathcal{P}, l, u)$ not pruned **then**

 BranchOnVariable($\hat{y}, l, u, \mathcal{H}$)

Branch-and-Bound: Summary and Take-Away Points

Nonlinear Branch-and-Bound / Branch-and-Cut Summary

- Three fundamental building blocks for MINLP algorithms
 - ① Relaxation: generate a simpler problem, easy-to-solve
 - ② Constraint enforcement: modify simpler problem to get solution
 - ③ Upper Bounds: pruning search space for efficiency
- Branch-and-bound problem by branching globally for convex MINLPs
- Need careful implementation of
 - Branching variable & node selection
 - Software infrastructure ... build on other frameworks!

Presolve: Improving Coefficients: An Example

$$(1) \quad x_1 + 21x_2 \leq 30$$

$$0 \leq x_1 \leq 14$$

$$x_2 \in \{0, 1\}$$

Presolve: Improving Coefficients: An Example

$$(1) \quad x_1 + 21x_2 \leq 30$$

$$0 \leq x_1 \leq 14$$

$$x_2 \in \{0, 1\}$$

$$\boxed{\text{If } x_2 = 0}$$

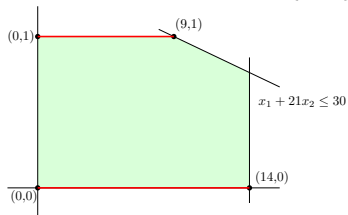
$$x_1 + 0 \leq 30$$

(1) is loose.

$$\boxed{\text{If } x_2 = 1}$$

$$x_1 \leq 9$$

(1) is tight.

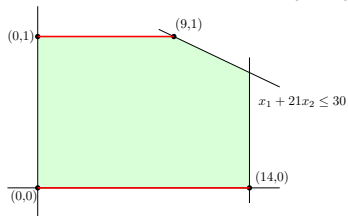


Presolve: Improving Coefficients: An Example

$$(1) \quad x_1 + 21x_2 \leq 30$$

$$0 \leq x_1 \leq 14$$

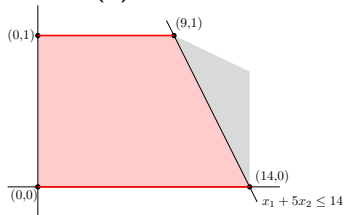
$$x_2 \in \{0, 1\}$$



$$\boxed{\text{If } x_2 = 0}$$

$$x_1 + 0 \leq 30$$

(1) is loose.



$$\boxed{\text{If } x_2 = 1}$$

$$x_1 \leq 9$$

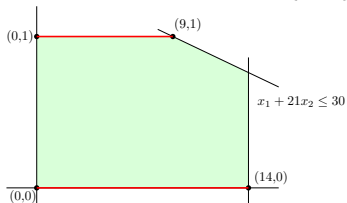
(1) is tight.

Presolve: Improving Coefficients: An Example

$$(1) \quad x_1 + 21x_2 \leq 30$$

$$0 \leq x_1 \leq 14$$

$$x_2 \in \{0, 1\}$$

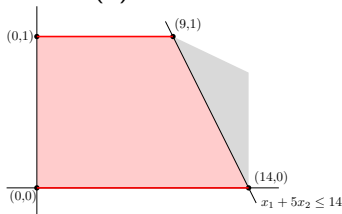


Reformulation:

$$(2) \quad x_1 + 5x_2 \leq 14$$

$$0 \leq x_1 \leq 14$$

$$x_2 \in \{0, 1\}$$



$$\text{If } x_2 = 0$$

$$x_1 + 0 \leq 30$$

(1) is loose.

$$\text{If } x_2 = 1$$

$$x_1 \leq 9$$

(1) is tight.

$$\text{If } x_2 = 0$$

$$x_1 + 0 \leq 14$$

(2) is tight.

$$\text{If } x_2 = 1$$

$$x_1 \leq 9$$

(2) is tight.

(1) and (2) equivalent. But relaxation of (2) is tighter.

Presolve: Improving Coefficients: Linear to Nonlinear

$$\begin{aligned}c(x_1, x_2, \dots, x_k) &\leq M(1 - y) \\ l_i &\leq x_i \leq u_i, \quad i = 1, \dots, k \\ y &\in \{0, 1\}\end{aligned}$$

- If $c(x_1, x_2, \dots, x_k) \leq M(1 - 0)$, is loose, tighten it!

$$\begin{aligned}\text{Let } c^u &= \max_x c(x_1, \dots, x_k) && \text{(MAX-c)} \\ \text{s.t. } &l_i \leq x_i \leq u_i, \quad i = 1, \dots, k\end{aligned}$$

- If $c^u < M$, then tighten: $c(x_1, \dots, x_k) \leq c^u(1 - y)$
- (MAX-c) is a nonconvex NLP ... time-consuming
- Upper bound on (MAX-c) will also tighten
- Trade-off between time and quality of bound: Fast or Tight!

