Some Recent Advances in Mixed-Integer Nonlinear Programming

Andreas Wächter

IBM T.J. Watson Research Center
Yorktown Heights, New York
andreasw@us.ibm.com

SIAM Conference on Optimization 2008
Boston, MA

May 12, 2008

An MINLP Research Initiative

- CMU-IBM research collaboration, started in 2004
- The Team:

CMU

- Pietro Belotti
- Lorenz T. Biegler
- Gérard Cornuéjols
- Ignacio E. Grossmann
- Carl D. Laird (Texas A&M)
- François Margot
- Nick Sawaya
- Nick Sahinidis

IBM

- Pierre Bonami (CNRS Marseilles)
- Andrew R. Conn
- Claudia D'Ambrosio (U Bologna)
- John J. Forrest
- Joao Goncalves
- Oktay Günlük
- Laszlo Ladanyi
- Jon Lee
- Andrea Lodi (U Bologna)
- Andreas Wächter

Mixed-Integer Nonlinear Programming (MINLP)

$$\begin{aligned} \min \quad & f(x, y) \\ s.t. \quad & c(x, y) \le 0 \\ & y_L \le y \le y_U \\ & x \in \{0,1\}^n,\ y \in \mathbb{R}^p \end{aligned}$$

f, c sufficiently smooth (e.g., C^2)

- Often in practice: Simplify original problem to obtain
  - NLP by relaxing integrality conditions (rounding)
  - MILP by approximating nonlinearities (piece-wise linear)

Mixed-Integer Nonlinear Programming (MINLP)

$$\begin{aligned} \min \quad & f(x, y) \\ s.t. \quad & c(x, y) \leq 0 \\ & y_L \leq y \leq y_U \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p \end{aligned}$$

f, c sufficiently smooth (e.g., C^2) and convex

- Often in practice: Simplify original problem to obtain
  - NLP by relaxing integrality conditions (rounding)
  - MILP by approximating nonlinearities (piece-wise linear)
- Goal: Design exact algorithms
- In this talk: Convex MINLP (f, c convex)

The Power Of MILP

- MILP has been extensively explored for decades
  - Based on branch-and-bound [Dakin (1965)]
  - Very powerful algorithms, techniques, and codes
  - Can solve very large problems
  - Used heavily in practice

The Power Of MILP

- MILP has been extensively explored for decades
  - Based on branch-and-bound [Dakin (1965)]
  - Very powerful algorithms, techniques, and codes
  - Can solve very large problems
  - Used heavily in practice

- How can this be used for MINLP?

- Use MILP solvers directly:
  - Piece-wise linear approximation (SOS constraints)
  - Outer approximation

The Power Of MILP

- MILP has been extensively explored for decades
  - Based on branch-and-bound [Dakin (1965)]
  - Very powerful algorithms, techniques, and codes
  - Can solve very large problems
  - Used heavily in practice
- How can this be used for MINLP?
- Use MILP solvers directly:
  - Piece-wise linear approximation (SOS constraints)
  - Outer approximation
- In a “nonlinear” branch-and-bound algorithm:
  - Try to learn from MILP tricks

The Power Of MILP

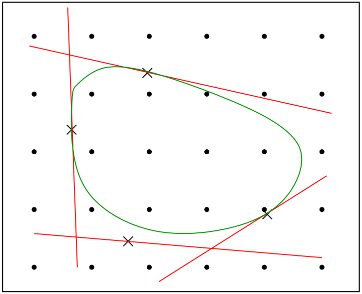
- MILP has been extensively explored for decades
  - Based on branch-and-bound [Dakin (1965)]
  - Very powerful algorithms, techniques, and codes
  - Can solve very large problems
  - Used heavily in practice
- How can this be used for MINLP?
- Use MILP solvers directly:
  - Piece-wise linear approximation (SOS constraints)
  - Outer approximation
- In a "nonlinear" branch-and-bound algorithm:
  - Try to learn from MILP tricks

Outer Approximation (Duran, Grossmann [1986])

$$\begin{array}{ll} \min & z \qquad \text{(linear objective)} \\ s.t. & f(x,y) \le z \\ & c(x,y) \le 0 \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} \end{array}$$

Outer Approximation (Duran, Grossmann [1986])

$$
\begin{aligned}
\min \quad & z \qquad \text{(linear objective)} \\
s.t. \quad & f(x, y) \leq z \\
& c(x, y) \leq 0 \\
& x \in \{0,1\}^n,\ y \in \mathbb{R}^p,\ z \in \mathbb{R}
\end{aligned}
$$



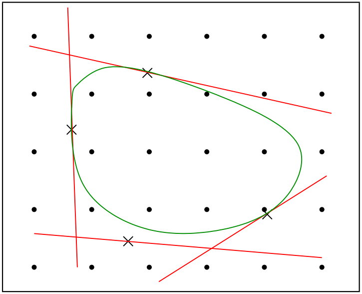
Approximate by MILP (hyperplanes)

$$
\begin{aligned}
\min \quad & z \\
s.t. \quad & \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + f(x^k, y^k) \leq z \\
& \nabla c(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + c(x^k, y^k) \leq 0 \\
& \qquad \text{for all } (x^k, y^k) \in \mathcal{T} \\
& x \in \{0,1\}^n,\ y \in \mathbb{R}^p,\ z \in \mathbb{R}
\end{aligned}
$$

- $\mathcal{T}$ contains linearization points

Outer Approximation (Duran, Grossmann [1986])

$$\begin{aligned} \min \quad & z \qquad \text{(linear objective)} \\ s.t. \quad & f(x, y) \leq z \\ & c(x, y) \leq 0 \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} \end{aligned}$$



