



*A new MINLP Solver*

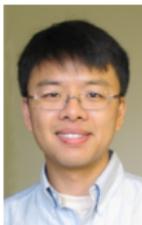
Ashutosh Mahajan and Team Minotaur

Mathematics and Computer Science Division  
Argonne National Laboratory

2011 DOE Applied Mathematics Program Meeting  
Washington, DC  
October 18, 2011.



Todd Munson



Zhen Xie



Ashutosh Mahajan



Sven Leyffer



Argonne  
NATIONAL  
LABORATORY



Andrew Miller



Jeff Linderth



THE UNIVERSITY  
of  
**WISCONSIN**  
MADISON



Mahdi Namazifar



Hyemin Jeon



Mustafa Kilinc



Jim Luedtke



Mahdi Hamzeei



## Mixed-Integer Nonlinear Optimization

$$\begin{aligned} & \min_x f(x) \\ & \text{s.t.} \quad g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ & \quad \quad x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned} \quad (\text{MINLP})$$

where  $f, g_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, 2, \dots, m$ .

## Mixed-Integer Nonlinear Optimization

$$\begin{aligned} \min_x \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned} \quad (\text{MINLP})$$

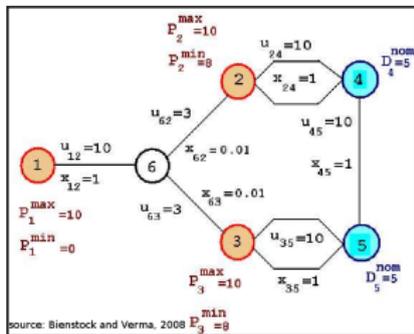
where  $f, g_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, 2, \dots, m$ .

- When  $f$  is a convex quadratic function and  $g_i$  are linear, – **Convex MIQP**.
- If  $f, g_i$  are convex and twice differentiable, – **Convex MINLP**.
- If  $p = 0$ , – **Global Optimization**.
- In general,  $f, g_i$  may be black-box, non-differentiable functions.
- For this talk, **we assume**  $f, g_i$  are
  - twice differentiable at points of interest, and,
  - “factorable”.



# MINLP Applications

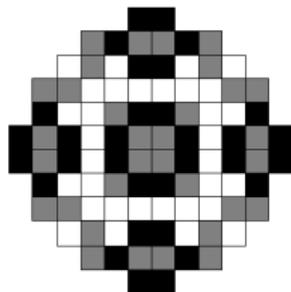
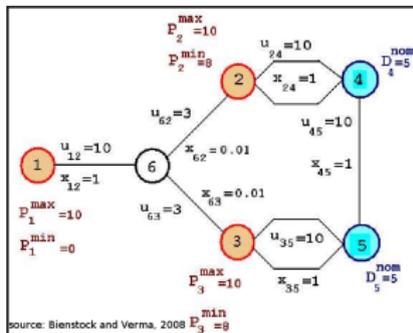
- Power Transmission
  - Optimal Power Flow
  - Network Expansion
  - Contingency Analysis





# MINLP Applications

- Power Transmission
  - Optimal Power Flow
  - Network Expansion
  - Contingency Analysis
- Power Generation
  - Nuclear Core Reloading
  - Design of Cogeneration Plants

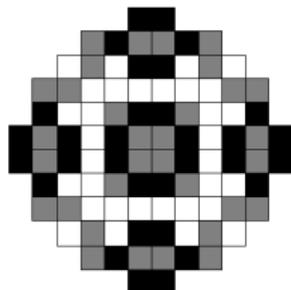
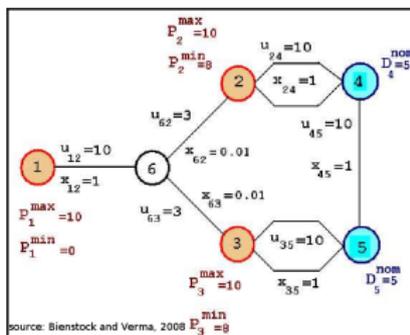


source: Terlaky et al., 1994



# MINLP Applications

- Power Transmission
  - Optimal Power Flow
  - Network Expansion
  - Contingency Analysis
- Power Generation
  - Nuclear Core Reloading
  - Design of Cogeneration Plants
- Chemical Reactors
  - Design and Redesign
  - Blending and Pooling

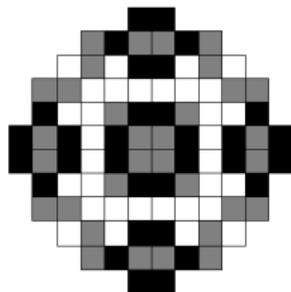
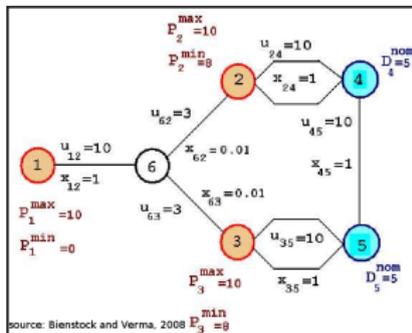


source: Terlaky et al., 1994

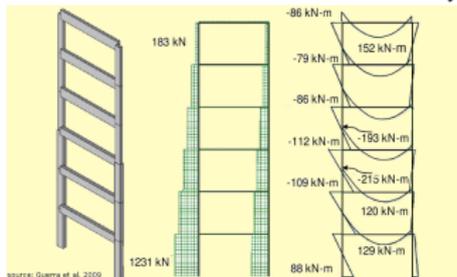


# MINLP Applications

- Power Transmission
  - Optimal Power Flow
  - Network Expansion
  - Contingency Analysis
- Power Generation
  - Nuclear Core Reloading
  - Design of Cogeneration Plants
- Chemical Reactors
  - Design and Redesign
  - Blending and Pooling
- Infrastructure
  - Water Distribution
  - Concrete Buildings
  - Open-Pit Mining
  - Wireless Networks
- Biology, Logistics, Finance, ...



source: Terlaky et al., 1994



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

Goals:

- **Fast**, usable MINLP solvers.
- Callable library for use in applications.
- **Ease** of developing new algorithms.



