Workshop on Integer Programming and Continuous Optimization
Chemnitz, November 2004

Integer Nonlinear Optimization

Sven Leyffer

leyffer@mcs.anl.gov

Mathematics and Computer Science Division, Argonne National Laboratory
1. Introduction & Applications
2. Classical MINLP Methods
3. Modern MINLP Methods
4. Conclusions & Future Work
Integer Nonlinear Optimization

Sven Leyffer

1. Introduction & Applications
2. Classical MINLP Methods
3. Modern MINLP Methods
4. Conclusions & Future Work

Do not trust this expert!
1. Introduction & Applications

Mixed Integer Nonlinear Programming (MINLP)

\[
\begin{align*}
\text{minimize} & \quad f(x, y) \\
\text{subject to} & \quad c(x, y) \leq 0 \\
& \quad x \in X, \ y \in Y \text{ integer}
\end{align*}
\]

- \( f, c \) smooth (convex) functions
- \( X, Y \) polyhedral sets, e.g. \( Y = \{0, 1\} \)
- \( y \in Y \text{ integer} \Rightarrow \text{hard problem} \)
1.1. Core Reload Operation [Quist:97]

• maximize reactor efficiency after reload subject to diffusion PDE & safety

• approx. diffusion by nonlinear equation

⇒ integer & nonlinear model

• avoid reactor becoming sub-critical
1.1. Core Reload Operation [Quist:97]

- maximize reactor efficiency after reload subject to diffusion PDE & safety
- approx. diffusion by nonlinear equation ⇒ integer & nonlinear model
- avoid reactor becoming overheated
1.1. Core Reload Operation [Quist:97]

- look for cycles for moving bundles:
  e.g. $4 \rightarrow 6 \rightarrow 8 \rightarrow 10$
  means bundle moved from 4 to 6 to ...
- model with integer variables $x_{ilm} \in \{0, 1\}$
  $x_{ilm} = 1$: node $i$ has bundle $l$ of cycle $m$
1.2. Other Applications

• Chemical Engineering Applications:
  ○ process synthesis [Kocis & Grossmann:88]
  ○ batch plant design [Grossmanno & Sargent:79]
  ○ cyclic scheduling [Jain & Grossmann:98]
  ○ design of distillation columns [Viswanathan:93]
  ○ pump configuration optimization [Westerlund:94]

• trimloss minimization in paper industry [Westerlund:98]

• topology optimization [Sigurd:00]
  ○ finite element structural optimization
  ○ 0-1 to model presence/absence of material
2. Classical Methods for MINLP

Basic Methods:
1. Branch-and-Bound
2. Outer Approximation, Benders Decomposition et al.

Hybrid Methods:
3. LP/NLP Based Branch-and-Bound
4. Integrating SQP with Branch-and-Bound
2.1. Branch-and-Bound

Solve relaxed NLP \((0 \leq y \leq 1 \text{ continuous relaxation})\)

- Branch on \(y_i\) non-integral
- Solve NLPs & branch until ...
  1. Node infeasible ...
  2. Node integer feasible ...
     \[ y_i = 1 \]
     \[ \rightarrow \text{upper bound (} U \text{)} \]
  3. Lower bound \(\geq U\) ...

Search until no unexplored nodes left on tree
2.2. Outer Approximation [Duran & Grossmann]

**Motivation:** avoid *huge number* of NLPs
- Take advantage of MILP codes: decompose integer & nonlinear part

**Key idea:** reformulate MINLP as MILP (implicit)
- Solve alternating sequence of MILP & NLP

NLP subproblem $y^j$ fixed:

\[
\text{NLP}(y^j) \begin{cases} 
\text{minimize} & f(x, y^j) \\
\text{subject to} & c(x, y^j) \leq 0 \\
& x \in X 
\end{cases}
\]

Main Assumption: $f$, $c$ are convex
2.2. Outer Approximation [Duran & Grossmann]

