Rounding-Based Heuristics for Nonconvex MINLPs

Giacomo Nannicini\textsuperscript{1}, Pietro Belotti\textsuperscript{2}

\textsuperscript{1} Tepper School of Business, Carnegie Mellon University, Pittsburgh, USA
\textsuperscript{2} Department of Industrial & Systems Engineering, Lehigh University, USA

April 12, 2010
Summary of Talk

1. Introduction

2. Basic Algorithmic Scheme
   - Feasibility heuristic
   - Improvement heuristic

3. Computational experiments
1 Introduction

2 Basic Algorithmic Scheme
   - Feasibility heuristic
   - Improvement heuristic

3 Computational experiments
The Problem

• We consider nonconvex Mixed-Integer Nonlinear Programs (MINLPs) of the form:

\[
\begin{align*}
\min & \quad f(x) \\
\forall j \in M & \quad g_j(x) \leq 0 \\
\forall i \in N & \quad x_i^L \leq x_i \leq x_i^U \\
\forall j \in N_I & \quad x_j \in \mathbb{Z},
\end{align*}
\]

with \( n = |N| \) variables and \( m = |M| \) constraints

• An exact solution method is the Branch-and-Bound algorithm, where lower bounds are obtained by linearizing the nonconvex continuous relaxation of each subproblem

• Good upper bounds are doubly important for this kind of algorithm:
  ◦ More pruning
  ◦ Bound tightening (through constraint propagation techniques), therefore better relaxations
1 Introduction

2 Basic Algorithmic Scheme
   - Feasibility heuristic
   - Improvement heuristic

3 Computational experiments
Some Definitions

- Let $\mathcal{P}$ be the original problem, $\mathcal{Q}$ its continuous relaxation.
- Let $F$ be the feasible region of the linearization of $\mathcal{P}$.
- Given an integer feasible point $x^I$, we define $\mathcal{Q}(x^I)$ the NLP which is obtained from $\mathcal{Q}$ by fixing the values of the integer variables in $x^I$.
- Let $NG(\tilde{x})$ be any constraint that cuts off a point $\tilde{x}$. 
Basic Idea

1. Start with a constraint feasible point $x'$ (i.e. satisfies the constraints) which is not integral feasible

2. Round $x'$ subject to linear constraints:

$$x^I = \arg \min_{x \in F} \|x - x'\|_1$$

3. Solve $Q(x^I)$ to obtain $x^*$

4. If some termination condition is met, stop; otherwise, set $F \leftarrow F \cap NG(x^I)$ and return to step 2)
Basic Idea

1. Start with a constraint feasible point $x'$ (i.e. satisfies the constraints) which is not integral feasible $\leftarrow (?)$

2. Round $x'$ subject to linear constraints:

$$x^I = \arg \min_{x \in F} \|x - x'\|_1$$

3. Solve $Q(x^I)$ to obtain $x^*$

4. If some termination condition is met, stop; otherwise, set $F \leftarrow F \cap NG(x^I)$ and return to step 2) $\leftarrow (?)$
Comments

- Borrows some ideas from Tabu Search [Glover, 1989] and Feasibility Pump [Bonami et al., 2009]
- We get different heuristics depending on the way we choose $x'$ and the linear constraints that define the MILP
- We do not need to solve the problems (NLP, MILP) to optimality – we just need feasibility
- Should work if $x'$ can guide us towards the feasible region
Easy case: binary variables

- No-good cuts are an easy way to cut solutions:

\[
NG(\tilde{x}) : \sum_{i \in B: \tilde{x}_i = 1} (1 - x_i) + \sum_{i \in B: \tilde{x}_i = 0} x_i \geq 1
\]

where \(B\) is the set of binary variables

- Drawbacks:
  1. May be dense

- Advantages:
  1. \(\tilde{x}\) is the only point which is cut off\(^1\)
  2. Linear cut with integer coefficients
  3. Can be computed very quickly

\(^1\)This statement is actually not true (but please don’t pay attention to this comment).
General integer variables

- Observation: in practice, very often $x^I$ has integer variables at one of their bounds.

