Solving Mixed-Integer Nonlinear Optimization Problems Using MINOTAUR

Mustafa Vora, Meenarli Sharma, Prashant Palkar and Ashutosh Mahajan

Industrial Engineering and Operations Research,
Indian Institute of Technology Bombay, India

52nd Annual Convention of ORSI & International Conference
Preconference Workshop
IIM Ahmedabad, December 15, 2019

Outline

Introduction to MINLPs

Algorithms and Solvers for MINLPs

MINOTAUR Solver

Important Algorithmic Components

Exercise I: Portfolio Optimization (a Convex MINLP Example)

Exercise II: Packing Circles in a Triangle (a Nonconvex MINLP Example)

Setting up Your Computer

Follow these steps to install Minotaur binaries with AMPL

1. If you do not have AMPL IDE, download the free demo version:
   - Windows
     `https://ampl.com/try-ampl/download-a-free-demo/#windows`
   - Linux
     `https://ampl.com/try-ampl/download-a-free-demo/#linux`
   - Follow the instructions on the AMPL website to unzip the files

2. Download Minotaur files
   - Windows
     `http://www.ieor.iitb.ac.in/files/minotaur-win.zip`
   - Linux
     `http://www.ieor.iitb.ac.in/files/minotaur-linux.zip`

Setting up Your Computer

- Unzip Minotaur files
- All files (bnb, mcqg, all .mod files, etc.) in the folder should be copied to AMPL directory
- AMPL directory is the one that contains ampl.lic file and other AMPL files
- Open file manager (Windows explorer) and go to AMPL directory
- Open the amplide folder and start `amplide`
- From the left panel, change the 'Current Directory' to the folder containing ampl.lic and all MINOTAUR files
- Double click on test.mod and run it (ctrl+r)

Outline

Introduction to MINLPs

Algorithms and Solvers for MINLPs

MINOTAUR Solver

Important Algorithmic Components

Exercise I: Portfolio Optimization (a Convex MINLP Example)

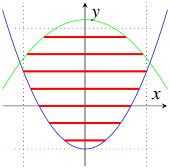
Exercise II: Packing Circles in a Triangle (a Nonconvex MINLP Example)

Mixed-Integer Nonlinear Programs (MINLPs)

An optimization problem of the form

$$\begin{aligned} \min_{x,y} \; & f(x,y) \\ \text{s.t.} \;\; & c(x,y) \le 0, \\ & (x,y) \in X \subset \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}, \end{aligned} \tag{P}$$

where the functions $f : \mathbb{R}^n \to \mathbb{R}$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ are typically nonlinear, x and y are continuous and integer constrained, respectively, decision variables, and X is bounded integral-polyhedral set.



- MILP (NP-hard, *Kannan and Monma, 1978*), nonconvex NLP (untractable, *Jeroslow, 1973*) are special cases.
- If feasible region is convex on relaxing integrality, then we call (P) **convex MINLP**.

Applications and Research Areas

Applications

- Cutting stock, portfolio optimization, facility layout, process design, unit commitment, water and gas networks etc.
- others: cybersecurity, brachytherapy, energy management, statistics, cloud, supercomputers, environment, weapons target assignment etc.

Academic Research

- Algorithms, relaxations, cuts, branchers, heuristics, presolving, structure exploitation, etc.
- others: representability, parallelism, overlaps with new areas: DFO, PDEs, ML, bilevel etc.

IEOR IIT BOMBAY

Outline

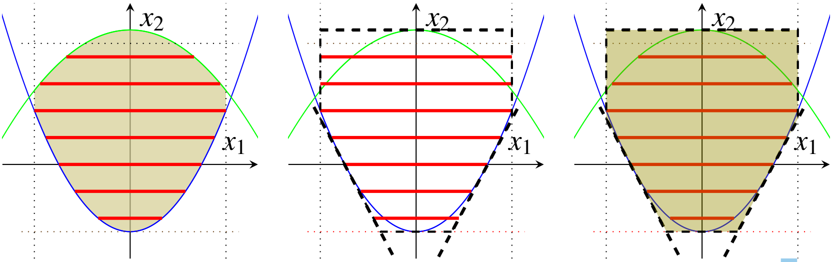
Algorithms for MINLPs

Basic Idea

- get lower bound (L) on optimal value using *tractable* relaxations of (P)
- get upper bound (U) on optimal value using feasible solutions of (P)
- improve both bounds until the sequences converge