Approximate by MILP (hyperplanes)

$$\begin{aligned} \min \quad & z \\ s.t. \quad & \nabla f(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + f(x^k, y^k) \leq z \\ & \nabla c(x^k, y^k)^T \begin{pmatrix} x - x^k \\ y - y^k \end{pmatrix} + c(x^k, y^k) \leq 0 \\ & \qquad \text{for all } (x^k, y^k) \in \mathcal{T} \\ & x \in \{0,1\}^n, y \in \mathbb{R}^p, z \in \mathbb{R} \end{aligned}$$

- $\mathcal{T}$ contains linearization points
  - augmented during algorithm
- Algorithm: Repeat
  1. solve current MILP $\rightarrow (x^l, \tilde{y}^l)$
  2. solve NLP with x^l fixed $\rightarrow y^l$
  3. add (x^l, y^l) to $\mathcal{T}$

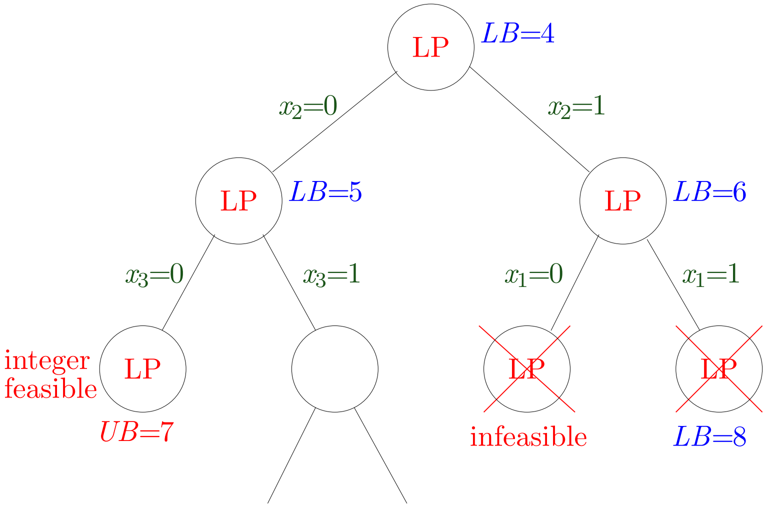
Outer Approximation Discussion

- Original algorithm:
  - Alternatingly solve NLPs and MILPs
  - Finite termination
  - Advantage: Simple to implement; uses all MILP techniques
  - Disadvantage: Solve every MILP from scratch

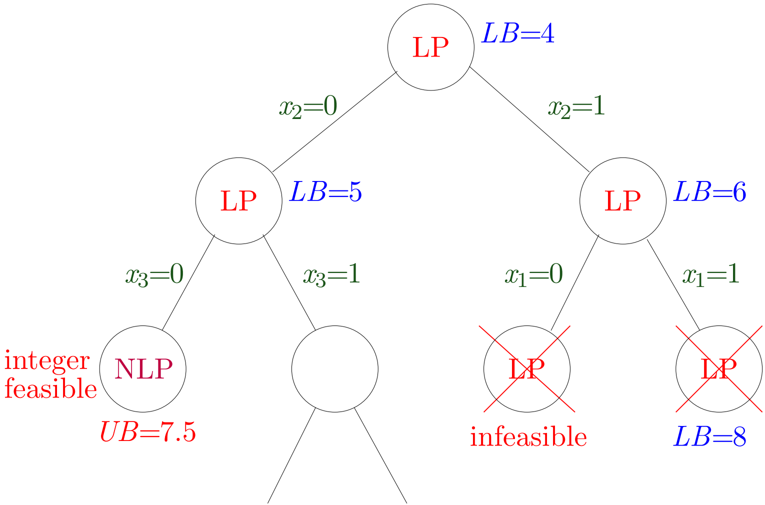
Outer Approximation Discussion

- Original algorithm:
  - Alternatingly solve NLPs and MILPs
  - Finite termination
  - Advantage: Simple to implement; uses all MILP techniques
  - Disadvantage: Solve every MILP from scratch
- Improvement [Quesada, Grossmann (1992)]:
  - Build only <u>one</u> MILP enumeration tree

Quesada-Grossmann



Quesada-Grossmann



Outer Approximation Discussion

- Original algorithm:
  - Alternatingly solve NLPs and MILPs
  - Finite termination
  - Advantage: Simple to implement; uses all MILP techniques
  - Disadvantage: Need to solve every MILP from scratch
- Improvement [Quesada, Grossmann (1992)]:
  - Build only <u>one</u> MILP enumeration tree
  - Solve NLP for every MILP integer feasible solution
  - Add new outer approximation cuts to current MILP

Outer Approximation Discussion

- Original algorithm:
  - Alternatingly solve NLPs and MILPs
  - Finite termination
  - Advantage: Simple to implement; uses all MILP techniques
  - Disadvantage: Need to solve every MILP from scratch
- Improvement [Quesada, Grossmann (1992)]:
  - Build only one MILP enumeration tree
  - Solve NLP for every MILP integer feasible solution
  - Add new outer approximation cuts to current MILP
- "Hybrid" approach [Bonami et al. (2005)]
  - Solve NLPs also at non-integer nodes
  - For example, solve NLP in every 10th node
    - \+ Includes information about nonlinear geometry more quickly
    - – Requires solution of more NLPs
  - Abhishek, Leyffer, Linderoth (2007) (`FilMINT` code):
    - ⋆ Don't solve NLP, just add linearization (Extended cutting plane)

Preliminary Numerical Experiments

- Software implementation
  - Bonmin (Open source software on COIN-OR)

    http://www.coin-or.org/Bonmin
  - Based on other COIN-OR projects (Cbc, Clp, Cgl, Ipopt, ...)

