Development of the new MINLP Solver Decogo using SCIP - Status Report

Pavlo Muts
with Norman Breitfeld, Vitali Gintner, Ivo Nowak
SCIP Workshop 2018, Aachen
1. Introduction

2. Automatic decomposition

3. Adaptive Outer Approximation

4. Inner- and Outer-Refinement

5. Preliminary results with column generation

6. Final Remarks
Introduction
Motivation and Goal

Motivation:

1. **Branch&Bound** is currently the standard MINLP approach, but search-tree can grow rapidly
2. Experience with parallel **Column Generation (Inner Approximation)** for solving huge crew scheduling problems with 100,000,000 variables, but duality gap must be small
3. **Adaptive Outer Approximation (OA)** faster than B&B for some special nonconvex MINLPs, e.g. separable, but number of binary variables can be large

Goal:

Implementation of the nonconvex MINLP solver **Decogo** using:

1. **Automatic Decomposition** : for block-separable reformulation
2. **Adaptive OA**: for solving medium-scale MINLPs
3. **Parallel Inner/Outer Refinement**: for solving large-scale MINLPs
1. Energy system planning
   Modules: system components
   Subproblem: MINLP under uncertainty
   (project 2017-2020 with TU-Berlin)
1. Energy system planning
Modules: system components
Subproblem: MINLP under uncertainty
(project 2017-2020 with TU-Berlin)

2. Topology optimization
Modules: system components
(e.g. wing rib)
Subproblem: PDE-constrained MIQQP

Altair Hyperworks
Complex optimization problems with a modular structure

1. Energy system planning
   Modules: system components
   Subproblem: MINLP under uncertainty
   (project 2017-2020 with TU-Berlin)

2. Topology optimization
   Modules: system components
   (e.g. wing rib)
   Subproblem: PDE-constrained MIQQP
   Altair Hyperworks

3. Crew scheduling (LH Systems)
   Modules: crew members
   Subproblem: constrained shortest path
Decogo (Decomposition-based Global Optimizer)

- **Decogo** is written in Python and uses the modeling and optimization framework **Pyomo**.
- **Ipopt** is used for solving for NLPs, LPs and **SCIP** for MINLPs, MIPs.
Decogo (Decomposition-based Global Optimizer)

- **Decogo** is written in Python and uses the modeling and optimization framework **Pyomo**.
- **Ipopt** is used for solving for NLPs, LPs and **SCIP** for MINLPs, MIPs.
Decogo is developed in the project (2017-2020): **MINLP-Optimization of Design and Operation of Complex Energy Systems**

- HAW-Hamburg (V. Gintner, P. Muts, I. Nowak): development of Decogo
- TU-Berlin (S. Bruche, E. Papadis, G. Tsatsaronis): development of energy system models under uncertainty
- Mitsubishi Hitachi Power Systems Europe GmbH (A. Heberle): providing data and consultation

**References:**

- For more information see: www.researchgate.net/project/Decomposition-Methods-for-Mixed-Integer-Nonlinear-Optimization
Automatic decomposition
Goal: reformulation of a sparse optimization problem as a block-separable MINLP with arbitrary block-sizes

\[ \min \{ c^T x : x \in P \cap G \} \] , \quad G := \prod_{k \in K} G_k

with linear global constraints:

\[ P := \{ x \in \mathbb{R}^n : Ax \leq b \} \]

nonlinear local (nonconvex) constraints:

\[ G_k := \{ y \in \mathbb{R}^{n_k1} \times \mathbb{Z}^{n_k2} : g_j(y) \leq 0, j \in J_k \} \]
Basic steps

$X$ - set of variables, $K$ - number of blocks

1. Divide the set of variables $X$ into $K$ disjoint subsets with arbitrary size

2. Reformulate the problem by adding new variables with following requirements:
   - Objective function is linear
   - Global constraints are linear
   - Nonlinear constraints are local
$G = (V, E)$,
$V = \{v_i : v_i \in X\}$,
$E = \{e_{ij} : e_{ij} = (x_i, x_j), \frac{\partial^2 f_m}{\partial x_i \partial x_j} \neq 0\}$,

where $\frac{\partial^2 f_m}{\partial x_i \partial x_j}$ is Hessian of Lagrangian.
Blocks partition: connected components

\[ G = (V, E), \]
\[ V = \{v_i : v_i \in X\}, \]
\[ E = \{e_{ij} : e_{ij} = (x_i, x_j), \frac{\partial^2 f_m}{\partial x_i \partial x_j} \neq 0\}, \]
where \( \frac{\partial^2 f_m}{\partial x_i \partial x_j} \) is Hessian of Lagrangian.