Type of Relaxations

- NLP (relax integrality), MILP (relax nonlinearity), LP (relax both)



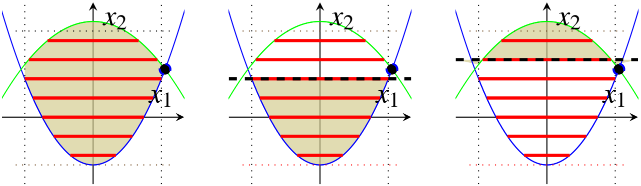
- Other: semidefinite, second-order cones etc. (*Lubin et al, 2017, 2019*)

Algorithms

- Nonlinear Branch-and-Bound
- Extended Cutting Plane
- Outer Approximation, Generalized Bender's Decomposition
- LP/NLP based Branch-and-Bound, Extended Supporting Hyperplane
- Spatial Branch-and-Bound for nonconvex MINLPs

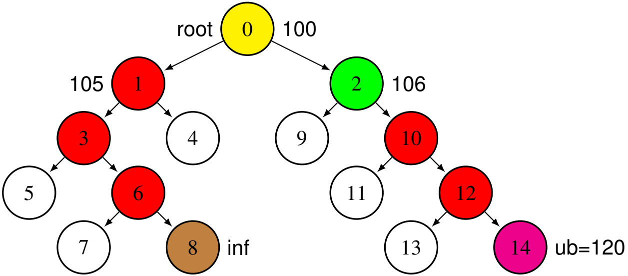
Nonlinear Branch-and-Bound (NLP-BB)

- Form the NLP relaxation of (P) by relaxing integrality on y variables
- If the solution of NLP is integer feasible, update the upper bound U
- Otherwise, branch on some $y_j \notin \mathbb{Z}$ and create new subproblems.
- Solve the subproblems, update U when feasible solutions are obtained and *prune* infeasible or bound-inferior subproblems.
- Continue until the bounds converge or all subproblems exhausted.



Nonlinear Branch-and-Bound (NLP-BB)

- Form the NLP relaxation of (P) by relaxing integrality on y variables
- If the solution of NLP is integer feasible, update the upper bound U
- Otherwise, branch on some $y_j \notin \mathbb{Z}$ and create new subproblems.
- Solve the subproblems, update U when feasible solutions are obtained and *prune* infeasible or bound-inferior subproblems.
- Continue until the bounds converge or all subproblems exhausted.



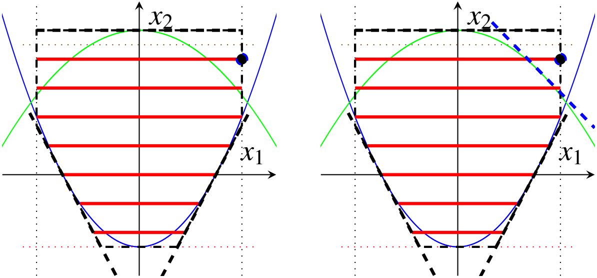
Outer Approximation (OA)

Alternating sequence of NLP/MILP solving (multi-tree)

- Solve the NLP relaxation of (P) and at its optimal $(\hat{x}, \hat{y})$, generate linearizations for all nonlinear constraints

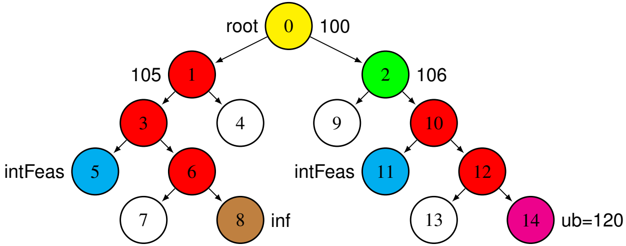
$$c_k(\hat{x}, \hat{y}) + ((x, y) - (\hat{x}, \hat{y}))^T \nabla c_k(\hat{x}, \hat{y}) \leq 0, \tag{1}$$