    - Essential for fast development: Availability of open source
  - NLP solvers: FilterSQP [Fletcher, Leyffer] and Ipopt

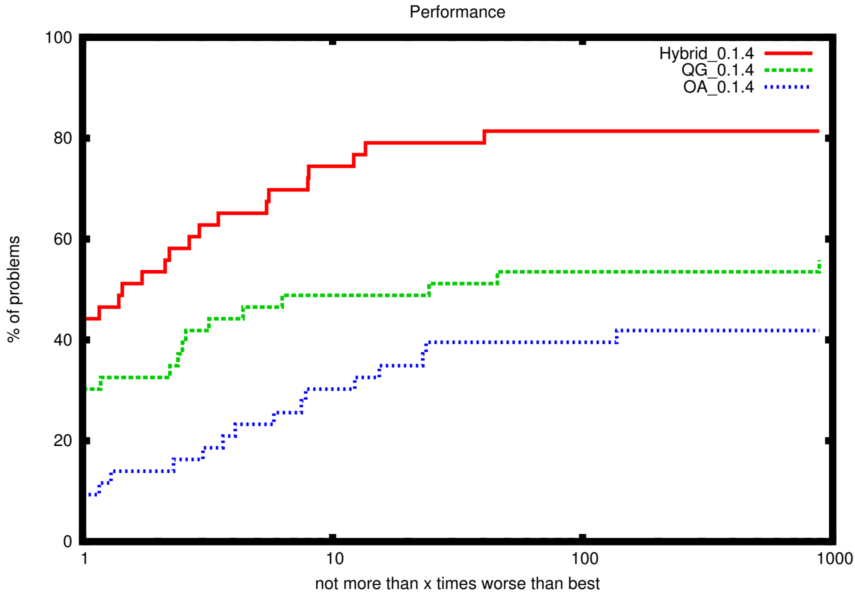
Preliminary Numerical Experiments

- Software implementation
  - Bonmin (Open source software on COIN-OR)

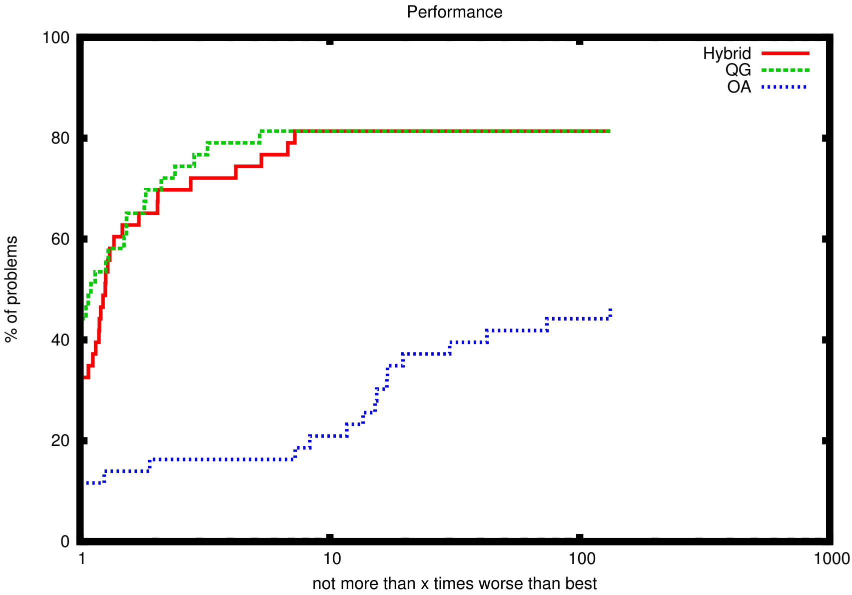
    http://www.coin-or.org/Bonmin
  - Based on other COIN-OR projects (Cbc, Clp, Cgl, Ipopt, ...)
    - Essential for fast development: Availability of open source
  - NLP solvers: FilterSQP [Fletcher, Leyffer] and Ipopt
- Test problems
  - Representative selection of 44 convex MINLPs from
    - CMU/IBM library

      http://egon.cheme.cmu.edu/ibm/page.htm
    - MacMinlp [Leyffer]
  - Difficult, but mostly solvable within 3 hour time limit
  - Problem statistics
    - ★ # total vars: 42–1796 (289.8); # discrete vars: 14–432 (93.7)
    - ★ # constraints: 42–3190 (395.4)

Bonmin 0.1.4 with Ipopt (CPU)



Developer Version with FilterSQP (CPU)



The Success Story Of MILP

In: Bixby, Fenelon, Gu, Rothberg, Wunderling (2004)
Mixed-Integer Programming: A Progress Report

What lead to the dramatic improvement of MILP solvers?

The Success Story Of MILP

In: Bixby, Fenelon, Gu, Rothberg, Wunderling (2004)
Mixed-Integer Programming: A Progress Report

What lead to the dramatic improvement of MILP solvers?

- Very efficient node solvers

The Success Story Of MILP

In: Bixby, Fenelon, Gu, Rothberg, Wunderling (2004)
Mixed-Integer Programming: A Progress Report

What lead to the dramatic improvement of MILP solvers?