- let \((x^j, y^j)\) solve NLP\((y^j)\)
- linearize \(f, c\) about \((x^j, y^j) =: z^j\)
- new objective variable \(\eta \geq f(x, y)\)
- MINLP \((P) \equiv MILP \((M)\)

\[
\begin{align*}
(M) \quad \text{minimize} & \quad \eta \\
\text{subject to} & \quad \eta \geq f^j + \nabla f^j^T (z - z^j) \quad \forall y^j \in Y \\
& \quad 0 \geq c^j + \nabla c^j^T (z - z^j) \quad \forall y^j \in Y \\
& \quad x \in X, y \in Y \text{ integer}
\end{align*}
\]

**SNAG:** need all \(y^j \in Y\) linearizations
2.2. Outer Approximation [Duran & Grossmann]

\((M^k)\): lower bound (underestimate convex \(f, c\))

\(\text{NLP}(y^j)\): upper bound \(U\) (fixed \(y^j\))

\[\Rightarrow\text{stop, if lower bound} \geq \text{upper bound}\]
2.2. OA & Benders Decomposition

Take OA master ... \( z := (x, y) \)

\[
(M) \begin{cases}
\begin{aligned}
\text{minimize} & \quad \eta \\
\text{subject to} & \quad \eta \geq f^j + \nabla f^j^T (z - z^j) \quad \forall y^j \in Y \\
& \quad 0 \geq c^j + \nabla c^j^T (z - z^j) \quad \forall y^j \in Y \\
x \in X, \ y \in Y \text{ integer}
\end{aligned}
\end{cases}
\]

sum constraints \( 0 \geq c^j \ldots \) weighted with multipliers \( \lambda^j \ \forall j \)

\[
\Rightarrow \quad \eta \geq f^j + \lambda^j^T c^j + (\nabla f^j + \nabla c^j \lambda^j)^T (z - z^j) \quad \forall y^j \in Y
\]

... valid inequality.
2.2. OA & Benders Decomposition

Valid inequality from OA master; \( z = (x, y) \):

\[
\eta \geq f^j + \lambda^j c^j + (\nabla f^j + \nabla c^j \lambda^j)^T (z - z^j)
\]

use KKT conditions of NLP\((y^j)\) ... 

\[
\nabla_x f^j + \nabla_x c^j \lambda^j = 0
\]

... to eliminate \( x \) components from valid inequality 

\[
\Rightarrow \quad \eta \geq f^j + \lambda^j c^j + (\nabla_y f^j + \nabla_y c^j \lambda^j)^T (y - y^j)
\]

\[
\Leftrightarrow \quad \eta \geq L^j + (\mu^j)^T (y - y^j)
\]

where \( L^j \) Lagrangian ... 

\[
\mu^j = \nabla_y f^j + \nabla_y c^j \lambda^j \quad \text{multiplier of } y = y^j \text{ in NLP}(y^j)\]
2.2. OA & Benders Decomposition

⇒ remove $x$ from master problem & obtain Benders master problem

$$(M_B) \begin{cases} 
\text{minimize} & \eta \\
\text{subject to} & \eta \geq \mathcal{L}^j + (\mu^j)^T (y - y^j) \forall y^j \in Y \\
y \in Y \text{ integer}
\end{cases}$$

where $\mathcal{L}^j$ Lagrangian & $\mu^j$ multiplier of $y = y^j$ in NLP($y^j$)

- $(M_B)$ has less constraints & variables (no $x$!)
- $(M_B)$ almost ILP (except for $\eta$)
- $(M_B)$ weaker than OA (from derivation)
2.2. OA & Similar Methods

**Extended Cutting Plane Method** [Westerlund:95]:
- no NLP \( y^j \) solves; Kelley’s cutting plane method instead
- linearize about \( (\hat{x}^j, y^j) \), solution of \( (M^k) \)
- add most violated linearization to master \( (M^k) \)
  \( \Rightarrow \) slow nonlinear convergence; > 1 evaluation per \( y \)