- In this case, we can use a generalized no-good cut:

$$NG(\tilde{x}) : \sum_{i \in N_I: \tilde{x}_i = x_i^U} (x_i^U - x_i) + \sum_{i \in N_I: \tilde{x}_i = x_i^L} x_i \geq \lambda$$

- If the number of integer variables at one of their bounds is small, we “branch” on a random variable: choose $i \in N_I$ at random, then add the constraint $x_i \geq \tilde{x}_i + 1$ or the constraint $x_i \leq \tilde{x}_i - 1$

- Drawbacks:
  - May cut off a large number of feasible solutions

- Advantages:
  - Extremely fast (just a bound change!)
A Simple Feasibility Heuristic

- Scheme: find a point $x'$ which is feasible to $Q$, and round subject to the linearization $F$ of the feasible region
- If this does not work, we try different points $x'$, computing them by maximizing their “feasibility”
- $x'$ is obtained by:

\[
\begin{align*}
\min & \quad \mu f(x) + (1 - \mu) \left( \sum_{j \in M} g_j(x) \right) \\
\text{subject to:} & \quad g_j(x) \leq 0 \ \forall j \in M
\end{align*}
\]

- This is equivalent to requiring $x'$ to be as much as possible in the interior of the feasible region, using $\mu$ to scale
- Idea: start with $\mu = 1$; if we cannot find a feasible solution, then iteratively decrease $\mu$ to obtain different points $x'$ until $\mu = 0$
- We stop as soon as we find a feasible solution, or after a given number of iterations
- We call this heuristic Feasibility-based Iterative Rounding (F-IR)
Sketch of the Algorithm

1: **Input:** parameters $\mu, \text{MaxIter}$
2: **Output:** feasible solution $x^*$
3: Initialization: stop $\leftarrow$ false, NumIter $\leftarrow$ 0
4: Compute with a NLP solver:

$$x' = \arg\min_{\forall j \in M} \min_{g_j(x) \leq 0} \{ \mu f(x) + (1 - \mu) \sum_{j \in M} g_j(x) \}$$

5: Set $R \leftarrow F$
6: **while** ¬stop **do**
7: Compute $x^I = \arg\min_{x \in R} \|x - x'\|_1$ with a MILP solver
8: Solve $Q(x^I)$ with a NLP solver and obtain point $x^*$
9: **if** $(x^*$ feasible) $\lor$ (NumIter $\geq$ MaxIter) **then**
10: Set stop $\leftarrow$ true
11: **else**
12: Set $R \leftarrow R \cap NG(x^*)$
13: Set NumIter $\leftarrow$ NumIter + 1
14: **return** $x^*$
An improvement heuristic

- We assume that we have an incumbent $\bar{x}$, and want to find a better solution in a neighborhood $H(\bar{x})$ of $\bar{x}$

- Binary case: we let $H(\bar{x})$ be defined by a local branching constraint [Fischetti and Lodi, 2003]:
  $$\sum_{i \in B: \bar{x}_i = 1} (1 - x_i) + \sum_{i \in B: \bar{x}_i = 0} x_i \leq k$$
  with $k > 0$

- General integer case: if $\bar{x}$ has a “sufficient” number of variables at one of their bounds, we still use a local branching constraint; otherwise, we use a small box centered at $\bar{x}$

- We obtain $x'$ by solving $Q \cap H(\bar{x})$

- When rounding $x'$, we add $H(\bar{x})$ to the linearization $F$ of the feasible region:
  $$x^I = \arg \min_{x \in F \cap H(\bar{x})} \|x - x'\|_1$$

- We call this heuristic LocalBranching-based Iterative Rounding (LB-IR)
Sketch of the Algorithm

1. **Input**: incumbent $\bar{x}$, parameters $k$, $MaxIter$
2. **Output**: improved solution $x^*$
3. Initialization: $\text{stop} \leftarrow \text{false}$, $\text{NumIter} \leftarrow 0$
4. Solve $Q \cap H(\bar{x})$ with a NLP solver to obtain $x'$
5. Set $R \leftarrow F \cap H(\bar{x})$
6. **while** $\text{¬stop}$ **do**
7. Compute $x^I = \arg\min_{x \in R} \|x - x'\|_1$ with a MILP solver
8. Solve $Q(x^I)$ with a NLP solver and obtain point $x^*$
9. **if** $((x^\text{feasible} \land f(x^*) < f(\bar{x})) \lor (\text{NumIter} \geq MaxIter))$ **then**
10. Set $\text{stop} \leftarrow \text{true}$
11. **else**
12. Set $R \leftarrow R \cap NG(x^*)$
13. Set $\text{NumIter} \leftarrow \text{NumIter} + 1$
14. **return** $x^*$
1 Introduction

2 Basic Algorithmic Scheme
   - Feasibility heuristic
   - Improvement heuristic

3 Computational experiments
We implemented these heuristics within the open source Branch-and-Bound solver for nonconvex MINLPs Couenne [Belotti, 2009], using the linearization of the original problem computed at the root node.