- Very efficient node solvers
- Variable/node selection
- Primal heuristics
- Presolve
- Cutting planes

The Success Story Of MILP

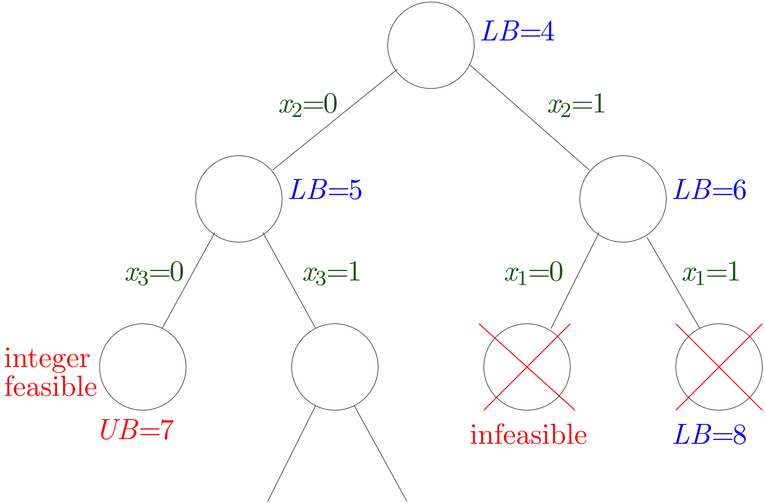
In: Bixby, Fenelon, Gu, Rothberg, Wunderling (2004)
Mixed-Integer Programming: A Progress Report

What lead to the dramatic improvement of MILP solvers?

- Very efficient node solvers
- Variable/node selection
- Primal heuristics
- Presolve
- Cutting planes

What can we learn from this for a B&B-based method for MINLP?

Branch-and-bound: Variable Selection



Variable Selection

Some possible options:

- Random
- Most-fractional (most integer-infeasible)
  - used in MINLP-BB [Fletcher, Leyffer]

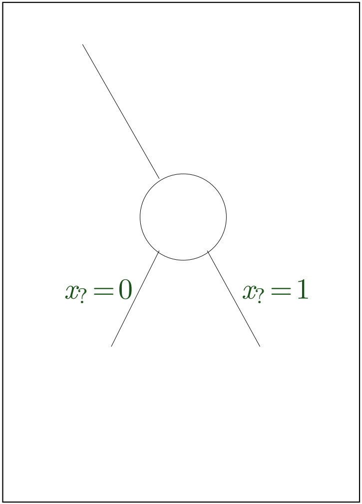
Variable Selection

Some possible options:

- Random
- Most-fractional (most integer-infeasible)
  - used in `MINLP-BB` [Fletcher, Leyffer]
- Strong branching [Applegate et al. (1995)]
- Pseudo costs [Benichou et al. (1971), Forrest et al. (1974)]
  - optional in `SBB` [GAMS]
- Reliability branching [Achterberg et al. (2005)]

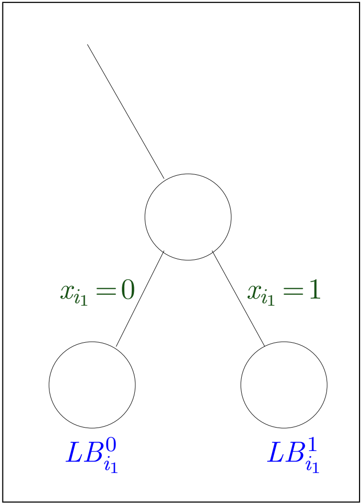
Strong Branching

- Q: Which variable x_i should be branched on?



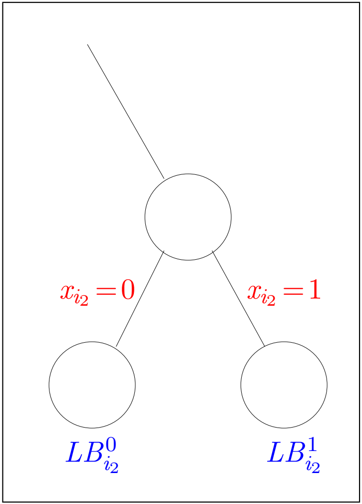
Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates x_{i_1},



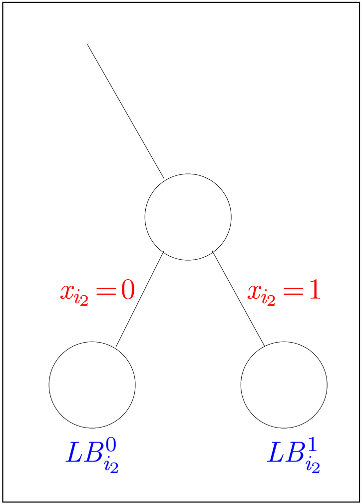
Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates $x_{i_1}, x_{i_2}, \ldots$



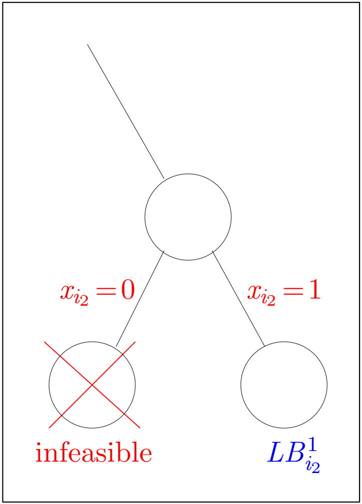
Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates $x_{i_1}, x_{i_2}, \ldots$
- Choose candidate with largest LB_i^0 and LB_i^1