## The Story So Far

- 1 Implemented 4 solvers.
- 2 > 49k lines of code excluding unit tests and examples.
- 3 > 26k cpu-hours of testing.

 *Features*

---

	Bonmin	FilMINT	BARON	Couenne	Minotaur
--	--------	---------	-------	---------	----------

---

**Algorithms:**

NLP B&B	✓	×	×	×	✓
Quesada-Grossmann	✓	✓	×	×	✓
Branch-and-Reduce	×	×	✓	✓	✓

 *Features*

---

	Bonmin	FilMINT	BARON	Couenne	Minotaur
--	--------	---------	-------	---------	----------

---

**Algorithms:**

NLP B&B	✓	×	×	×	✓
Quesada-Grossmann	✓	✓	×	×	✓
Branch-and-Reduce	×	×	✓	✓	✓

**Support for Nonlinear Functions:**

Parse Comp. Graph	×	×	✓	✓	✓
Nonlinear Reform.	×	×	×	×	✓
Native Derivat.	×	×	×	×	✓

---

	Bonmin	FilMINT	BARON	Couenne	Minotaur
--	--------	---------	-------	---------	----------

---

**Algorithms:**

NLP B&B	✓	×	×	×	✓
Quesada-Grossmann	✓	✓	×	×	✓
Branch-and-Reduce	×	×	✓	✓	✓

**Support for Nonlinear Functions:**

Parse Comp. Graph	×	×	✓	✓	✓
Nonlinear Reform.	×	×	×	×	✓
Native Derivat.	×	×	×	×	✓

**Interfaces:**

AIMMS	×	×	✓	×	×
AMPL	✓	✓	×	✓	✓
GAMS	✓	×	✓	✓	×

 *Features*

	Bonmin	FilMINT	BARON	Couenne	Minotaur
<b>Algorithms:</b>					
NLP B&B	✓	×	×	×	✓
Quesada-Grossmann	✓	✓	×	×	✓
Branch-and-Reduce	×	×	✓	✓	✓
<b>Support for Nonlinear Functions:</b>					
Parse Comp. Graph	×	×	✓	✓	✓
Nonlinear Reform.	×	×	×	×	✓
Native Derivat.	×	×	×	×	✓
<b>Interfaces:</b>					
AIMMS	×	×	✓	×	×
AMPL	✓	✓	×	✓	✓
GAMS	✓	×	✓	✓	×
Open Source	✓	×	×	✓	✓



# *Basic Algorithm: Branch-and-Bound*

RELAX

BOUND

BRANCH

REPEAT



# *Basic Algorithm: Branch-and-Bound*

## RELAX

- Relax integrality restrictions.
- Relax nonconvex constraints.
  - Linear: Secant approx., McCormick approx., ..., BARON, Couenne.
  - Quadratic:  $\alpha$ -BB.
  - ...

## BOUND

## BRANCH

## REPEAT



## *Basic Algorithm: Branch-and-Bound*

### RELAX

- Relax integrality restrictions.
- Relax nonconvex constraints.
  - Linear: Secant approx., McCormick approx., ..., BARON, Couenne.
  - Quadratic:  $\alpha$ -BB.
  - ...

### BOUND

- Solve using an LP, QP, NLP or SDP solver. Obtain a lower bound.

### BRANCH

### REPEAT

# *Basic Algorithm: Branch-and-Bound*

## RELAX

- Relax integrality restrictions.
- Relax nonconvex constraints.
  - Linear: Secant approx., McCormick approx., ..., BARON, Couenne.
  - Quadratic:  $\alpha$ -BB.
  - ...

## BOUND

- Solve using an LP, QP, NLP or SDP solver. Obtain a lower bound.

## BRANCH

- If solution of relaxation satisfies (MINLP), update  $ub$ ,
- Otherwise, partition the feasible region into two (or more) parts.

## REPEAT

# *Basic Algorithm: Branch-and-Bound*

## RELAX

- Relax integrality restrictions.
- Relax nonconvex constraints.
  - Linear: Secant approx., McCormick approx., ..., BARON, Couenne.
  - Quadratic:  $\alpha$ -BB.
  - ...

## BOUND

- Solve using an LP, QP, NLP or SDP solver. Obtain a lower bound.

## BRANCH

- If solution of relaxation satisfies (MINLP), update  $ub$ ,
- Otherwise, partition the feasible region into two (or more) parts.

## REPEAT

- Repeat RELAX-BOUND-BRANCH on each part **recursively**,
- until  $\frac{ub-lb}{|ub|} \leq \epsilon$  for a given  $\epsilon \in \mathbb{R}$ .

# *Four Main Components of Toolkit*

*Interfaces* for reading input

- AMPL

*Engines* to solve LP/NLP/QP

- Bqpdp
- Filter-SQP
- Ipopt
- OSI-CLP

*Algorithms* to solve MINLP

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

*Base*

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

# *Four Main Components of Toolkit*

*Interfaces* for reading input

- AMPL
- **Your Interface Here**

*Engines* to solve LP/NLP/QP

- Bqpdp
- Filter-SQP
- Ipopt
- OSI-CLP
- **Your engine here**

*Algorithms* to solve MINLP

- Branch-and-Bound
- Outer-Approximation
- Quesada-Grossmann
- Branch-and-Reduce
- **Your algorithm here**

*Base*

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

**Highly Customizable**

- Above constructs are independent of type of problem.
- **Handlers** implement (type specific) above methods.
- Popular in constraint-programming, SCIP.
- Examples:
  - LinearHandler
  - BilinearHandler
  - QuadraticHandler
  - MultilinearHandler
  - ...
  -
- Handlers contain type specific methods to:
  - Presolve, reformulate and relax a problem.
  - Check feasibility of a given point.
  - Separate or cut a given point.
  - Find a branching candidate.