**Drawbacks of OA, GBD & ECP**:  
- MILP tree-search can be bottle-neck  
- potentially large number of iterations [FL:94]

**Second order master (MIQP)** [FL:94]:  
- add Hessian term to MILP \( (M) \) \( \Rightarrow \) MIQP  
  - solve MIQP by B&B; similar to MILP
2.3. LP/NLP Based Branch-and-Bound [Quesada & Grossmann]

**AIM**: avoid re-solving MILP master \((M)\)

Consider MILP branch-and-bound
2.3. LP/NLP Based Branch-and-Bound [Quesada & Grossmann]

**AIM**: avoid re-solving MILP master \((M)\)

Consider MILP branch-and-bound

interrupt MILP, when new \(y^j\) found

\(\rightarrow\) solve NLP\((y^j)\) get \(x^j\)
2.3. LP/NLP Based Branch-and-Bound [Quesada & Grossmann]

**AIM**: avoid re-solving MILP master \((M)\)

Consider MILP branch-and-bound

interrupt MILP, when new \(y^j\) found

→ solve NLP\((y^j)\) get \(x^j\)

→ linearize \(f, c\) about \((x^j, y^j)\)

→ add linearization to MILP tree
2.3. LP/NLP Based Branch-and-Bound [Quesada & Grossmann]

**AIM**: avoid re-solving MILP master ($M$)

Consider MILP branch-and-bound
interrupt MILP, when new $y_j$ found
→ solve NLP($y_j$) get $x_j$;
→ linearize $f$, $c$ about $(x_j, y_j)$
→ add linearization to MILP tree
→ continue MILP tree-search

... until lower bound $\geq$ upper bound
2.3. LP/NLP Based Branch-and-Bound [Quesada & Grossmann]

- need access to MILP solver ... call back
  - exploit good MILP (branch-cut-price) solver
  - [Akrotirianakis&Rustem:00] use Gomory cuts in tree-search

- no commercial implementation of this idea
- preliminary results: order of magnitude faster than OA
  - same number of NLPs, but only one MILP
- similar ideas for Benders & Cutting Plane methods

... see [Quesada/Grossmann:92]
2.4. Integrating SQP & Branch-and-Bound

**AIM:** Avoid solving NLP node to convergence.

- Sequential Quadratic Programming (SQP)
  \[ (QP^k) \begin{aligned} &\text{minimize} \quad f^k + \nabla f^{kT} d + \frac{1}{2} d^T W^k d \\ &\text{subject to} \quad c^k + \nabla c^{kT} d \leq 0 \\ &\quad x^k + d_x \in X, \quad y^k + d_y \in \hat{Y}. \end{aligned} \]

- Early branching rule [Borchers & Mitchell:94]; after QP step:
  \[ \rightarrow \text{choose non-integral } y_{i+1}^k \text{ to branch on} \]
  \[ \rightarrow \text{branch and continue SQP on branch} \]
2.4. Integrating SQP & Branch-and-Bound

**SNAG**: \((QP^k)\) not lower bound
\[ \Rightarrow \text{no fathoming from upper bound} \Rightarrow \text{less efficient B&B} \]

\[
\begin{align*}
\text{minimize} & \quad f^k + \nabla f^k d + \frac{1}{2} d^T W^k d \\
\text{subject to} & \quad c^k + \nabla c^k d \leq 0 \\
& \quad x^k + d_x \in X, \ y^k + d_y \in \hat{Y}.
\end{align*}
\]
2.4. Integrating SQP & Branch-and-Bound

**Snag:** \((QP^k)\) not lower bound
⇒ no fathoming from upper bound ⇒ less efficient B&B

\[
\begin{align*}
\text{minimize} & \quad f^k + \nabla f^k T d + \frac{1}{2} d^T W^k d \\
\text{subject to} & \quad c^k + \nabla c^k T d \leq 0 \\
& \quad x^k + dx \in X, \ y^k + dy \in \hat{Y}.
\end{align*}
\]

