Primal heuristic for MINLPs in SCIP

Ambros Gleixner, and Felipe Serrano

Zuse Institute Berlin · serrano@zib.de
SCIP Optimization Suite · http://scip.zib.de

Workshop on Discrepancy Theory and Integer Programming

Amsterdam · June 12, 2018

SPONSORED BY THE

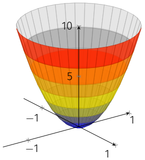
Federal Ministry of Education and Research

Outline

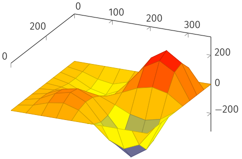
Mixed-Integer Nonlinear Programs (MINLPs)

$$
\begin{aligned}
\min\ & c^{\mathsf{T}}x \\
\text{s.t. } & g_k(x) \le 0 && \forall k \in [m] \\
& x_i \in \mathbb{Z} && \forall i \in \mathcal{I} \subseteq [n] \\
& x_i \in [\ell_i, u_i] && \forall i \in [n]
\end{aligned}
$$

The functions $g_k \in C^1([\ell, u], \mathbb{R})$ can be



convex or



nonconvex

One way of solving MINLPs to global optimality

- Methods for finding (good) feasible solutions
  $\rightsquigarrow$ Primal heuristics

- Proof that there is no better solution
  $\rightsquigarrow$ LP-based spatial branch and bound

LP based spatial Branch & Bound

- Build a (extended formulation of a) polyhedral relaxation $\mathcal{R}$
- Solve $\mathcal{R}$ and get solution x^*
- If x^* is feasible we are done. If not,
- Try to strengthen $\mathcal{R}$ by separating x^*
- When not possible, branch possibly on continuous variables (spatially)

Building polyhedral relaxations: the problem

- We can start with the variable's bounds as our relaxation.
- Then we have to solve the separation problem: Given $\{x \in [l, u] : g(x) \leq 0\}$ and $\bar{x}$ s.t. $g(\bar{x}) > 0$ either
  - Find a separating inequality or
  - prove that none exists.

Building polyhedral relaxations: the problem

- We can start with the variable's bounds as our relaxation.
- Then we have to solve the separation problem: Given $\{x \in [l, u] : g(x) \leq 0\}$ and $\bar{x}$ s.t. $g(\bar{x}) > 0$ either
  - Find a separating inequality or
  - prove that none exists.
- Very expensive for general g

Building polyhedral relaxations: the problem

- We can start with the variable's bounds as our relaxation.
- Then we have to solve the separation problem: Given $\{x \in [l, u] : g(x) \leq 0\}$ and $\bar{x}$ s.t. $g(\bar{x}) > 0$ either
  - Find a separating inequality or
  - prove that none exists.
- Very expensive for general g
- However, when g is convex, it is as easy as computing a gradient:

$$g(\bar{x}) + \nabla g(\bar{x})(x - \bar{x}) \leq 0$$

Building polyhedral relaxations: the problem

- We can start with the variable's bounds as our relaxation.
- Then we have to solve the separation problem: Given $\{x \in [l, u] : g(x) \leq 0\}$ and $\bar{x}$ s.t. $g(\bar{x}) > 0$ either
  - Find a separating inequality or
  - prove that none exists.
- Very expensive for general g
- However, when g is convex, it is as easy as computing a gradient:

$$g(\bar{x}) + \nabla g(\bar{x})(x - \bar{x}) \leq 0$$

- Idea: find convex underestimator $\hat{g}$ of $g(x)$

Building polyhedral relaxations: the problem

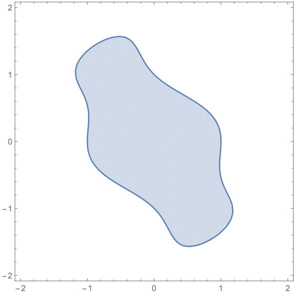
- We can start with the variable's bounds as our relaxation.
- Then we have to solve the separation problem: Given $\{x \in [l, u] : g(x) \leq 0\}$ and $\bar{x}$ s.t. $g(\bar{x}) > 0$ either
    - Find a separating inequality or
    - prove that none exists.
- Very expensive for general g
- However, when g is convex, it is as easy as computing a gradient:

$$g(\bar{x}) + \nabla g(\bar{x})(x - \bar{x}) \leq 0$$

- Idea: find convex underestimator $\hat{g}$ of $g(x)$
- Then $\{x \in [l, u] : g(x) \leq 0\} \subseteq \{x \in [l, u] : \hat{g}(x) \leq 0\}$
- If $\hat{g}(\bar{x}) > 0$ we can separate.

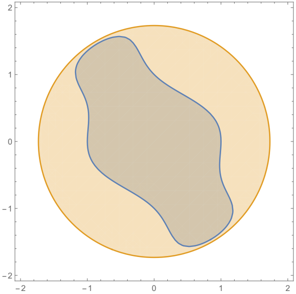
Building polyhedral relaxations: an example

- $x^2 + y^2 + 2\exp(xy^3) \leq 3$ with $x, y \in [-2, 2]$



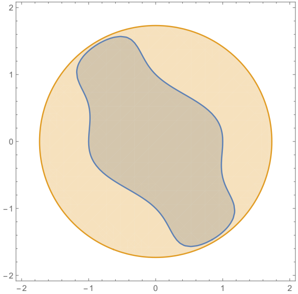
Building polyhedral relaxations: an example

- $x^2 + y^2 + 2\exp(xy^3) \leq 3$ with $x, y \in [-2, 2]$
- $\exp(\cdot) > 0 \Rightarrow x^2 + y^2 \leq x^2 + y^2 + 2\exp(xy^3)$



Building polyhedral relaxations: an example

- $x^2 + y^2 + 2\exp(xy^3) \leq 3$ with $x, y \in [-2, 2]$
- $\exp(\cdot) > 0 \Rightarrow x^2 + y^2 \leq x^2 + y^2 + 2\exp(xy^3)$



- Admittedly, ad-hoc argument
- In practice: if functions are simple enough, we know convex/concave envelopes
- If function is not simple enough, make it simpler!

Building polyhedral relaxation in SCIP

- $x^2 + y^2 + 2\exp(xy^3) \leq 3$
- Introduce auxiliary variables
  - $z_1 = y^3$
  - $z_2 = xz_1$
  - $x^2 + y^2 + 2\exp(z_2) \leq 3$
- We can find polyhedral relaxations of $z_1 = y^3$
- For $z_2 = xz_1$ we have McCormick inequalities:

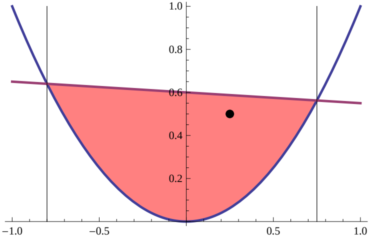
$$\max\{x\underline{z_1} + z_1\underline{x} - \underline{x}\underline{z_1}, x\bar{z}_1 + z_1\bar{x} - \bar{x}\bar{z}_1\} \leq z_2 \leq \min\{x\bar{z}_1 + z_1\underline{x} - \underline{x}\bar{z}_1, x\underline{z_1} + z_1\bar{x} - \bar{x}\underline{z_1}\}$$

- Finally, $x^2 + y^2 + 2\exp(z_2)$ is convex

Spatial Branch and bound

Solutions might not be separable: $conv\{(x, y) : x^2 = y, x \in [\ell, u]\}$ is

$$x^2 \leq y \leq \ell^2 + \frac{u^2 - \ell^2}{u - \ell}(x - \ell) \quad \forall x \in [\ell, u].$$