- Above constructs are independent of type of problem.
- **Handlers** implement (type specific) above methods.
- Popular in constraint-programming, SCIP.
- Examples:
  - LinearHandler
  - BilinearHandler
  - QuadraticHandler
  - MultilinearHandler
  - ...
  - Your application specific Handler
- Handlers contain type specific methods to:
  - Presolve, reformulate and relax a problem.
  - Check feasibility of a given point.
  - Separate or cut a given point.
  - Find a branching candidate.

## *Example of a Handler*

How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

RELAX

BOUND

BRANCH

REPEAT

## *Example of a Handler*

How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

RELAX

$$\begin{array}{ll} \min_x f(x) & \min_x f(x) \\ \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, & \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, & x \in \mathbb{R}^n. \end{array}$$

BOUND

BRANCH

REPEAT

## Example of a Handler

How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

### RELAX

$$\begin{array}{ll} \min_x f(x) & \min_x f(x) \\ \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, & \rightarrow \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, & x \in \mathbb{R}^n. \end{array}$$

### BOUND

- Solve using an NLP solver. Obtain a lower bound, say  $f(x^{NLP})$ .

### BRANCH

### REPEAT

## Example of a Handler

How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

### RELAX

$$\begin{array}{ll} \min_x f(x) & \min_x f(x) \\ \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, & \rightarrow \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, & x \in \mathbb{R}^n. \end{array}$$

### BOUND

- Solve using an NLP solver. Obtain a lower bound, say  $f(x^{NLP})$ .

### BRANCH

- If  $x^{NLP}$  integer feasible, update  $ub$ .
- Otherwise partition:  $x_i \leq \lfloor x_i^{NLP} \rfloor \vee x_i \geq \lceil x_i^{NLP} \rceil$ .

### REPEAT

## Example of a Handler

How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

### RELAX

$$\begin{array}{ll} \min_x f(x) & \min_x f(x) \\ \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, & \rightarrow \text{s.t. } g_i(x) \leq 0, \quad i = 1, 2, \dots, m, \\ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, & x \in \mathbb{R}^n. \end{array}$$

### BOUND

- Solve using an NLP solver. Obtain a lower bound, say  $f(x^{NLP})$ .

### BRANCH

- If  $x^{NLP}$  integer feasible, update  $ub$ .
- Otherwise partition:  $x_i \leq \lfloor x_i^{NLP} \rfloor \vee x_i \geq \lceil x_i^{NLP} \rceil$ .

### REPEAT

- Repeat **RELAX-BOUND-BRANCH** on each part **recursively**,
- until  $\frac{ub-lb}{|ub|} \leq \epsilon$  for a given  $\epsilon \in \mathbb{R}$ .

## Example of a Handler

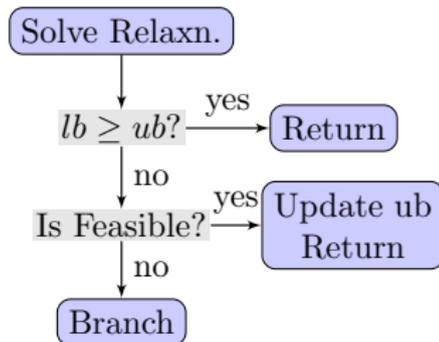
How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

Node Relaxer

Node Processor

Brancher

Do nothing!



Pick a  
fractional variable.

Use `Minotaur::IntVarHandler` for all three

## Example of a Handler

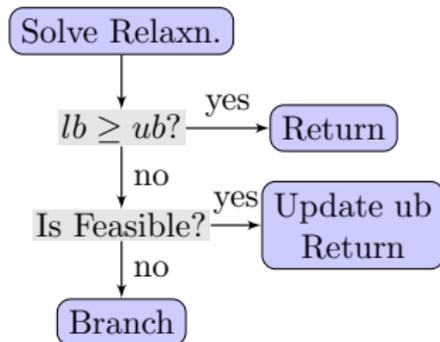
How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

Node Relaxer

Node Processor

Brancher

Do nothing!



Pick a fractional variable.

Use Minotaur::IntVarHandler for all three

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

```
bool isFeasible() {  
  // test integrality  
}
```

```
cand* findBrCandidates() {  
  // return fractional variables  
}  
branch(cand) {  
  // return modifications  
}
```

## Example of a Handler

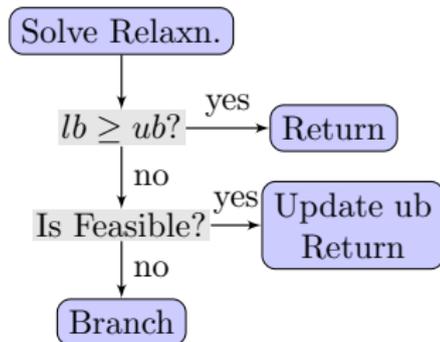
How to write a convex **NLP Branch-and-Bound** solver with Minotaur.

Node Relaxer

Node Processor

Brancher

Do nothing!