**Remedy:** Exploit OA underestimating [L:01]:

- add objective cut \(f^k + \nabla f^k T d \leq U - \epsilon\) to \((QP^k)\)
- fathom node, if \((QP^k)\) inconsistent

⇒ convergence for convex MINLP
3. Modern Methods for MINLP

1. Branch-and-Cut
   - nonlinear cuts [Stubbs&Mehrotra:99]
   - linear cuts from OA [Akrotirianakis&Rustem:00]

2. Disjunctive Programming [Lee&Grossmann:99]

3. Parallel Tree Search Strategies
3.1. Nonlinear Branch-and-Cut [Mehrotra:99]

Consider MINLP

\[
\begin{align*}
\text{minimize} & \quad f^T_x x + f^T_y y \\
\text{subject to} & \quad c(x, y) \leq 0 \\
& \quad y \in \{0, 1\}, \ 0 \leq x \leq U
\end{align*}
\]

Linear objective

- important to exploit convex hull of constraints
- reformulate nonlinear objectives ...

\[
\min f(x, y) \iff \min \eta \text{ s.t. } \eta \geq f(x, y)
\]
Continuous relaxation ($z := (x, y)$):

\[
C := \{z|c(z) \leq 0, \ 0 \leq y \leq 1, \ 0 \leq x \leq U\}
\]

$C := \text{conv}(C)$ convex hull
Continuous relaxation ($z := (x, y)$):

$$
C' := \{ z | c(z) \leq 0, \ 0 \leq y \leq 1, \ 0 \leq x \leq U \}
$$

$$
C := \text{conv}(C') \text{ convex hull}
$$
Continuous relaxation \((z := (x, y))\):

\[
C := \{ z | c(z) \leq 0, \ 0 \leq y \leq 1, \ 0 \leq x \leq U \}
\]

\[
C := \text{conv}(C) \quad \text{convex hull}
\]

\[
C_{0/1}^j := \{ z \in C | y_j = 0/1 \}
\]

let \(M_j(C) := \left\{ \begin{array}{l} z = \lambda_0 u_0 + \lambda_1 u_1 \\ \lambda_0 + \lambda_1 = 1, \ \lambda_0, \lambda_1 \geq 0 \\ u_0 \in C_0^j, \ u_1 \in C_1^j \end{array} \right\} \)

\[
\Rightarrow P_j(C) := \text{projection of } M_j(C) \text{ onto } z
\]

\[= \text{conv} (C \cap y_j \in \{0, 1\}) \text{ and } P_{1...p}(C) = C\]
3.1. Nonlinear Branch-and-Cut [Mehrotra:99]

Given \( \hat{z} \) with \( \hat{y}_j \not\in \{0, 1\} \) find separating hyperplane

\[
\Rightarrow \begin{cases} 
\text{minimize} & \|z - \hat{z}\|_\infty \\
\text{subject to} & z \in \mathcal{P}_j(C)
\end{cases}
\]

convex reformulation of \( \mathcal{M}_j(C) \) with \( \mathcal{M}_j(\tilde{C}) \), where

\[
\tilde{C} := \left\{ (z, \mu) \middle| \begin{array}{c}
\mu c_i(z/\mu) \leq 0 \\
0 \leq \mu \leq 1 \\
0 \leq x \leq \mu U, \ 0 \leq y \leq \mu
\end{array} \right\}
\]

where \( c(0/0) = 0 \Rightarrow \) convex representation

\( \Rightarrow \) separating hyperplane: \( \psi^T (z - \hat{z}) \), where \( \psi \in \partial \|z - \hat{z}\|_\infty \)
3.1. Nonlinear Branch-and-Cut [Mehrotra:99]

- at each (?) node of Branch&Bound tree:
  - generate cutting planes

- generalize disjunctive approach from MILP
  - solve one convex NLP per cut

- generalizes Sherali/Adams and Lovaczi/Schrijver cuts
- tighten cuts by adding semi-definite constraint
3.2. Disjunctive Programming [Grossmann]