- Solve MILP relaxation. If infeasible, STOP, else update L, obtain $(\bar{x}, \bar{y})$
- Solve an NLP by fixing, $y = \bar{y}$, obtain $(\hat{x}, \hat{y})$
- Update U if NLP is feasible. Add linearization cuts 1 at $(\hat{x}, \hat{y})$ to MILP
- Repeat NLP/MILP solving until bounds converge or (P) infeasible



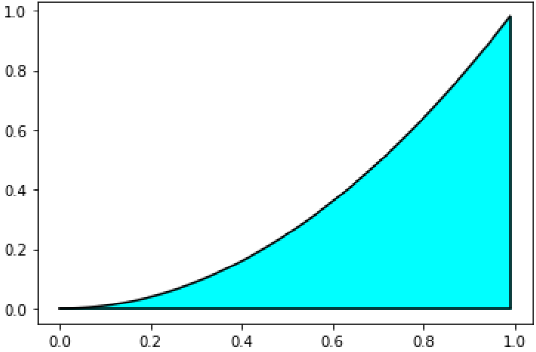
LP/NLP based Branch-and-Bound (QG)

- MILP solving is expensive!
- In OA, consecutive MILPs differ in only a few linearization constraints!
- Improvise OA: avoid multiple MILP solves from scratch (*Quesada and Grossmann, 1992*)
- Maintain a *single MILP tree*, add linearizations to open nodes when integer solution is obtained



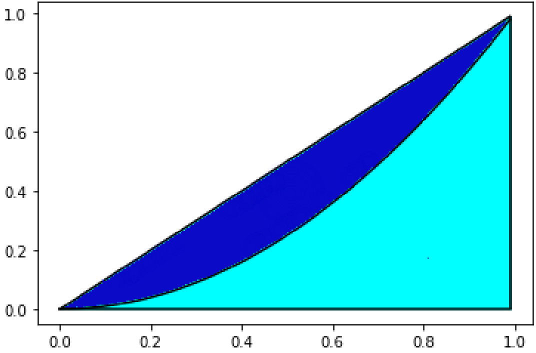
Spatial Branch-and-Bound

- For nonconvex problems, relaxing variable integrality does not give convex relaxation
- Example: a nonconvex region defined as $y \leq x^2, 0 \leq x \leq 1$



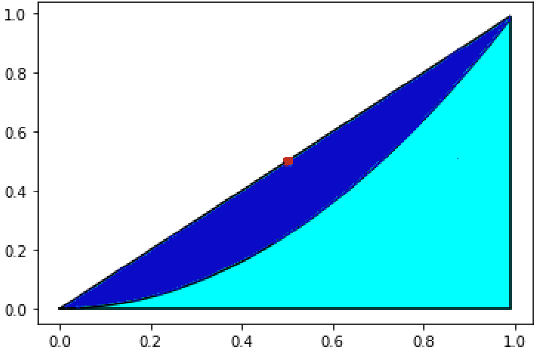
Spatial Branch-and-Bound

- For nonconvex problems, relaxing variable integrality does not give convex relaxation
- Example: a nonconvex region defined as $y \leq x^2, 0 \leq x \leq 1$
- Add an overestimator to get a linear relaxation



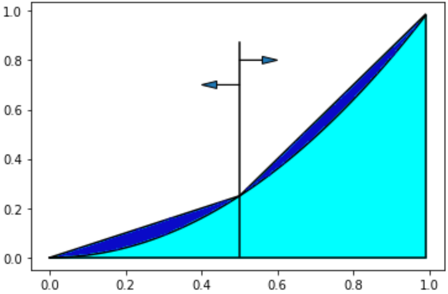
Spatial Branch-and-Bound

- Let linear relaxation solution be $(0.5, 0.5)$ (not feasible to the original problem)



Spatial Branch-and-Bound

- Let linear relaxation solution be $(0.5, 0.5)$ (not feasible to the original problem)
- Branch on the continuous variable x - one branch is $x \leq 0.5$ and the other branch is $x \geq 0.5$ - to obtain two subproblems
- Perform the same steps on each subproblems to refine relaxation



Solvers for Convex MINLPs

Convex

- NLP-BB: BONMIN, MINOTAUR, etc.
- OA: FilMINT, BONMIN, Muriqui, SHOT
- QG: BONMIN, MINOTAUR

Nonconvex

- Spatial BB: BARON, SCIP, MINOTAUR, etc.