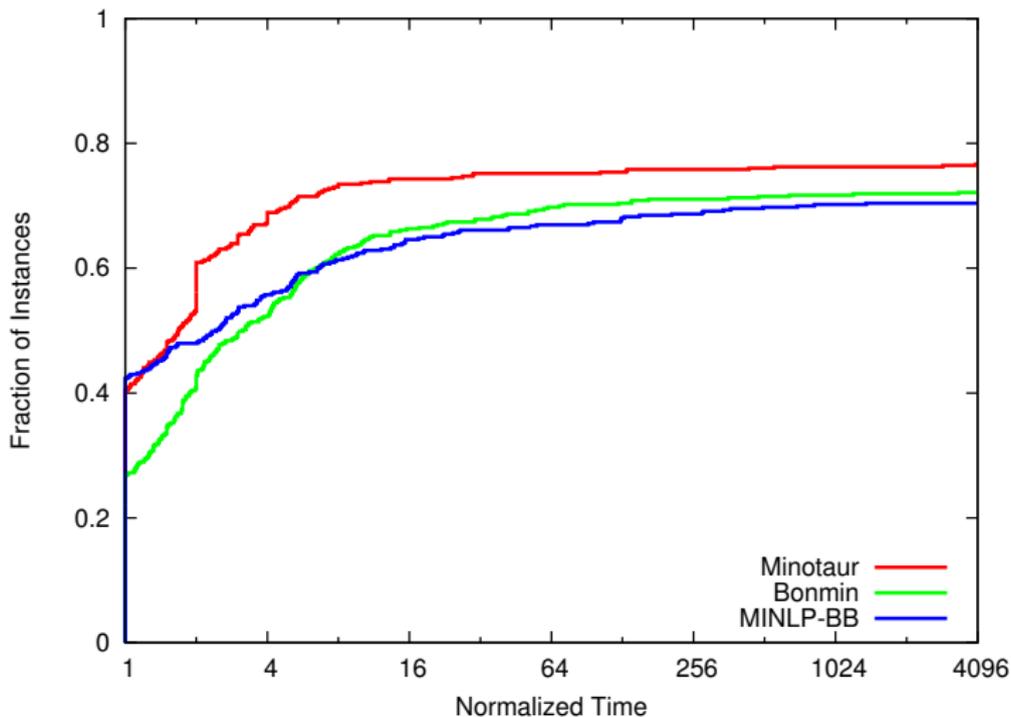
Pick a fractional variable.

Use `Minotaur::IntVarHandler` for all three

Solver in < 200 lines:

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

# Performance



Time taken for 463 MINLP Instances from GAMS, MacMINLP, CMU test-sets.

PART - II  
Reformulating MINLPs

# *Basic Algorithm: Branch-and-Bound*

## RELAX

- Relax integrality restrictions.
- Relax nonconvex constraints.
  - Linear: Secant approx., McCormick approx., ..., BARON, Couenne.
  - Quadratic:  $\alpha$ -BB.
  - ...

## BOUND

- Solve using an LP, QP, NLP or SDP solver. Obtain a lower bound.

## BRANCH

- If solution of relaxation satisfies (MINLP), update  $ub$ ,
- Otherwise, partition the feasible region into two (or more) parts.

## REPEAT

- Repeat **RELAX-BOUND-BRANCH** on each part **recursively**,
- until  $\frac{ub-lb}{|ub|} \leq \epsilon$  for a given  $\epsilon \in \mathbb{R}$ .



## *What Could Go Wrong*

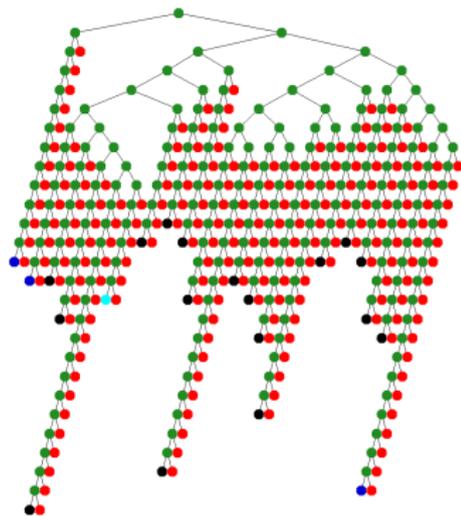
Syn20M04M : a synthesis design problem in  
chemical engineering