Consider disjunctive NLP

\[
\begin{align*}
\text{minimize} & \quad \sum_{k} f_k + f(x) \\
\text{subject to} & \quad \begin{bmatrix} Y_i \\ c_i(x) \leq 0 \end{bmatrix} \lor \begin{bmatrix} -Y_i \\ B_i x = 0 \end{bmatrix} \quad \forall i \in I \\
& \quad \begin{bmatrix} f_i = \gamma_i \\ f_i = 0 \end{bmatrix} \\
& \quad 0 \leq x \leq U, \ \Omega(Y) = \text{true}, \ Y \in \{\text{true, false}\}^p
\end{align*}
\]

Application: process synthesis
- \( Y_i \) represents presence/absence of units
- \( B_i x = 0 \) eliminates variables if unit absent

Exploit disjunctive structure
- special branching ... OA/GBD algorithms
3.2. Disjunctive Programming [Grossmann]

Consider disjunctive NLP

\[
\begin{align*}
\text{minimize} & \quad \sum_{k} f_k + f(x) \\
\text{subject to} & \quad \begin{bmatrix}
Y_i \\
c_i(x) \\
f_i = \gamma_i
\end{bmatrix} \lor \begin{bmatrix}
-Y_i \\
B_i x = 0 \\
f_i = 0
\end{bmatrix} \quad \forall i \in I \\
0 \leq x \leq U, \quad \Omega(Y) = \text{true}, \quad Y \in \{\text{true, false}\}^p
\end{align*}
\]

Big-M formulation (notoriously bad), \( M > 0 \):

\[
\begin{align*}
c_i(x) & \leq M (1 - y_i) \\
-M y_i & \leq B_i x \leq M y_i \\
f_i & = y_i \gamma_i \quad \Omega(Y) \text{ converted to linear inequalities}
\end{align*}
\]
3.2. Disjunctive Programming [Grossmann]

Consider disjunctive NLP

\[
\begin{align*}
\text{minimize} & \quad \sum_{k} f_k + f(x) \\
\text{subject to} & \quad \begin{bmatrix} Y_i \\ c_i(x) \leq 0 \\ f_i = \gamma_i \end{bmatrix} \lor \begin{bmatrix} -Y_i \\ B_i x = 0 \\ f_i = 0 \end{bmatrix} \quad \forall i \in I \\
& \quad 0 \leq x \leq U, \quad \Omega(Y) = \text{true}, \quad Y \in \{\text{true, false}\}^p
\end{align*}
\]

convex hull representation ...

\[
\begin{align*}
x &= v_{i1} + v_{i0}, & \lambda_{i1} + \lambda_{i0} &= 1 \\
\lambda_{i1} c_i(v_{i1}/\lambda_{i1}) &\leq 0, & B_i v_{i0} &= 0 \\
0 &\leq v_{ij} \leq \lambda_{ij} U, & 0 &\leq \lambda_{ij} \leq 1, & f_i &= \lambda_{i1} \gamma_i
\end{align*}
\]
3.2. Disjunctive Programming: Example

\[
\begin{bmatrix}
Y_1 \\
x_1^2 + x_2^2 \leq 1
\end{bmatrix}
\]

\[
\bigvee \begin{bmatrix}
Y_2 \\
(x_1 - 4)^2 + (x_2 - 1)^2 \leq 1
\end{bmatrix}
\]

\[
\bigvee \begin{bmatrix}
Y_3 \\
(x_1 - 2)^2 + (x_2 - 4)^2 \leq 1
\end{bmatrix}
\Rightarrow
\]
3.2. Disjunctive Programming & MPECs