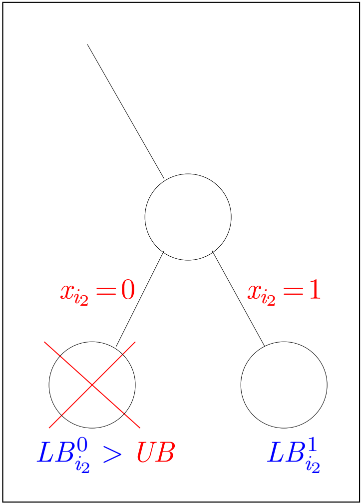
Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates $x_{i_1}, x_{i_2}, \ldots$
- Choose candidate with largest LB_i^0 and LB_i^1
- If candidate's child infeasible: fix variable



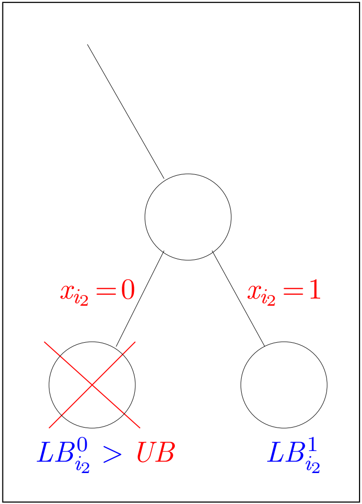
Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates $x_{i_1}, x_{i_2}, \ldots$
- Choose candidate with largest LB_i^0 and LB_i^1
- If candidate's child infeasible: fix variable
- If $LB_i^{0/1} > UB$: fix variable



Strong Branching

- Q: Which variable x_i should be branched on?
- Idea: Try some candidates $x_{i_1}, x_{i_2}, \ldots$
- Choose candidate with largest LB_i^0 and LB_i^1
- If candidate's child infeasible: fix variable
- If $LB_i^{0/1} > UB$: fix variable
- Requires to solve many relaxations



Strong Branching Improvements

Approximate node solutions

- For MILP: Limit the number of simplex iterations
  - Dual simplex algorithm gives valid bounds

Strong Branching Improvements

Approximate node solutions

- For MILP: Limit the number of simplex iterations
  - Dual simplex algorithm gives valid bounds
- For MINLP: Solve approximation problem
  - LP: Linearize functions at parent solution
  - QP: Use QP from last SQP iteration (BQPD [Fletcher])

Strong Branching Improvements

Approximate node solutions

- For MILP: Limit the number of simplex iterations
  - Dual simplex algorithm gives valid bounds
- For MINLP: Solve approximation problem
  - LP: Linearize functions at parent solution
  - QP: Use QP from last SQP iteration (BQPD [Fletcher])
- Can use hot-starts (reuse factorization)
  - Only one bound changes

Strong Branching Improvements

Pseudo costs

- Idea: Collect statistical data about the effect of fixing each x_i:
  - Average change in LB_i^0 and LB_i^1 per unit change in x_i (up and down change separately)
- Use to estimate LB_i^0 and LB_i^1 of child nodes

Strong Branching Improvements

Pseudo costs

- Idea: Collect statistical data about the effect of fixing each x_i:
  - Average change in LB_i^0 and LB_i^1 per unit change in x_i (up and down change separately)
- Use to estimate LB_i^0 and LB_i^1 of child nodes
- Initialize with strong branching
- Update each time a node has been solved

Strong Branching Improvements

Pseudo costs

- Idea: Collect statistical data about the effect of fixing each x_i:
  - Average change in LB_i^0 and LB_i^1 per unit change in x_i (up and down change separately)
- Use to estimate LB_i^0 and LB_i^1 of child nodes
- Initialize with strong branching
- Update each time a node has been solved

Reliability branching

- Pseudo costs, but do strong-branching on non-trusted variables
- Limit the number of strong-branching solves

Variable Selection

Comparative experiments in literature:

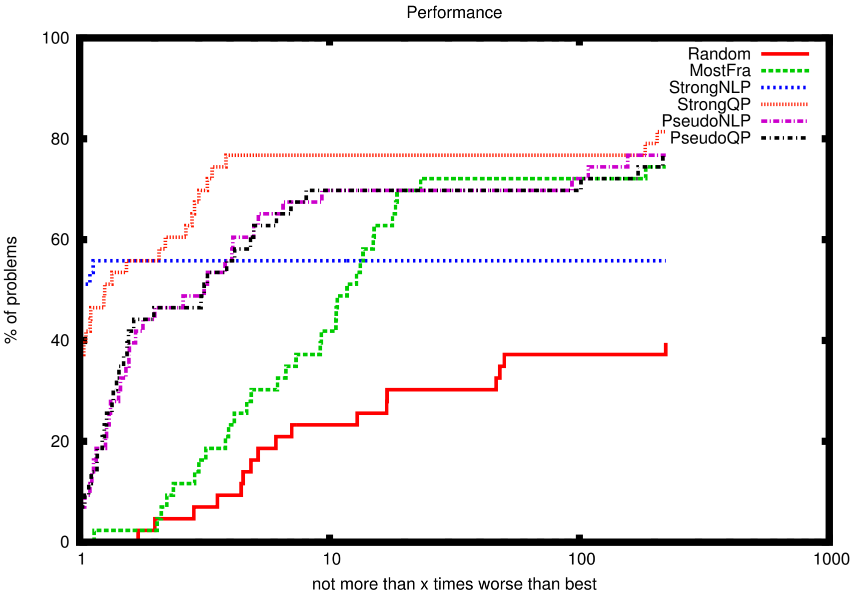
- MILP
  - Linderoth, Savelsbergh (1999):
    - Pseudo costs work very well
  - Achterberg, Koch, Martin (2005):
    - Reliability branching best
    - Most-fractional about as good as Random