We used Cplex 12.1 as MILP solver (with MIP_EMPHASIS = HIDDEN_FEAS, Feasibility Pump and Local Branching turned ON, max 50 nodes or 5 seconds CPU time), Ipopt as NLP solver.

We try at most 10 different roundings of $x'$.

The rhs of the local branching constraint is set to 15, and the rhs of the no-good cuts is set to be equal to the average value of $x_i^U - x_i^L$ for the variables which are in the cuts.
Instances

- We tested the heuristic on the full MINLPLib
- Results only for 87 difficult instances: those which are unsolved by Couenne after 30 minutes

| beuster  | fo7_ar5_1   | netmod_dol1 | nuclearva | oil     | water4 |
|----------|-------------|-------------|-----------|---------|--------|
| contvar  | fo7         | netmod_dol2 | nuclearvb | product2| waterz |
| csched2a | fo8_ar2_1   | netmod_kar1 | nuclearvc | qap     |        |
| csched2  | fo8_ar25_1  | netmod_kar2 | nuclearvd | synheat |        |
| deb6     | fo8_ar3_1   | no7_ar2_1   | nuclearve | tln12   |        |
| deb7     | fo8_ar4_1   | no7_ar25_1  | nuclearvf | tln5    |        |
| deb8     | fo8_ar5_1   | no7_ar3_1   | nvs23     | tln6    |        |
| deb9     | fo8         | no7_ar4_1   | nvs24     | tln7    |        |
| eg_all_s | fo9_ar2_1   | no7_ar5_1   | o7_2      | tls12   |        |
| eg_int_s | fo9_ar25_1  | nuclear10a  | o7_ar2_1  | tls4    |        |
| ex1233   | fo9_ar3_1   | nuclear14a  | o7_ar25_1 | tls5    |        |
| feedtray | fo9_ar4_1   | nuclear14b  | o7_ar3_1  | tls6    |        |
| fo7_2    | fo9_ar5_1   | nuclear24a  | o7_ar4_1  | tls7    |        |
| fo7_ar2_1| fo9         | nuclear24b  | o7_ar5_1  | uselinear|       |
| fo7_ar25_1| gasnet     | nuclear25a  | o7         | var_con10|       |
| fo7_ar3_1| lop97ic     | nuclear25b  | o8_ar4_1  | var_con5 |       |
| fo7_ar4_1| lop97icx    | nuclear49a  | o9_ar4_1  | waste   |       |
Feasibility-based Iterative Rounding

- Time limit of 300 seconds, we try to round at most 5 points
- Solutions found for 50 instances out of 87 (57.5%); on 36 instances, Couenne does not find any solution after 30 minutes, and we find a solution at the root for 11 of them (30.5%)
- Average time: 92.5 seconds
- Average time for successful runs: 23.1 seconds
- Average relative distance from the best known solutions: 54.4%
## Feasibility-based Iterative Rounding

| Instance       | Obj.    | Instance       | Obj.    | Instance       | Obj.    |
|----------------|---------|----------------|---------|----------------|---------|
| contvar        | 811036  | fo9_ar3.1      | 70.4345 | o7_2           | 159.828 |
| csched2a       | -146327 | fo9_ar4.1      | 45.9274 | o7_ar2.1       | 168.324 |
| ex1233         | 211571  | fo9_ar5.1      | 60.0466 | o7_ar25.1      | 173.34  |
| fo7_2          | 28.9906 | fo9            | 58.3746 | o7_ar25.1      | 173.34  |
| fo7_ar2.1      | 44.1149 | lop97icx       | 4688.47 | o7_ar3.1       | 201.461 |
| fo7_ar25.1     | 48.8353 | netmod_dol2    | 0.017267| o7_ar4.1       | 158.038 |
| fo7_ar3.1      | 36.6557 | netmod_kar1    | -0.39045| o7_ar5.1       | 178.009 |
| fo7_ar4.1      | 33.9258 | netmod_kar2    | -0.39045| o8_ar4.1       | 335.312 |
| fo7_ar5.1      | 48.8451 | no7_ar2.1      | 137.029 | o9_ar4.1       | 348.934 |
| fo7            | 34.0949 | no7_ar25.1     | 123.965 | synheat        | 196206  |
| fo8_ar2.1      | 42.0356 | no7_ar3.1      | 150.529 | tln5           | 15.4    |
| fo8_ar25.1     | 44.5491 | no7_ar4.1      | 147.009 | var_con10      | 444.214 |
| fo8_ar3.1      | 43.9849 | no7_ar5.1      | 133.696 | var_con5       | 285.874 |
| fo8_ar4.1      | 46.7868 | nuclear14a     | -1.12965| water4         | 1022.47 |
| fo8_ar5.1      | 58.8946 | nuclear24a     | -1.12965|               |         |
| fo8            | 59.1281 | nuclear24b     | -1.08866|               |         |
| fo9_ar2.1      | 65.2216 | nuclearvd      | -1.0393 |               |         |
| fo9_ar25.1     | 47.1241 | nuclearve      | -1.02347|               |         |
Feasibility-based Iterative Rounding