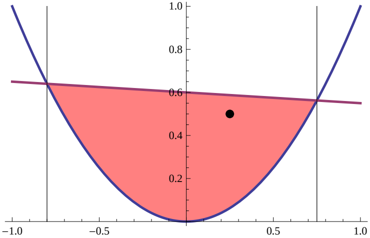


Spatial Branch and bound

Solutions might not be separable: $conv\{(x, y) : x^2 = y, x \in [\ell, u]\}$ is

$$x^2 \leq y \leq \ell^2 + \frac{u^2 - \ell^2}{u - \ell}(x - \ell) \quad \forall x \in [\ell, u].$$

Branching on a nonlinear variable in a nonconvex constraint allows for tighter relaxations:



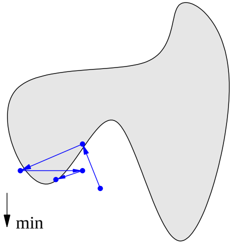
Outline

Sub-NLP

- Idea: fix integer variables to integer values and run a local NLP solver
- Good for: MINLP

Let $\bar{x}$ be LP-optimum of the current node's relaxation

- If there is a $i \in \mathcal{I}$ such that $\bar{x}_i \notin \mathbb{Z} \rightarrow$ STOP
- Fix x_i to $\bar{x}_i$.
- Solve remaining NLP to local optimality using $\bar{x}$ as initial point.

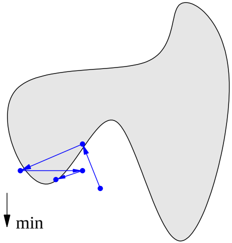


Sub-NLP

- Idea: fix integer variables to integer values and run a local NLP solver
- Good for: MINLP

Let $\bar{x}$ be LP-optimum of the current node's relaxation

- If there is a $i \in \mathcal{I}$ such that $\bar{x}_i \notin \mathbb{Z} \rightarrow$ STOP
- Fix x_i to $\bar{x}_i$.
- Solve remaining NLP to local optimality using $\bar{x}$ as initial point.



Extends MIP heuristics to MINLP

- SCIP runs its MIP heuristics on MIP relaxation of MINLP
- use heuristic's proposed solution as $\bar{x}$

NLP diving

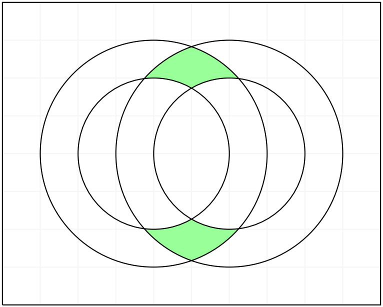
- Idea: Solve NLP relaxations, fixing an integer variable after each NLP
- Good for: MINLP

- solve NLP relaxation
- fix an integer variable
- propagate
- repeat
- if fixing is infeasible, backtrack
  - undo last fixing and fix to another value
  - if infeasible again, abort

Multi-start Heuristic [Chinneck et al. 2013]

- Idea: use different starting points for NLP solver
- Good for: NLPs without too many integer variables

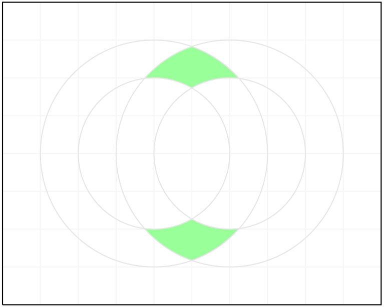
$$4 \leq x^2 + y^2 \leq 9$$
$$4 \leq (x-2)^2 + y^2 \leq 9$$
$$x \in [-4, 6]$$
$$y \in [-4, 4]$$



Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

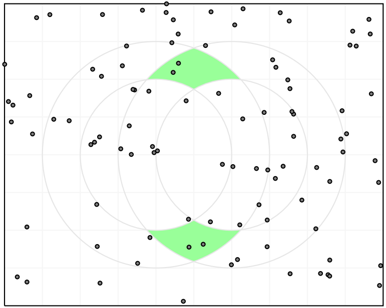
1. generate random points in $[\ell, u]$



Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$

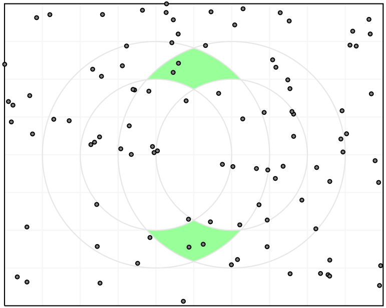


Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$
2. for each point x_k:
   - $s_i^k := \frac{-g_i(x_k)}{||\nabla g_i(x_k)||^2} \nabla g_i(x_k) \; \forall i$
   - Update:

$$x_k^+ = x_k + \frac{1}{n} \sum_{i=1}^{n} s_i^k$$

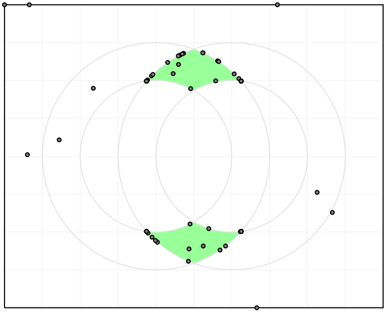


Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$
2. for each point x_k:
   - $s_i^k := \frac{-g_i(x_k)}{||\nabla g_i(x_k)||^2} \nabla g_i(x_k) \; \forall i$
   - Update:

$$x_k^+ = x_k + \frac{1}{n} \sum_{i=1}^{n} s_i^k$$

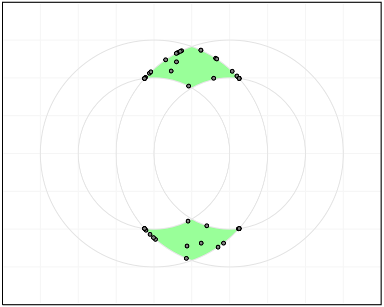


Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$
2. for each point x_k:
   - $s_i^k := \frac{-g_i(x_k)}{||\nabla g_i(x_k)||^2} \nabla g_i(x_k) \; \forall i$
   - Update:

$$x_k^+ = x_k + \frac{1}{n} \sum_{i=1}^{n} s_i^k$$



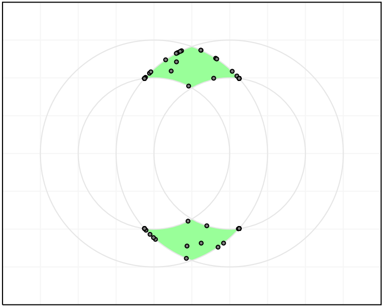
Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$
2. for each point x_k:
   - $s_i^k := \frac{-g_i(x_k)}{||\nabla g_i(x_k)||^2} \nabla g_i(x_k) \; \forall i$
   - Update:

$$x_k^+ = x_k + \frac{1}{n} \sum_{i=1}^{n} s_i^k$$

3. identify clusters of points $C_1, \ldots, C_p$



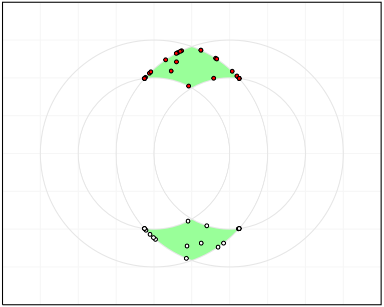
Multi-start in SCIP

Input: Nonlinear Constraints $g_i(x) \leq 0$

1. generate random points in $[\ell, u]$
2. for each point x_k:
   - $s_i^k := \frac{-g_i(x_k)}{||\nabla g_i(x_k)||^2} \nabla g_i(x_k) \; \forall i$
   - Update:

$$x_k^+ = x_k + \frac{1}{n} \sum_{i=1}^{n} s_i^k$$

3. identify clusters of points $C_1, \ldots, C_p$
4. for each cluster
   - build convex combination $y := \frac{1}{|C_j|} \sum_{x \in C_j} x$
   - round each fractional integer variable to closest integer
   - use resulting point as starting point for NLP