Variable Selection

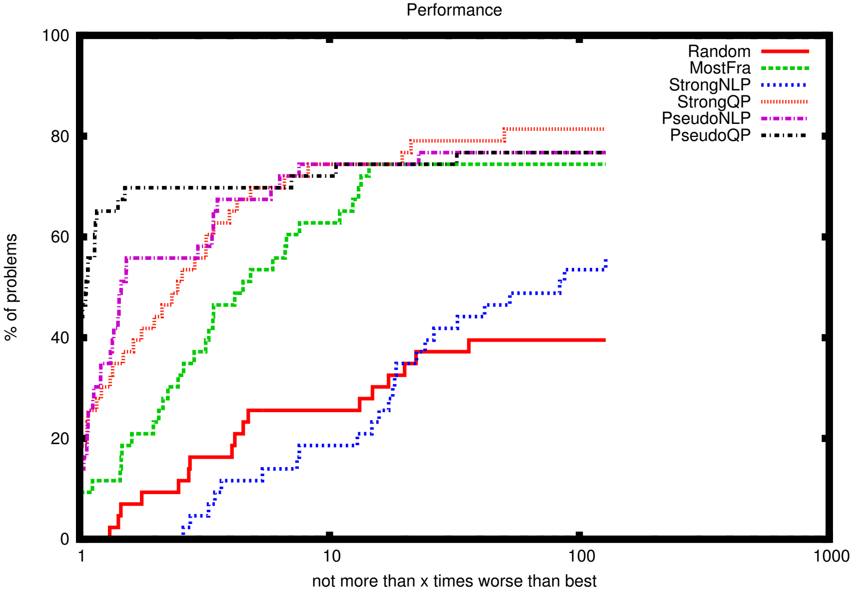
Comparative experiments in literature:

- MILP
  - Linderoth, Savelsbergh (1999):
    - Pseudo costs work very well
  - Achterberg, Koch, Martin (2005):
    - Reliability branching best
    - Most-fractional about as good as Random
- MINLP
  - Gupta, Ravindran (1985)
    - Most-fractional works best

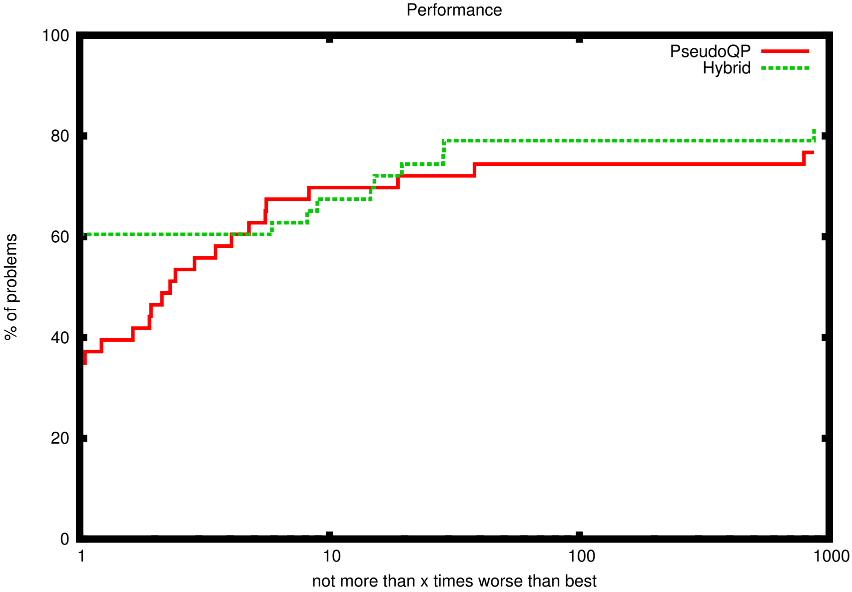
Branch-And-Bound Comparison (# Nodes)



Branch-And-Bound Comparison (CPU time)



B&B and Hybrid Comparison



Experiments Summary

- Strong-branching, pseudo-costs work for nonlinear B&B
  - Hot-started QP approximations improve performance
  - LP approximation not efficient
  - In these experiments: Reliability branching not helpful

Experiments Summary

- Strong-branching, pseudo-costs work for nonlinear B&B
  - Hot-started QP approximations improve performance
  - LP approximation not efficient
  - In these experiments: Reliability branching not helpful
- B&B competitive to OA-based Hybrid method
  - Methods should "learn from each other"
    - e.g., use nonlinear strong-branching in Hybrid approach
- Best choice depends on problem instance
  - Need to identify relevant problem characteristics

Experiments Summary

- Strong-branching, pseudo-costs work for nonlinear B&B
  - Hot-started QP approximations improve performance
  - LP approximation not efficient
  - In these experiments: Reliability branching not helpful
- B&B competitive to OA-based Hybrid method
  - Methods should "learn from each other"
    - e.g., use nonlinear strong-branching in Hybrid approach
- Best choice depends on problem instance
  - Need to identify relevant problem characteristics
- Number of nodes for solved problems:

	Min	Max	GeoMean
Hybrid	8	436393	6226.5
StrongQP	14	2033352	1685.8

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

In MINLP:

- NLP solvers now much more robust and efficient than in the past
  - For `trimloss4`: Solved >2,000,000 NLPs! (105 [85] var, 64 con)

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

In MINLP:

- NLP solvers now much more robust and efficient than in the past
  - For `trimloss4`: Solved >2,000,000 NLPs! (105 [85] var, 64 con)
- Large-scale problems:
  - Large-scale active-set methods?
  - Combine interior-point and active-set methods?