![Bar chart showing the number of instances at different distances from the best known solution.]
Combined Heuristics: Evaluation criteria

- Main purpose of primal heuristic?
- How do we evaluate performance?
- Consider the time interval of 30 minutes
- Compare default Couenne and Couenne + Iterative Rounding by recording for how much time one algorithm has a **strictly better** incumbent
- Can be interpreted as the probability of having a better incumbent using that particular algorithm at a given time instant
Example: lop97icx
Summary of Results: F-IR + LB-IR

On the instances where a solution is found by at least one of the two methods (60 instances), on average:

- Couenne + Iterative Rounding has a better incumbent for 50.9% of the time
- Couenne has a better incumbent for 31.4% of the time

At the end of the solution process:

- Couenne + Iterative Rounding returns a better solution on 29 instances
- Couenne returns a better solution on 18 instances
- On the remaining 13 instances they return the same solution
Comparison: Couenne w/ and w/o Iterative Rounding

![Comparison bar graph showing number of instances versus distance from best known solution for Couenne with and without Iterative Rounding.](image)
Understanding the heuristic: part I

- This whole idea is based on rounding; so what about true rounding?
- Use same scheme: obtain a point $x'$, round it, solve NLP while fixing integer variables; repeat as necessary with a different rounding
- Does this work?
This whole idea is based on rounding; so what about true rounding? Use same scheme: obtain a point \( x' \), round it, solve NLP while fixing integer variables; repeat as necessary with a different rounding.

Does this work?

Not very well!

On the 87 instances: F-IR with simple rounding finds solutions on 12 instances (rounding subject to linearization finds 50)

LB-IR with simple rounding improves a solution only 6 times (rounding subject to linearization improves 98 times)
Understanding the heuristic: part II

- The rounding phase consists in finding \( \min_{x \in F} \|x - x'\|_1 \)

- Does the norm-1 distance really matter?

- Experiment: apply the same heuristics, but take the first solution found by Cplex when solving the rounding MILP; call this \( \text{IR}^- \), and compare to \( \text{IR} \)

- Results:
  - \( \text{Couenne} + \text{IR} \) has a better incumbent for 34.6\% of the time
  - \( \text{Couenne} + \text{IR}^- \) has a better incumbent for 28.7\% of the time

- At the end of the solution process:
  - \( \text{Couenne} + \text{IR} \) returns a better solution on 23 instances
  - \( \text{Couenne} + \text{IR}^- \) returns a better solution on 15 instances
Conclusions

- Simple heuristics to quickly obtain good upper bounds
- Main ideas: start NLP search from integer points in the feasible region of the linearization (extremely important) which are near a constraint-feasible point (important, but to a smaller extent)
- This suggests that using a better linearization could greatly improve the results
- Available soon in Couenne!²

²Hopefully
Thank you!
Bibliography

Belotti, P. (2009).
Couenne: a user’s manual.
Technical report, Lehigh University.

Bonami, P., Cornuéjols, G., Lodi, A., and Margot, F. (2009).
A feasibility pump for Mixed Integer Nonlinear Programs.
Mathematical Programming, 119(2):331–352.

Fischetti, M. and Lodi, A. (2003).
Local branching.
Mathematical Programming, 98:23–37.

Glover, F. W. (1989).
Tabu search – part I.
ORSA Journal on Computing, 1(3):190–206.