Problem size : 160 Integer Variables,  
56 Nonlinear constraints

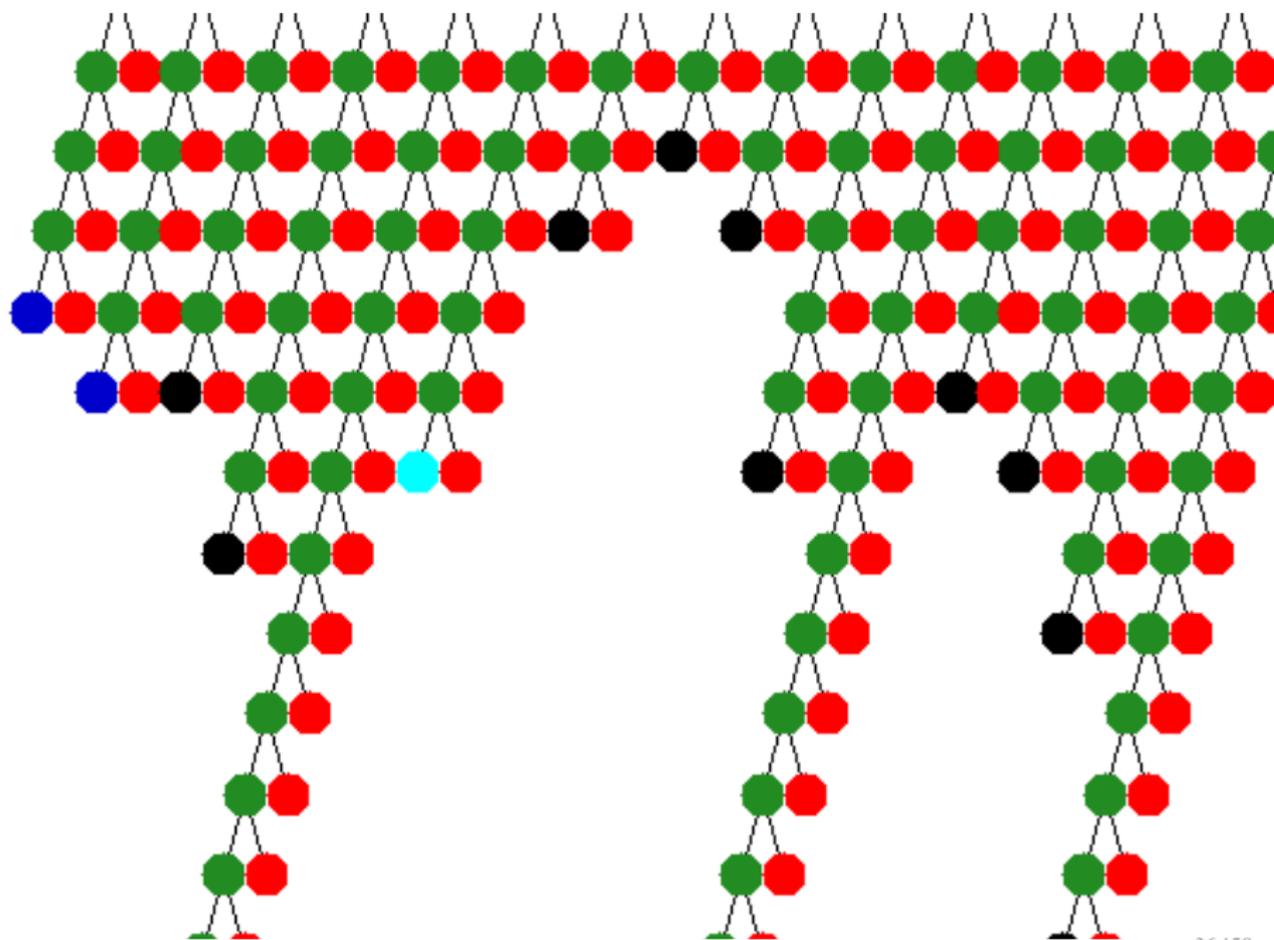
# *What Could Go Wrong*

Syn20M04M : a synthesis design problem in  
chemical engineering

Problem size : 160 Integer Variables,  
56 Nonlinear constraints



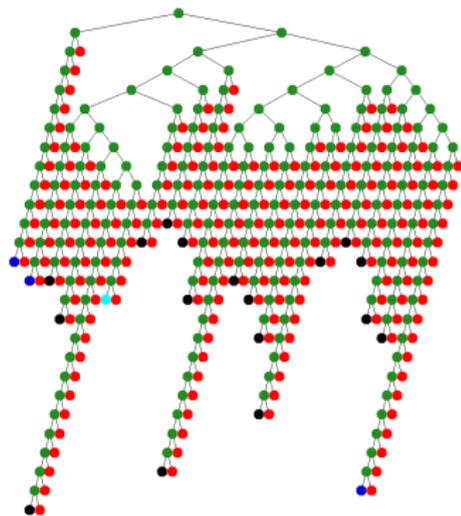
250+ nodes after solving for 45s



# *What Could Go Wrong*

Syn20M04M : a synthesis design problem in  
chemical engineering

Problem size : 160 Integer Variables,  
56 Nonlinear constraints

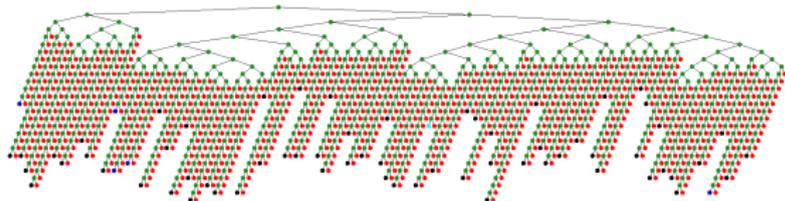


250+ nodes after solving for 45s

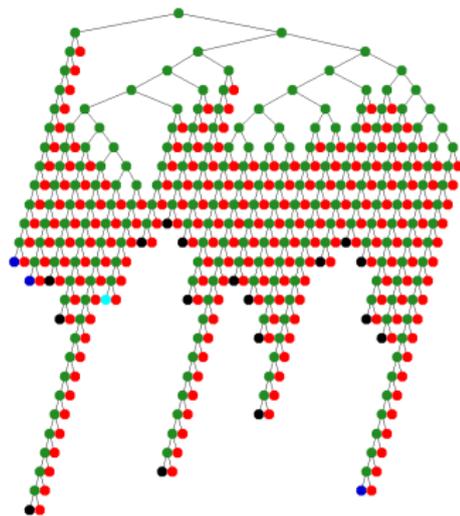
## *What Could Go Wrong*

Syn20M04M : a synthesis design problem in  
chemical engineering

Problem size : 160 Integer Variables,  
56 Nonlinear constraints



1000+ nodes after solving for 75s

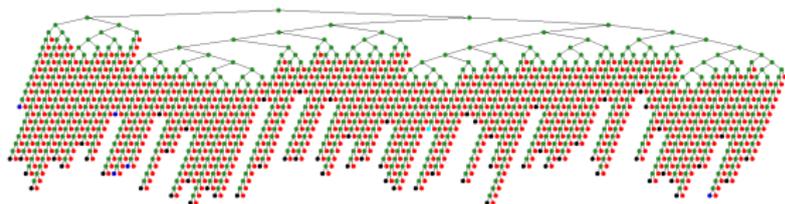


250+ nodes after solving for 45s

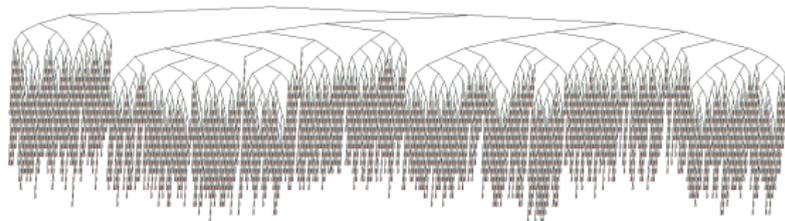
# What Could Go Wrong

Syn20M04M : a synthesis design problem in  
chemical engineering

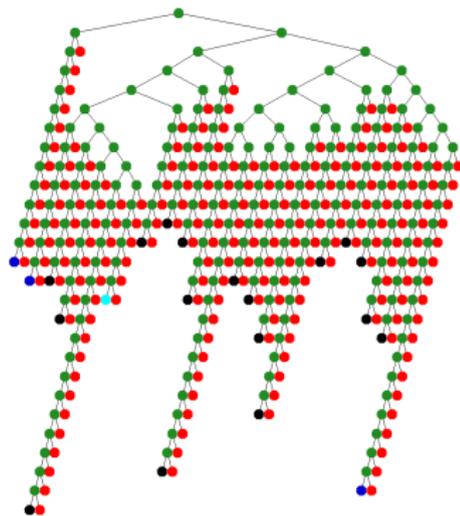
Problem size : 160 Integer Variables,  
56 Nonlinear constraints