MPEC Heuristic [Schewe and Schmidt 2016]

- Idea: write $x \in \{0,1\}$ as $x(1-x)=0$, relax to $x(1-x) \leq \varepsilon$ and solve sequences of NLP with $\varepsilon \to 0$
- Good for: Mixed Binary NLPs
- MPEC stands for Mathematical Program with Equilibrium Constraints

$$
\begin{aligned}
& \min f(x) \\
s.t.\quad & g_j(x) \leq 0 && \forall j \in \{1, \dots, m\} \\
& x_i \in [\ell_i, u_i] && \forall i \in \{1, \dots, k\} \\
& x_i \in \{0, 1\} && \forall i \in \{k+1, \dots, n\}
\end{aligned}
$$

MPEC Heuristic [Schewe and Schmidt 2016]

- Idea: write $x \in \{0,1\}$ as $x(1-x)=0$, relax to $x(1-x) \leq \varepsilon$ and solve sequences of NLP with $\varepsilon \to 0$
- Good for: Mixed Binary NLPs
- MPEC stands for Mathematical Program with Equilibrium Constraints

$$
\begin{aligned}
& \min f(x) \\
s.t.\quad & g_j(x) \leq 0 && \forall j \in \{1,\dots,m\} \\
& x_i \in [\ell_i, u_i] && \forall i \in \{1,\dots,k\} \\
& x_i \in [0,1] && \forall i \in \{k+1,\dots,n\} \\
& x_i(1-x_i) \leq 0 && \forall i \in \{k+1,\dots,n\}
\end{aligned} \tag{NLP}
$$

MPEC Heuristic [Schewe and Schmidt 2016]

- Idea: write $x \in \{0,1\}$ as $x(1-x) = 0$, relax to $x(1-x) \leq \varepsilon$ and solve sequences of NLP with $\varepsilon \to 0$
- Good for: Mixed Binary NLPs

- MPEC stands for Mathematical Program with Equilibrium Constraints

$$
\begin{aligned}
& \min f(x) \\
s.t.\quad & g_j(x) \leq 0 && \forall j \in \{1,\dots,m\} \\
& x_i \in [\ell_i, u_i] && \forall i \in \{1,\dots,k\} \\
& x_i \in [0,1] && \forall i \in \{k+1,\dots,n\} \\
& x_i(1-x_i) \leq \varepsilon && \forall i \in \{k+1,\dots,n\}
\end{aligned}
\qquad (NLP_\varepsilon)
$$

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \to$ STOP

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \rightarrow$ STOP
3. if x^* is feasible for NLP_ε but not binary:

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \rightarrow$ STOP
3. if x^* is feasible for NLP_ε but not binary:
   - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. x^* := solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \rightarrow$ STOP
3. if x^* is feasible for NLP_ε but not binary:
    - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$
4. x^* is infeasible for NLP_ε but satisfies $x_i^*(1 - x_i^*) \leq \varepsilon \rightarrow$ STOP

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. x^* := solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \to$ STOP
3. if x^* is feasible for NLP_ε but not binary:
    - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$
4. x^* is infeasible for NLP_ε but satisfies $x_i^*(1 - x_i^*) \leq \varepsilon \to$ STOP
5. x^* is infeasible for NLP_ε and there are $x_i^*(1 - x_i^*) > \varepsilon$

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \rightarrow$ STOP
3. if x^* is feasible for NLP_ε but not binary:
   - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$
4. x^* is infeasible for NLP_ε but satisfies $x_i^*(1 - x_i^*) \leq \varepsilon \rightarrow$ STOP
5. x^* is infeasible for NLP_ε and there are $x_i^*(1 - x_i^*) > \varepsilon$
   - reset or fix x_i for binary variables with $x_i^*(1 - x_i^*) > \varepsilon$