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

In MINLP:

- NLP solvers now much more robust and efficient than in the past
  - For `trimloss4`: Solved >2,000,000 NLPs! (105 [85] var, 64 con)
- Large-scale problems:
  - Large-scale active-set methods?
  - Combine interior-point and active-set methods?
- Hot-starts possible?

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

In MINLP:

- NLP solvers now much more robust and efficient than in the past
  - For `trimloss4`: Solved >2,000,000 NLPs! (105 [85] var, 64 con)
- Large-scale problems:
  - Large-scale active-set methods?
  - Combine interior-point and active-set methods?
- Hot-starts possible?
- Storing warm-start information more memory intensive
  - In experiments: Use optimal solution of root node

Node Solvers

In MILP:

- Very efficient implementation of dual simplex
  - Tailored to B&B: Changes in bounds; added cuts
- Hot-starts (reusing factorization) extremely efficient

In MINLP:

- NLP solvers now much more robust and efficient than in the past
  - For `trimloss4`: Solved >2,000,000 NLPs! (105 [85] var, 64 con)
- Large-scale problems:
  - Large-scale active-set methods?
  - Combine interior-point and active-set methods?
- Hot-starts possible?
- Storing warm-start information more memory intensive
  - In experiments: Use optimal solution of root node
- Need fast detection of infeasibility

Cuts

- Approximate convex hull of integer-feasible points
  - Strengthen the relaxation

Cuts

- Approximate convex hull of integer-feasible points
  - Strengthen the relaxation
- MILP: (hot topic over past 30 years)
  - Many cut generators available (many easy to compute)

Cuts

- Approximate convex hull of integer-feasible points
  - Strengthen the relaxation
- MILP: (hot topic over past 30 years)
  - Many cut generators available (many easy to compute)
- MINLP:
  - For linear parts, can use MILP machinery directly
    - Hybrid method works with linear formulation
    - B&B: could work with linearizations

Cuts

- Approximate convex hull of integer-feasible points
  - Strengthen the relaxation
- MILP: (hot topic over past 30 years)
  - Many cut generators available (many easy to compute)
- MINLP:
  - For linear parts, can use MILP machinery directly
    - Hybrid method works with linear formulation
    - B&B: could work with linearizations
  - Some research specific for nonlinear case:
    - Stubbs, Mehrotra (1999, 2002)
    - Atamtürk, Narayanan (2007)
    - ...
  - Can also use nonlinear cuts
  - Ideally: Need access to problem representation (expression tree)

Other MILP techniques

Primal heuristics (quickly finding good integer feasible points)

- Have answer when time limit exceeded
- Improve upper bounds (e.g, for strong branching)

Other MILP techniques

Primal heuristics (quickly finding good integer feasible points)

- Have answer when time limit exceeded
- Improve upper bounds (e.g, for strong branching)
- MILP: A dozen generic heuristics (root node and in tree)
  (hot topic over last 7 years)
- MINLP: Preliminary work, e.g.,
  - Nonlinear feasibility pump [Bonami et al. (2006)]

Other MILP techniques

Node selection

- In experiments: Use "best-bound" (node with smallest LB)
- Diving
  - Quickly find integer solution
  - Allows hot-starts when proceeding to child nodes

Other MILP techniques

Node selection

- In experiments: Use "best-bound" (node with smallest LB)
- Diving
  - Quickly find integer solution
  - Allows hot-starts when proceeding to child nodes

Presolve (tighten and simplify formulation)

- At root node and in search tree
- MILP: Just look at coefficients of linear functions
- MINLP: General nonlinear functions difficult to predict
  - Requires access to problem representation (e.g., expression tree)

What is Good Modeling?

Example: Uncapacitated facility location problem

$$\begin{array}{rl} \min & \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} y_{ij} \\ s.t. & \sum_{j=1}^{m} y_{ij} = 1 \quad (i = 1, \ldots, n) \\ \textit{Weak}: & \sum_{i=1}^{n} y_{ij} \leq n \cdot x_i \quad (j = 1, \ldots, m) \\ \textit{Strong}: & y_{ij} \leq x_i \quad (i = 1, \ldots, n;\ j = 1, \ldots, m) \\ & x \in \{0,1\}^n\ ,\ y \in \mathbb{R}_+^m \end{array}$$

	MILP		MINLP	
$n = 30, m = 100$	nodes	time	nodes	time
weak formulation	46,294	143.16		
strong formulation	0	0.18		

What is Good Modeling?

Example: Uncapacitated facility location problem

$$
\begin{array}{rl}
\min & \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} y_{ij}^2 \\
s.t. & \sum_{j=1}^{m} y_{ij} = 1 \quad (i = 1, \ldots, n) \\
\mathit{Weak}: & \sum_{i=1}^{n} y_{ij} \leq n \cdot x_i \quad (j = 1, \ldots, m) \\
\mathit{Strong}: & y_{ij} \leq x_i \quad (i = 1, \ldots, n;\ j = 1, \ldots, m) \\
 & x \in \{0,1\}^n \ , \ y \in \mathbb{R}_+^m
\end{array}
$$

	MILP		MINLP	
$n = 30, m = 100$	nodes	time	nodes	time
weak formulation	46,294	143.16	46,384	8117.52
strong formulation	0	0.18	30,112	7888.24

What is Good Modeling?