1000+ nodes after solving for 75s



5000+ nodes after solving for 200s

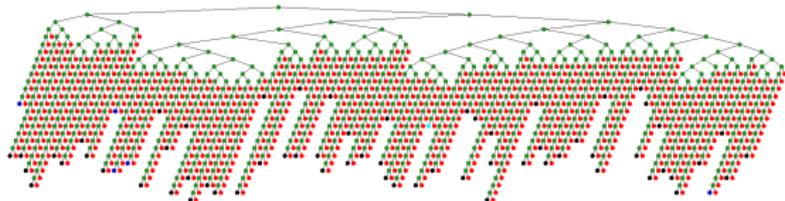


250+ nodes after solving for 45s

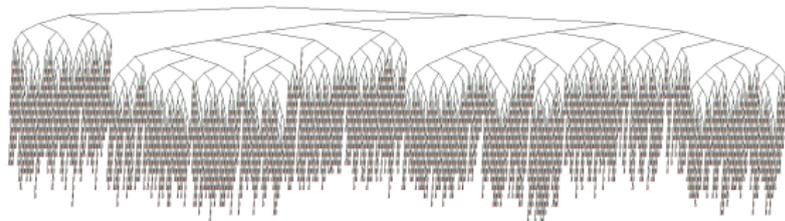
# What Could Go Wrong

Syn20M04M : a synthesis design problem in  
chemical engineering

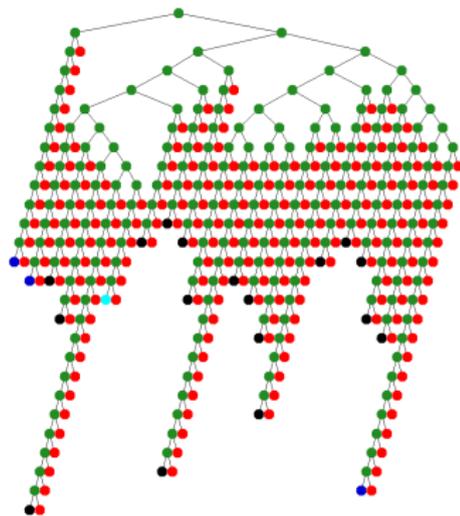
Problem size : 160 Integer Variables,  
56 Nonlinear constraints



1000+ nodes after solving for 75s



5000+ nodes after solving for 200s



250+ nodes after solving for 45s

Solver	Time	Nodes
Bonmin	>2h	>149k
MINLP-BB	>2h	>150k
Minotaur	>2h	>264k



## *Improving Coefficients: An Example*

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

$$0 \leq x_1 \leq 14$$

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



## Improving Coefficients: An Example

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

$$0 \leq x_1 \leq 14$$

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

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

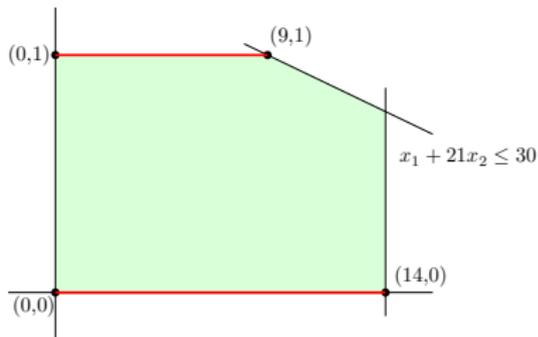
$$x_1 + 0 \leq 30$$

(1) is loose.

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

$$x_1 \leq 9$$

(1) is tight.



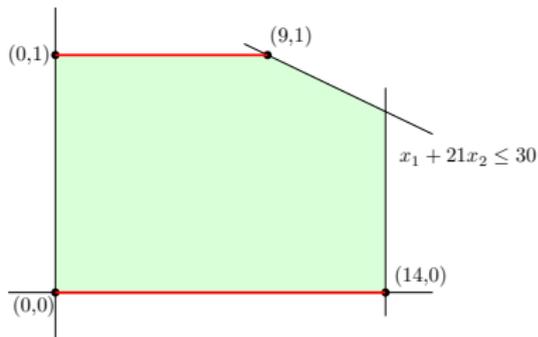


## Improving Coefficients: An Example

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

$$0 \leq x_1 \leq 14$$

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



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

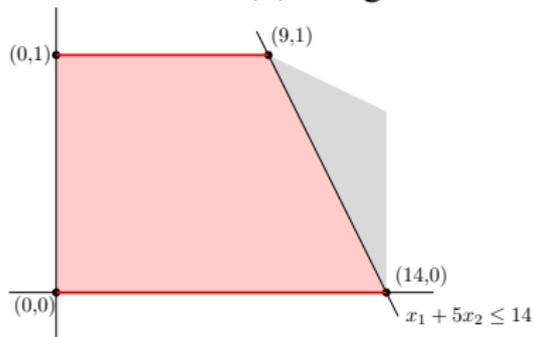
$$x_1 + 0 \leq 30$$

(1) is loose.

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

$$x_1 \leq 9$$

(1) is tight.