MPEC in SCIP

Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. $x^* :=$ solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \to$ STOP
3. if x^* is feasible for NLP_ε but not binary:
    - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$
4. x^* is infeasible for NLP_ε but satisfies $x_i^*(1 - x_i^*) \leq \varepsilon \to$ STOP
5. x^* is infeasible for NLP_ε and there are $x_i^*(1 - x_i^*) > \varepsilon$
    - reset or fix x_i for binary variables with $x_i^*(1 - x_i^*) > \varepsilon$
    - $x_i := \begin{cases} 0, & x_i^* > 0.5 \\ 1, & \text{otherwise} \end{cases}$

MPEC in SCIP

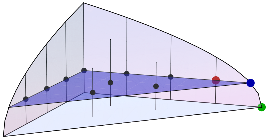
Algorithm

0. choose $\varepsilon \in [0, \frac{1}{4}]$ and $x \in \mathbb{R}^n$
1. x^* := solution of NLP_ε using x as starting point
2. if x^* is feasible for NLP_ε and $x_i^* \in \{0, 1\} \rightarrow$ STOP
3. if x^* is feasible for NLP_ε but not binary:
    - $\varepsilon := \frac{\varepsilon}{2}$; GOTO 1. with $x := x^*$
4. x^* is infeasible for NLP_ε but satisfies $x_i^*(1 - x_i^*) \leq \varepsilon \rightarrow$ STOP
5. x^* is infeasible for NLP_ε and there are $x_i^*(1 - x_i^*) > \varepsilon$
    - reset or fix x_i for binary variables with $x_i^*(1 - x_i^*) > \varepsilon$
    - $x_i := \begin{cases} 0, & x_i^* > 0.5 \\ 1, & \text{otherwise} \end{cases}$
    - GOTO 1.

Undercover [Berthold and Gleixner 2014]

- Idea: fix variables so that MINLP transforms into a MIP
- Good for: MIQCQP

- Solve a relaxation (LP, NLP)
- Identify minimum set $\mathcal{C}$ of variables to be fixed in order to obtain a MIP
- Fix variables in $\mathcal{C}$ to relaxation's solution.
- Solve MIP
- Postprocess: Fix integer values and solve NLP



Undercover in SCIP

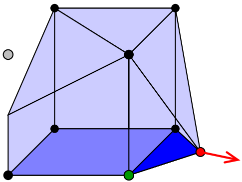
- Identifying minimum set $\mathcal{C}$ of variables to be fixed
  - If $\frac{\partial^2}{\partial x_i x_j} g_k(x) \neq 0$ then x_i or x_j must be fixed
  - Build graph with nodes x_i and arcs $\{x_i, x_j\}$ if $\frac{\partial}{\partial x_i x_j} g_k(x) \neq 0$
  - Minimum vertex cover yields the set we are looking for.
- Fix variables in $\mathcal{C}$ to relaxation's solution
  - Fix a variable, if variable must be integer, fix to rounded value.
  - Propagate bounds: to avoid "obvious" infeasible fixings
  - If next fixing value is outside bounds, choose closest bound or resolve relaxation
  - If infeasible, backtrack: undo last fixing and try new value

RENS: The optimal rounding [Berthold 2014]

- RENS stands for Relaxation Enforced Neighborhood Search
- Idea: restrict integer variables around LP-relaxation's solution and solve remaining MINLP
- Good for: MIP, MINLP

- Let $\bar{x}$ be the LP solution
- If $|\{i \in \mathcal{I} : \bar{x}_i \in \mathbb{Z}\}| > p|\mathcal{I}|$ for some $p \in [0, 1]$ then
- Change bounds of x_i to $\{\lfloor \bar{x}_i \rfloor, \lceil \bar{x}_i \rceil\}$
- Solve smaller MINLP

- This gives the feasible rounding with best objective value



Computational results for RENS

Table 1 Computing optimal roundings (aggregated results)