Improving Coefficients: Using Implications

$$c(x_1, x_2, \dots, x_k) \leq M(1 - y),$$

$$l_i \leq x_i \leq u_i, \quad i = 1, \dots, k,$$

$$y \in \{0, 1\}.$$

- Often, y , x_i also occur in other constraints of MINLP. e.g.

$$c(x_1, x_2, \dots, x_k) \leq M(1 - y)$$

$$0 \leq x_1 \leq M_1 y$$

$$0 \leq x_2 \leq M_2 y$$

...

$$y \in \{0, 1\}$$

- $y = 0 \Rightarrow x_1 = x_2, \dots = x_k = 0$. (Implications)
- If $c(0, \dots, 0) < M$, then we can tighten.
- No need to solve (MAX-c). Fast and Tight.

Presolve for MINLP

Advanced functions of presolve (Reformulating):

- Improve coefficients.
- Disaggregate constraints.
- Derive implications and conflicts.

Basic functions of presolve (Housekeeping):

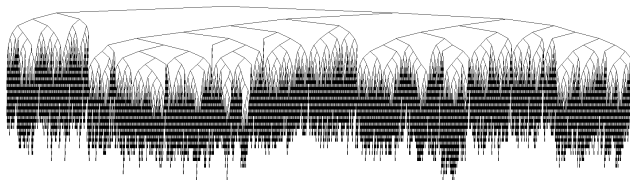
- Tighten bounds on variables and constraints.
- Fix/remove variables.
- Identify and remove redundant constraints.
- Check duplicacy.

Popular in Mixed-Integer Linear Optimization [[Savelsbergh, 1994](#)]

Presolve for MINLP: Computational Results

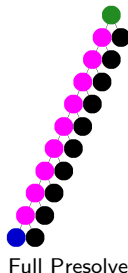
Syn20M04M from `egon.cheme.cmu.edu`

	No Presolve	Basic Presolve	Full Presolve
Variables:	420	328	292
Binary Vars:	160	144	144
Constraints:	1052	718	610
Nonlin. Constr:	56	56	56
Bonmin(sec):	>7200	NA	NA
Minotaur(sec):	>7200	>7200	2.3

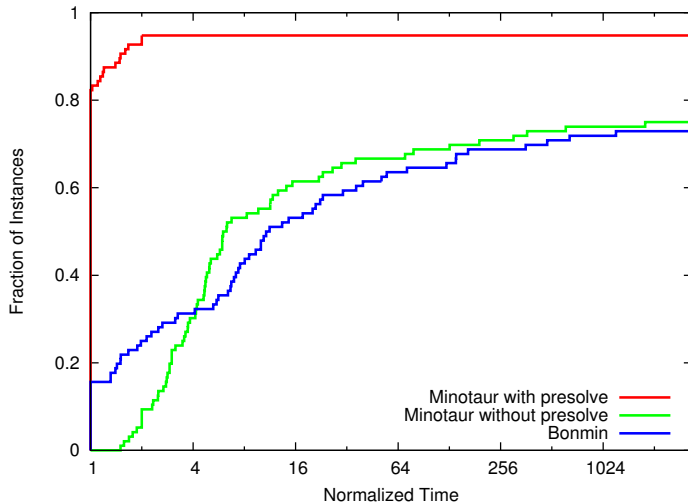


Minotaur, no presolve: 10000+ nodes after solving for 360s

Why does no one else do this?



Presolve for MINLP: Results



Time for B&B on 96 RSyn-X and Syn-X instances.



Abhishek, K., Leyffer, S., and Linderoth, J. T. (2010).

FilMINT: An outer-approximation-based solver for nonlinear mixed integer programs.

INFORMS Journal on Computing, 22:555–567.

DOI:10.1287/ijoc.1090.0373.



Bonami, P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., and Wächter, A. (2008).

An algorithmic framework for convex mixed integer nonlinear programs.

Discrete Optimization, 5(2):186–204.



Duran, M. A. and Grossmann, I. (1986).

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

Mathematical Programming, 36:307–339.



Mahajan, A., Leyffer, S., Linderoth, J., Luedtke, J., and Munson, T. (2021).

Minotaur: A mixed-integer nonlinear optimization toolkit.

Mathematical Programming Computation, 13(2):301–338.



Savelsbergh, M. W. P. (1994).

Preprocessing and probing techniques for mixed integer programming problems.

ORSA Journal on Computing, 6:445–454.



Schrijver, A. (1986).

Theory of Linear and Integer Programming.

Wiley, New York.