Outline

MINOTAUR Toolkit (Mahajan et al, 2011)

Mixed
I nteger
N onlinear
O ptimization
T oolkit:
A lgorithms,
U nderestimators,
R elaxations.

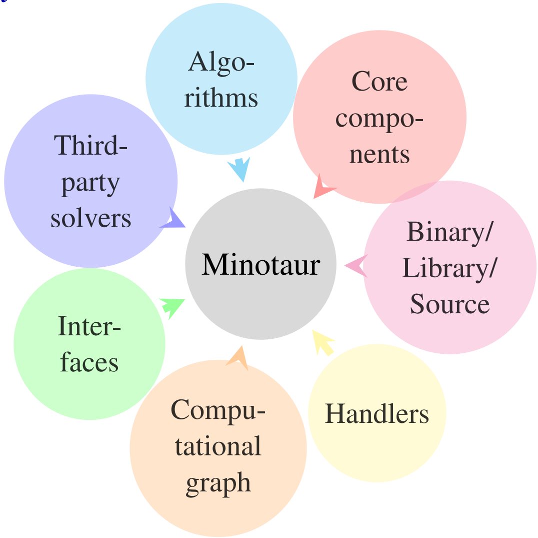
Goals:

- Fast, usable MINLP solver.
- Flexibility for modifying existing and Ease of developing new algorithms.

- > 55k lines of code excluding unit tests and examples
- Open source: https://github.com/minotaur-solver/minotaur.git

Convex MINLP Solvers	Global Optimization Solvers
NLP-BB (bnb)	QCQP global optimizer (glob)
LP/NLP QG (qg, mcqg)	Multistart NLP-BB Heuristic
OA (oa)	
QP Diving	

In a Nutshell



Developers: Argonne National Laboratory, University of Wisconsin-Madison, USA and IIT Bombay, India

MINOTAUR: Building Blocks

Core Components

- Problem Description Classes
  - Function
  - NonlinearFunction
  - LinearFunction
  - Variable, Constraint, Objective
- Branch-and-Bound Classes
  - NodeRelaxer, NodeProcessor
  - Brancher, TreeManager
  - Presolver, CutManager, etc.
- Structure Handlers
  - Linear, SOS2, CxUnivar, CxQuad, Multilinear etc.
  - QG, Perspective, Separability etc.
- Utility Classes
  - Timer, Options, Logger, Containers, Operations, etc.

Engines

LP

- CLP
- CPLEX

NLP

- Filter-SQP
- IPOPT
- BQPD

MILP

- CBC
- CPLEX

Interfaces

- AMPL
- C++

Outline

Introduction to MINLPs

Algorithms and Solvers for MINLPs

MINOTAUR Solver

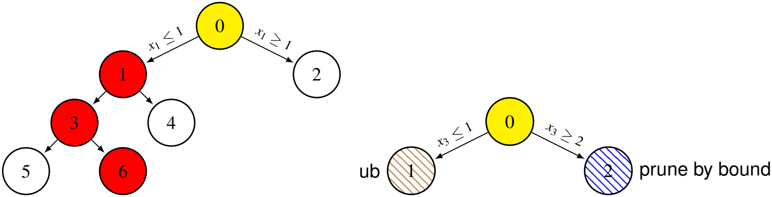
Important Algorithmic Components

Exercise I: Portfolio Optimization (a Convex MINLP Example)

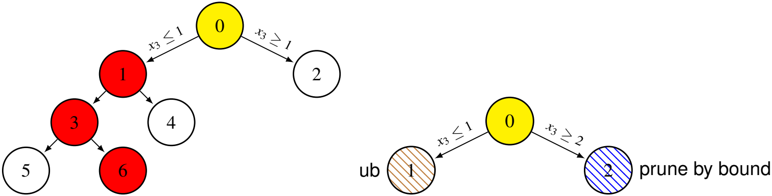
Exercise II: Packing Circles in a Triangle (a Nonconvex MINLP Example)

Important Algorithmic Components

- Branching: why important?