	Integrality		Succ	Prim. gap		Comp. effort	
	>90 %	Avg (%)		Mean	Std dev	Nodes	Time (s)
MIP + cuts	55/159	71.7	95/159	8.22	20.34	814.4	22.6
MIP − cuts	62/159	73.6	80/159	10.88	21.19	719.9	21.7
MIQCP (LP)	9/70	59.9	49/70	11.61	13.14	627.7	30.9
MIQCP (NLP)	1/70	13.8	48/70	1.30	3.37	7,078.8	168.1
MINLP (LP)	6/105	63.5	65/105	13.80	17.73	11,175.6	83.0
MINLP (NLP)	1/105	15.0	73/105	4.60	14.99	93,908.0	262.7

- Basic LP solutions often show high integrality
- Success rate seems to decreases when more integer variables are fixed
- Roundability of MIP and MINLP are similar

Computational results for RENS

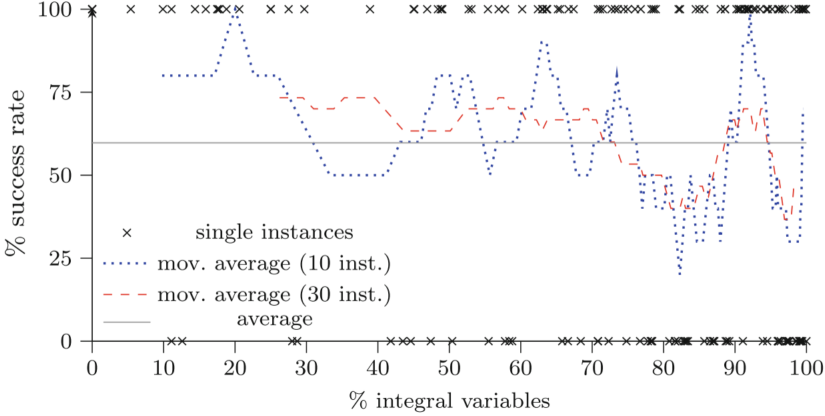


Fig. 3 Moving averages of success rate, MIPLIB instances, after cuts

- Basic LP solutions often show high integrality
- Success rate seems to decreases when more integer variables are fixed
- Roundability of MIP and MINLP are similar

Outline

Conclusion

Summary

- very brief introduction to LP-based branch and bound
- overview of MINLP heuristics in SCIP

Conclusion

Summary

- very brief introduction to LP-based branch and bound
- overview of MINLP heuristics in SCIP

What is missing in SCIP

- Non-linear Feasibility Pump
- Problem specific heuristics

Conclusion

Summary

- very brief introduction to LP-based branch and bound
- overview of MINLP heuristics in SCIP

What is missing in SCIP

- Non-linear Feasibility Pump
- Problem specific heuristics

Thank you for your attention

Primal heuristic for MINLPs in SCIP

Ambros Gleixner, and Felipe Serrano

Zuse Institute Berlin · serrano@zib.de
SCIP Optimization Suite · http://scip.zib.de

Workshop on Discrepancy Theory and Integer Programming

Amsterdam · June 12, 2018

ZIB

MODAL FORSCHUNGSCAMPUS

SPONSORED BY THE
Federal Ministry
of Education
and Research

Literature I

Timo Berthold.
RENS – the optimal rounding.
Mathematical Programming Computation, 6(1):33–54, 2014.
doi: 10.1007/s12532-013-0060-9.

Timo Berthold and Ambros M. Gleixner.
Undercover: a primal MINLP heuristic exploring a largest sub-MIP.
Mathematical Programming, 144(1-2):315–346, 2014.
doi: 10.1007/s10107-013-0635-2.

Lars Schewe and Martin Schmidt.
Computing feasible points for minlps with mpecs.
Technical report, Technical report, Optimization Online, 2016. URL
http://www.optimization-online.org/DB_HTML/2016/12/5778.html, 2016.

Literature II

Laurence Smith, John Chinneck, and Victor Aitken.
Improved constraint consensus methods for seeking feasibility in nonlinear programs.
Computational Optimization and Applications, 54(3):555–578, 2013.
doi: 10.1007/s10589-012-9473-z.