Consider Fourier Transform Infrared (FTIR) Spectroscopy

Disjunction modeled with large $P_{\text{max}}$ parameter

$$0 \leq P \leq Y P_{\text{max}} \quad Y \in \{0, 1\}^{M \times N}$$

Either $P_{i,j} = 0$, or “count” parameter in objective

$$f(P, Y) = \sum e_k^T R^{-1} e_k + 2 \sum Y_{i,j}$$

Alternative model avoids integrality of $Y$

$$1 \geq Y_{i,j} \quad \perp \quad P_{i,j} \geq 0$$

where $\perp$ means orthogonality, i.e.

$$(1 - Y_{i,j}) P_{i,j} \leq 0 \quad \forall (i, j)$$

$\Rightarrow$ nonlinear constraint ... use NLP solvers (SQP)
3.2. Disjunctive Programming & MPECs

Small FTIR example: initial MPEC solution \( f = 25.98 \)

<table>
<thead>
<tr>
<th>NLPs</th>
<th>lower bound</th>
<th>upper bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8.4</td>
<td>( \infty )</td>
</tr>
<tr>
<td>30</td>
<td>8.4</td>
<td>250.0</td>
</tr>
<tr>
<td>75</td>
<td>9.9</td>
<td>99.2</td>
</tr>
<tr>
<td>100</td>
<td>11.2</td>
<td>26.8</td>
</tr>
<tr>
<td>155</td>
<td>12.3</td>
<td>14.0</td>
</tr>
</tbody>
</table>

\( \Rightarrow \) MPECs give good upper bound on MINLPs!

\( 0 \leq y \perp y \leq 1 \) not always good idea! \( \rightarrow \) need structure ...
3.3. Parallel Branch-and-Bound

**meta-computing platforms:**

- set of **distributed heterogeneous computers**, e.g.
  - pool of workstations
  - group of supercomputers or anything

⇒ **low quality** with respect to bandwidth, latency, availability
  - **low cost**: it’s free !!!
  - potentially **huge amount of resources**

... use *Condor* to “build” MetaComputer
... high-throughput computing
3.3. Parallel Branch-and-Bound

Master Worker Paradigm (MWdriver)
Object oriented C++ library
Runs on top of Condor-PVM

Fault tolerance via master check-pointing
3.3. Parallel Branch-and-Bound

**First Strategy:** 1 worker $\equiv$ 1 NLP

$\Rightarrow$ grain-size *too small*

... NLPs solve in seconds

**New Strategy:**

1 worker $\equiv$ 1 subtree (MINLP)

... “streamers” running down tree
3.3. Parallel Branch-and-Bound

Trimloss optimization with 56 general integers
⇒ solve 96,408 MINLPs on 62.7 workers
⇒ 600,518,018 NLPs

Wall clock time = 15.5 hours
Cumulative worker CPU time = 752.7 hours ≃ 31 days

\[
\text{efficiency} := \frac{\text{work-time}}{\text{work} \times \text{job-time}} = \frac{752.7}{62.7 \times 15.5} = 80.5
\]

... proportion of time workers were busy
3.3. Parallel Branch-and-Bound: Results

- Number of workers vs. elapsed time [h]
- Lower & upper bounds vs. elapsed time [h]
- Number of problems on stack vs. elapsed time [h]
- Number of NLPs per task vs. elapsed time [h]
4.1. Conclusions

- MINLP important modeling paradigm; many applications
  - MINLP most used solver on NEOS

- *Outer Approximation* et al.
  - rely heavily on convexity
  - readily exploit MILP structure in branch-and-cut

- *Branch-and-Bound*
  - works OK’ish for nonconvex problems (e.g. reload operation)
  - harder to exploit branch-and-cut ideas
4.1. Challenges

• Global solution of nonconvex MINLP, see Mohit’s talk
  ○ automatic code generation for underestimators (≡ AD)

• Connection to MPECs, recall Stefan’s talk
  ○ generate upper bounds along tree ...
  ○ global solution of MPECs using branch-and-cut

• PDE constraints & surrogate models
  ○ e.g. core reload operation
  ○ multi-model ... trust-regions ...