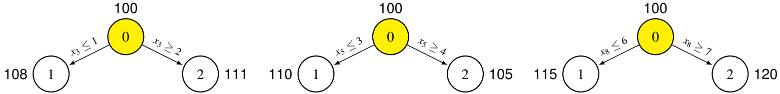
- Node selection: why important?



- Cuts: tighter relaxations, hence better lower bounds

Branching schemes

- Lexicographic: choose *candidate* with smallest index (no info used)
- Maximum violation: choose *most fractional* candidate (not successful)
  - $x_1 = 0.9,$ score $= 0.1(0.8) + 0.9 * (0.2) = 0.26$
  - $x_6 = 0.4,$ score $= 0.4(0.8) + 0.6 * (0.2) = \boxed{0.44}$
- *Strong Branching*: use bound change (expensive)



- *Pseudocost Branching*: use bound change
  - Maintain scores (up/down) for each variable based on bound change
  - Scores not representative initially
- ***Reliability Branching*** (most practical)
  - Hybrid of strong and pseudocost branching
  - Classify variables as reliable and unreliable
  - Strong branch on unreliable candidates (make them reliable), then maintain scores

More About MINOTAUR @ ORSI2019

1. "Linearization Schemes for LP/NLP Based Branch and Bound Algorithm for Convex MINLPs" on Monday, Dec 16, 12:00-1:30 PM, by Meenarli Sharma, Session MC2

2. "Accelerating LP, NLP, and MILP Based Algorithms for Convex MINLPs using Parallelization Schemes" on Monday, Dec 16, 2:30-4:00 PM, by Prashant Palkar, Session MD2

Outline

Portfolio Optimization Problem

Given:

- a set $\mathcal{A}$ of r risky assets with expected return $\mu_j, j \in \mathcal{A}$, and one nonrisky asset with return μ_0
- variance-covariance matrix $C \in \mathbb{R}^{r \times r}$

Find the investment in each asset which minimizes the risk (variance), such that,

- entire budget is invested
- a prespecified return level R is achieved
- if an asset is invested in, a minimum investment w_{min} is made

Mathematical Formulation

- Set: $\mathcal{A}$, Parameters: $R, C, \mu_j, j \in \mathcal{A}, \mu_0, w_{min}$
- Decision variables
  - w_0 in the nonrisky asset
  - w_j: investment in risky asset $j \in \mathcal{A}$
  - z_j: a binary variable, $= 1$ if we invest in asset j, otherwise 0.

Let w be $[w_1, w_2, \ldots w_r]^T$.

$$
\begin{aligned}
\min_{w_0, w, z} \quad & w^T C w \\
\text{s.t.} \quad & w_0 + \sum_j w_j = 1, \\
& \mu_0 \, w_0 + \sum_j \mu_j \, w_j \geq R, \\
& w_j \geq w_{min} \, z_j, \\
& w_j \leq z_j, \\
& z_j \in \{0, 1\}, \\
& w_j \in \mathbb{R}_+, \ \forall j \in \mathcal{A}.
\end{aligned} \tag{Ex-1}
$$

AMPL Syntax

Enter

- model file name
  `model exampleFileName.mod;`
- data file name
  `data exampleFileName.dat;`
- solver name, say `bnb`
  `option solver bnb;`
- solver options
  `option bnb_options '--bnb_time_limit 10';`
- solve
  `solve;`
- display output `display _varname, _var;`

A Few MINOTAUR Options

Option	Default Value	Possible values
`show_options`	0	0,1
`log_level`	2	0-3 (integer)
`presolve`	1	0,1
`display_problem`	0	0,1
`display_presolved_problem`	0	0,1
`brancher`	rel	rel, maxvio, lex
`tree_search`	BthenD	dfs, bfs, BthenD
`bnb_node_limit`	1e+9	> 0 (integer)
`bnb_time_limit`	1e+20	> 0 (in sec)
`cgtoqf`	0	0,1
`nlp_engine`	FilterSQP	IPOPT, FilterSQP
`threads`	1	1-# processors (int.)