Example: Uncapacitated facility location problem

$$\begin{array}{rl} \min & \sum_{i=1}^{n} c_i x_i + \sum_{i=1}^{n} \sum_{j=1}^{m} d_{ij} y_{ij}^2 \\ s.t. & \sum_{j=1}^{m} y_{ij} = 1 \quad (i = 1, \ldots, n) \\ \textit{Weak}: & \sum_{i=1}^{n} y_{ij} \leq n \cdot x_i \quad (j = 1, \ldots, m) \\ \textit{Strong}: & y_{ij} \leq x_i \quad (i = 1, \ldots, n;\ j = 1, \ldots, m) \\ & x \in \{0,1\}^n \ , \ y \in \mathbb{R}_+^m \end{array}$$

	MILP		MINLP	
$n = 30, m = 100$	nodes	time	nodes	time
weak formulation	46,294	143.16	46,384	8117.52
strong formulation	0	0.18	30,112	7888.24
weak with cuts/presolve	25	2.71		

The Nonconvex Case

- Global optimization already very difficult
  - Spatial branch-and-bound with convex under-estimators
  - Incorporation of discrete variables natural
  - Several algorithms and codes:
    `Alpha-BB` [Adjiman et al.], `BARON` [Sahinidis, Tawarmalani],
    `Couenne` [Belotti et al.], `LaGO`, [Nowak, Vigerske], ...
  - Limitation in problem size

The Nonconvex Case

- Global optimization already very difficult
  - Spatial branch-and-bound with convex under-estimators
  - Incorporation of discrete variables natural
  - Several algorithms and codes:
    `Alpha-BB` [Adjiman et al.], `BARON` [Sahinidis, Tawarmalani], `Couenne` [Belotti et al.], `LaGO`, [Nowak, Vigerske], ...
  - Limitation in problem size
- Heuristics based on convex MINLP algorithms
  - Outer-approximation based (e.g., `DICOPT` [Grossmann et al.])
    - use one side of equality constraints based on multipliers
    - allow penalized slack in OA cuts
    - delete violated OA cuts

The Nonconvex Case

- Global optimization already very difficult
  - Spatial branch-and-bound with convex under-estimators
  - Incorporation of discrete variables natural
  - Several algorithms and codes:
    `Alpha-BB` [Adjiman et al.], `BARON` [Sahinidis, Tawarmalani], `Couenne` [Belotti et al.], `LaGO`, [Nowak, Vigerske], ...
  - Limitation in problem size

- Heuristics based on convex MINLP algorithms
  - Outer-approximation based (e.g., `DICOPT` [Grossmann et al.])
    - use one side of equality constraints based on multipliers
    - allow penalized slack in OA cuts
    - delete violated OA cuts
  - Nonlinear branch-and-bound
    - resolve NLPs from different starting points
    - do not trust lower bounds or infeasibilities

Conclusions

- Encouraging progress
  - New algorithms and implementations (e.g., `Bonmin`, `FilMINT`)
  - Outer-approximation based algorithms
    - MILP framework with NLP solves
  - Nonlinear branch-and-bound
    - Pseudo costs, QP-based strong branching

Conclusions

- Encouraging progress
  - New algorithms and implementations (e.g., `Bonmin`, `FilMINT`)
  - Outer-approximation based algorithms
    - MILP framework with NLP solves
  - Nonlinear branch-and-bound
    - Pseudo costs, QP-based strong branching
- Many open questions
  - Can we repeat the success of MILP?
    - Further explore MILP techniques in the nonlinear case
    - Robust large-scale NLP solvers with hot starts?
    - Devise specific nonlinear techniques (e.g., cuts)

Conclusions

- Encouraging progress
  - New algorithms and implementations (e.g., `Bonmin`, `FilMINT`)
  - Outer-approximation based algorithms
    - MILP framework with NLP solves
  - Nonlinear branch-and-bound
    - Pseudo costs, QP-based strong branching
- Many open questions
  - Can we repeat the success of MILP?
    - Further explore MILP techniques in the nonlinear case
    - Robust large-scale NLP solvers with hot starts?
    - Devise specific nonlinear techniques (e.g., cuts)
  - Nonconvex problems

Conclusions

- Encouraging progress
  - New algorithms and implementations (e.g., `Bonmin`, `FilMINT`)
  - Outer-approximation based algorithms
    - MILP framework with NLP solves
  - Nonlinear branch-and-bound
    - Pseudo costs, QP-based strong branching
- Many open questions
  - Can we repeat the success of MILP?
    - Further explore MILP techniques in the nonlinear case
    - Robust large-scale NLP solvers with hot starts?
    - Devise specific nonlinear techniques (e.g., cuts)
  - Nonconvex problems
  - Implementation
    - Collaboration essential (through open source?)
    - "Accessible" nonlinear problem representation
    - Parallel implementation

Conclusions

- Encouraging progress
  - New algorithms and implementations (e.g., `Bonmin`, `FilMINT`)
  - Outer-approximation based algorithms
    - MILP framework with NLP solves
  - Nonlinear branch-and-bound
    - Pseudo costs, QP-based strong branching
- Many open questions
  - Can we repeat the success of MILP?
    - Further explore MILP techniques in the nonlinear case
    - Robust large-scale NLP solvers with hot starts?
    - Devise specific nonlinear techniques (e.g., cuts)
  - Nonconvex problems
  - Implementation
    - Collaboration essential (through open source?)
    - "Accessible" nonlinear problem representation
    - Parallel implementation
- Need representative real-world test problems

Thank you!