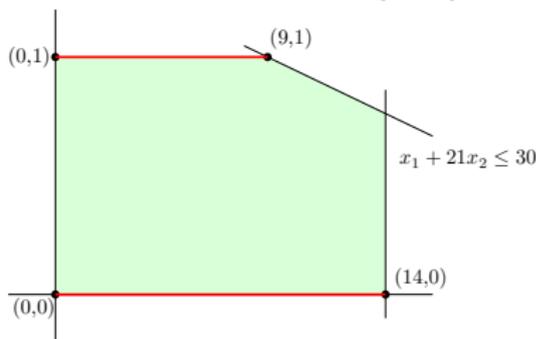


## Improving Coefficients: An Example

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

$$0 \leq x_1 \leq 14$$

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



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

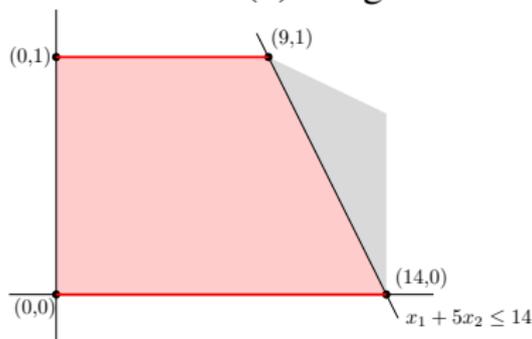
$$x_1 + 0 \leq 30$$

(1) is loose.

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

$$x_1 \leq 9$$

(1) is tight.



Reformulation:

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

$$0 \leq x_1 \leq 14$$

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

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

$$x_1 + 0 \leq 14$$

(2) is tight.

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

$$x_1 \leq 9$$

(2) is tight.

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



## *Improving Coefficients: Linear to Nonlinear*

$$g(x_1, x_2, \dots, x_k) + My \leq b,$$

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

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



## Improving Coefficients: Linear to Nonlinear

$$\begin{aligned}g(x_1, x_2, \dots, x_k) + My &\leq b, \\l_i &\leq x_i \leq u_i, \quad i = 1, \dots, k, \\y &\in \{0, 1\}.\end{aligned}$$

- If  $g(x_1, x_2, \dots, x_k) + M \cdot 0 \leq b$ , is loose, we can tighten the formulation.

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

- If  $g^u < b$ , then reformulate it as  $g(x_1, \dots, x_k) + (M - b + g^u)y \leq g^u$ .



## Improving Coefficients: Linear to Nonlinear

$$\begin{aligned}g(x_1, x_2, \dots, x_k) + My &\leq b, \\l_i &\leq x_i \leq u_i, \quad i = 1, \dots, k, \\y &\in \{0, 1\}.\end{aligned}$$

- If  $g(x_1, x_2, \dots, x_k) + M \cdot 0 \leq b$ , is loose, we can tighten the formulation.

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

- If  $g^u < b$ , then reformulate it as  $g(x_1, \dots, x_k) + (M - b + g^u)y \leq g^u$ .
- In general (MAX-g) is as difficult a problem as the original MINLP.
- An upper bound on (MAX-g) will also tighten it somewhat.
- Trade-off between time and quality of bound. Fast or Tight.



## Improving Coefficients: Using Implications

$$g(x_1, x_2, \dots, x_k) + My \leq b,$$

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

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

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

$$g(x_1, x_2, \dots, x_k) - My \leq b$$

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

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

...

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



## Improving Coefficients: Using Implications

$$g(x_1, x_2, \dots, x_k) + My \leq b,$$

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

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

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

$$g(x_1, x_2, \dots, x_k) - My \leq b$$

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

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

...

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

- $y = 0 \Rightarrow x_1 = x_2, \dots = x_k = 0$ . (**Implications**)
- If  $g(0, \dots, 0) < b$ , then we can tighten.



## Improving Coefficients: Using Implications

$$g(x_1, x_2, \dots, x_k) + My \leq b,$$

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

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

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

$$g(x_1, x_2, \dots, x_k) - My \leq b$$

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

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

...

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

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

## *Just one of Presolve Methods*

### 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: Computational Results**

Syn20M04M from `egon.cheme.cmu.edu`

---

**No Presolve**

---

Variables:	420
Binary Vars:	160
Constraints:	1052
Nonlin. Constr:	56
Bonmin(sec):	>7200
Minotaur(sec):	>7200

---

 **Presolve: Computational Results**Syn20M04M from `egon.cheme.cmu.edu`

---

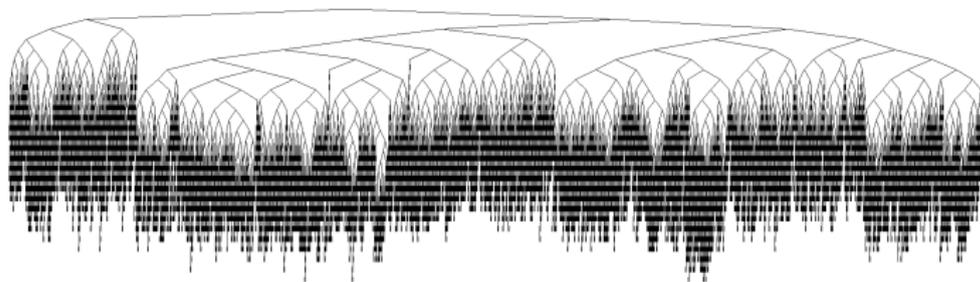
	No Presolve	Basic Presolve
Variables:	420	328
Binary Vars:	160	144
Constraints:	1052	718
Nonlin. Constr:	56	56
Bonmin(sec):	>7200	NA
Minotaur(sec):	>7200	>7200