Hands-on

- Following instances are available
  - `portfolM`
  - `portfol_buyin`
  - `portfol_roundlot`
  - `portfol_classical050_1`
- Recommended tests with NLP engine IPOPT and time limit 180s:
  - Solve `portfolM` using bnb and qg
  - qg with various branchers on `portfol_classical050_1`
  - bnb with different tree search strategies on `portfol_roundlot`
  - mcqg with multiple threads on `portfol_classical050_1`
- Observe the following statistics in each run.
  - number of cuts added
  - number of nodes processed
  - time taken in LP and NLP solving

Outline

Introduction to MINLPs

Algorithms and Solvers for MINLPs

MINOTAUR Solver

Important Algorithmic Components

Exercise I: Portfolio Optimization (a Convex MINLP Example)

Exercise II: Packing Circles in a Triangle (a Nonconvex MINLP Example)

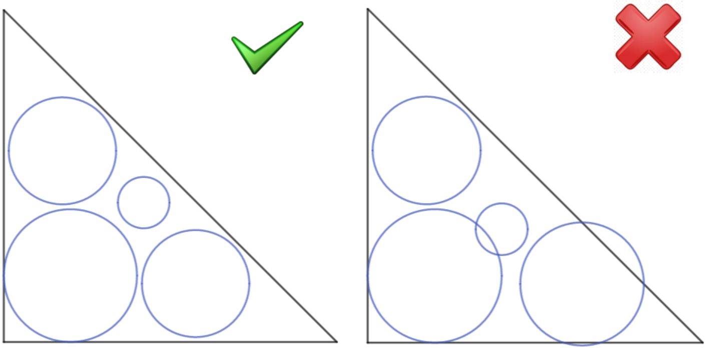
Packing Circles in a Triangle

Given:

- a set $\mathcal{S}$ of circles with radii $r_k, k \in \mathcal{S}$
- a right isosceles triangle with base length l

Find the maximum number of circles, such that:

- no two selected circles should overlap
- all selected circles should remain entirely within the triangle



Mathematical Formulation

- Set: $\mathcal{S}$, Parameters: $r_k \in \mathcal{S}, l$, M a large number
- Decision variables
  - x_k: x-coordinate of the centre of circle $k \in \mathcal{S}$
  - y_k: y-coordinate of the centre of circle $k \in \mathcal{S}$
  - z_k: binary variable, $= 1$ if circle $k \in \mathcal{S}$ is selected, otherwise 0

$$
\begin{aligned}
\max_{x,y,z} \ & \sum_{k \in S} z_k \\
\text{s.t.} \ & x_k \geq \ r_k, \ k \in \mathcal{S} \\
& y_k \geq \ r_k, \ k \in \mathcal{S} \\
& x_k + y_k \leq \ l - \sqrt{2} r_k, \ k \in \mathcal{S} \\
& (x_i - x_j)^2 + (y_i - y_j)^2 + M(2 - z_i - z_j) \geq \ (r_i + r_j)^2, \ i,j \in \mathcal{S}, \ i < j \\
& z_k \in \{0,1\}, \ x_k, y_k \in \mathbb{R} \ \forall \, k \in \mathcal{S},
\end{aligned}
\tag{Ex-1}
$$

Hands-on

- Solve following instance using: glob
  - `packing`
- Try the option `cgtoqf` and observe
  - # of nodes processed
  - time taken in solving
- Now change the `packing.dat` file and add two more circles of radii 2.3, 1.2 and increase the side length of triangle to 8.
- Run again and observe the change from the previous instance

THANK YOU.

For any discussions/questions, please contact:

- Ashutosh Mahajan (amahajan@iitb.ac.in)
- Meenarli Sharma (meenarli@iitb.ac.in)
- Prashant Palkar (prashant.palkar@iitb.ac.in)
- Mustafa Vora (mustafa.vora@iitb.ac.in)