---

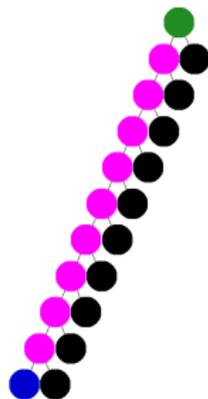
## Presolve: Computational Results

Syn20M04M from `egon.cheme.cmu.edu`

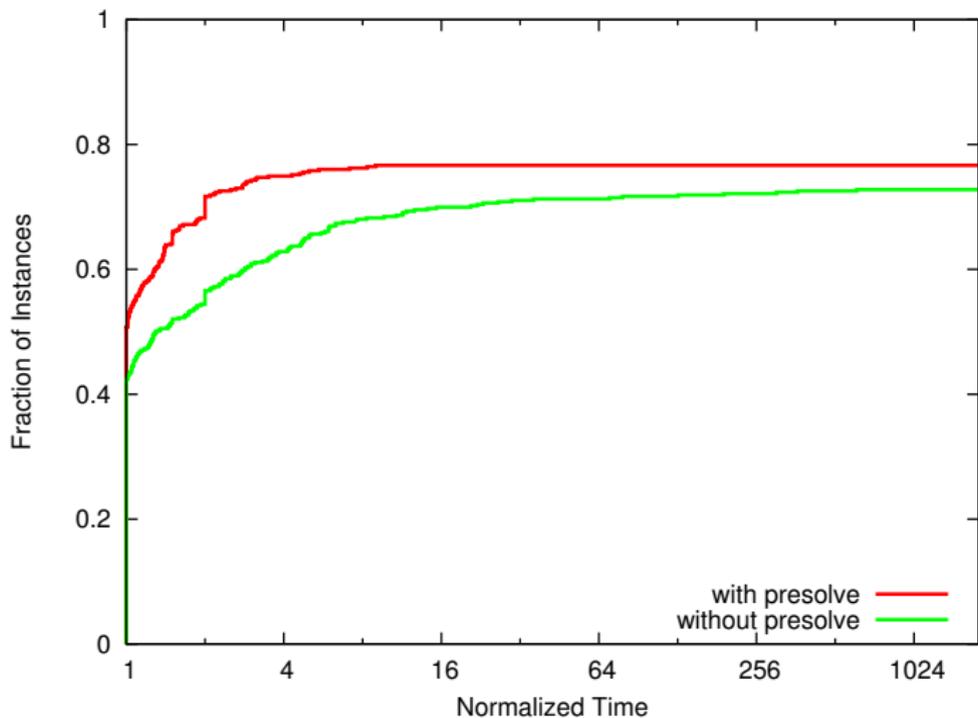
	No Presolve	Basic Presolve	Full Presolve
Variables:	420	328	292
Binary Vars:	160	144	144
Constraints:	1052	718	610
Nonlin. Constr:	56	56	56
Bonmin(sec):	>7200	NA	NA
Minotaur(sec):	>7200	>7200	2.3



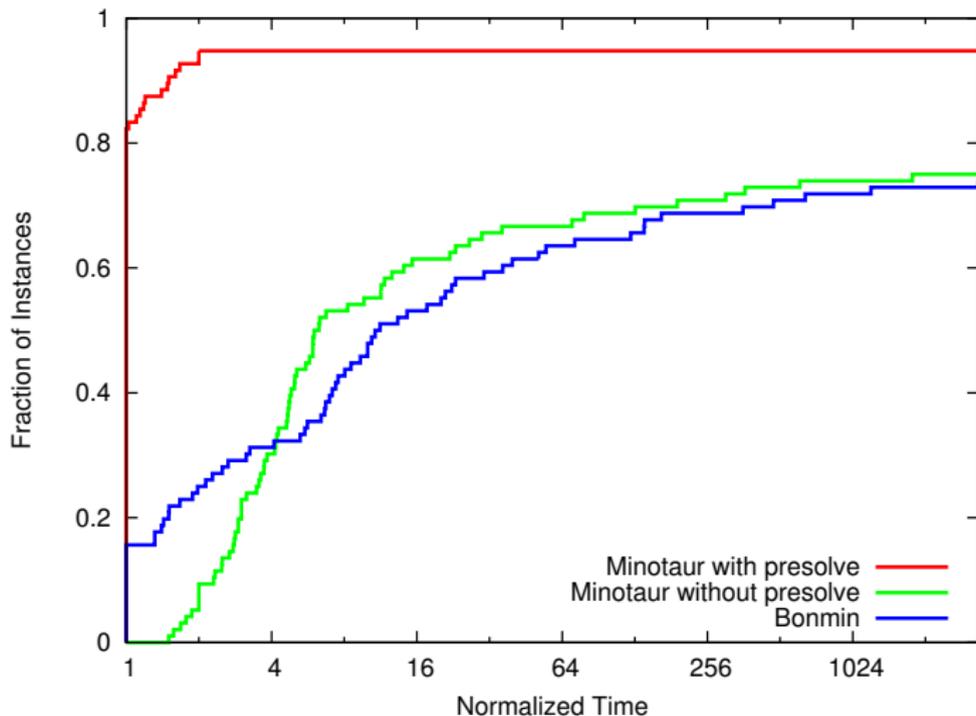
Minotaur, no presolve: 10000+ nodes after solving for 360s



Full Presolve



Time taken in Branch-and-Bound on all 463 instances.



Time taken in Branch-and-Bound on 96 RSyn-X and Syn-X instances.



## *Closing Remarks and Future Directions*

- There are several important applications of MINLP.
- This class of problems offer variety of challenges. Exciting times ahead!
- Both theory and computation.
- Many, many improvements required in several aspects:
  - Tackling non-convexity
  - Reformulating
  - Relaxing
  - Solving and re-solving relaxations
  - Branching
  - Heuristics
  - Representation, evaluation and derivatives of nonlinear functions
- Minotaur will be available soon at  
<http://www.mcs.anl.gov/minotaur>



*A new MINLP Solver*

Ashutosh Mahajan and Team Minotaur

Mathematics and Computer Science Division  
Argonne National Laboratory

2011 DOE Applied Mathematics Program Meeting  
Washington, DC  
October 18, 2011.