Advantages:
- Easy implementation
- Less number of new variables for the reformulation

Disadvantage:
- Impossible to divide the set of variables with a given block size
New weighted graph (or hypergraph):

\[ G = (V, E), \ V = \{v_i : v_i \in X\}, \ E = \{e_{ij} : e_{ij} = (x_i, x_j), \frac{\partial^2 f_m}{\partial x_i \partial x_j} \neq 0\}, \]

\[ w(v_i) = W_i, \ W_i = \{w_i \in \mathbb{N} : \text{number of terms in } \frac{\partial^2 f_m}{\partial^2 x_i}\}, \]

\[ w(e_{ij}) = W_{ij}, \ W_{ij} = \{w_{ij} \in \mathbb{N} : \text{number of terms in } \frac{\partial^2 f_m}{\partial x_i \partial x_j}\}. \]

The weights are possibly multiplied by some factors.
New weighted graph (or hypergraph):

\[ G = (V, E), \ V = \{v_i : v_i \in X\}, \ E = \{e_{ij} : e_{ij} = (x_i, x_j), \frac{\partial^2 f_m}{\partial x_i \partial x_j} \neq 0\}, \]

\[ w(v_i) = W_i, \ W_i = \{w_i \in \mathbb{N} : \text{number of terms in} \ \frac{\partial^2 f_m}{\partial x_i^2}\}, \]

\[ w(e_{ij}) = W_{ij}, \ W_{ij} = \{w_{ij} \in \mathbb{N} : \text{number of terms in} \ \frac{\partial^2 f_m}{\partial x_i \partial x_j}\}. \]

The weights are possibly multiplied by some factors.

**Minimum k-cut partition:**

\[
\min \left[ \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{n:v_n \in c_i} w(e_{nm}) + \sum_{i=1}^{k} \sum_{v_j \in c_i} w(v_j) \right],
\]

where \( C = \{c_1, ..., c_k\} \) disjoint subsets of \( V \).
Adaptive Outer Approximation
Convex-Concave Reformulation and MIP Outer-Approximation

**Reformulation** of MINLP as a Convex-Concave Program (CCP) by replacing nonconvex constraint functions by a DC-formulation

\[ g(x) \leq 0 \iff g(x) + \sigma(\|x\|^2 - t) \leq 0, \quad t - \|x\|^2 \leq 0 \]

where the convexification parameters \( \sigma \) are updated dynamically
Convex-Concave Reformulation and MIP Outer-Approximation

**Reformulation** of MINLP as a *Convex-Concave Program (CCP)* by replacing nonconvex constraint functions by a *DC-formulation*

\[ g(x) \leq 0 \iff g(x) + \sigma(\|x\|_2^2 - t) \leq 0, \quad t - \|x\|_2^2 \leq 0 \]

where the convexification parameters \( \sigma \) are updated dynamically

**MIP outer approximation:**

- replace convex functions by linear approximations \( \rightarrow \mathcal{\tilde{C}} \)
- replace separable concave functions by piecewise linear functions \( \rightarrow \mathcal{\tilde{Q}}_B \)
- **MIPOA:**
  \[ \min c(x) : x \in \mathcal{\tilde{P}}, \quad (x, t) \in \mathcal{\tilde{C}} \cap \mathcal{\tilde{Q}}_B \]
Adaptive MIP based OA-Solver

The OA-Solver solves a CCP by successively updating MIPOAs using a limited number of breakpoints.
Adaptive MIP based OA-Solver

The OA-Solver solves a CCP by successively updating MIPOAs using a limited number of breakpoints

adaptSolveCCP

1. compute a solution of the MIPOA
2. project solution onto feasible set $G$
3. add linearization cuts to $\mathcal{C}$
4. update breakpoints of $\mathcal{Q}_B$ and convexification parameters $\sigma$

The OA-Solver is used for

- solving medium-scale MINLP-subproblems and for
- computing lower bounds of the (large-scale) original MINLP
Inner- and Outer-Refinement
CG successively improves the LP inner-approximation (IA) (restricted master problem)

\[ x_0 = \text{argmin}\{c^T x : x \in P \cap \text{conv}(S)\} \]

by adding points to a sample set \( S \subset G \) by solving MINLP subproblems.

The IA is equivalent to

\[
\min \{c^T x(z) : x(z) \in P, \; z \in \prod_{k \in K} \Delta_{|V_k|}\}, \quad x(z) := \sum_{v \in V_k} \hat{y}_{kv} z_{kv}
\]
CG successively improves the LP inner-approximation (IA) (restricted master problem)

\[ x_0 = \arg\min \{ c^T x : x \in P \cap \text{conv}(S) \} \]

by adding points to a sample set \( S \subset G \) by solving MINLP subproblems.

The IA is equivalent to

\[ \min \{ c^T x(z) : x(z) \in P, z \in \prod_{k \in K} \Delta_{|V_k|} \}, \quad x(z) := \sum_{v \in V_k} \hat{y}_{kv} z_{kv} \]

\( z_{kv} \) is called L-score.

\( z_{kv} \sim 1/\text{dist}(y_{kv}, x_{0k}) \), i.e. if \( z_{kv} \approx 1 \), then \( \hat{y}_{kv} \approx x_{0k} \)

**Graphical Illustration**

- **Left:** similar L-scores
- **Right:** different L-scores

Duality gap \( c^T x^* - c^T x_0 \) is smaller
LP inner and outer approximation

LP outer-approximation (OA)

\[ \min \{ c^T x : x \in \hat{P} \} \]

where \( \hat{P} \supset P \cap G \) is a polyhedron

CG solves the convex relaxation (CR):

\[ \min \{ c^T x : x \in P \cap \text{conv}(G) \} \]

and generates inner and outer approximations.
1. $\tilde{P} \leftarrow \text{autoDecomp}$
2. $\tilde{C} \leftarrow \text{initCCP}(\tilde{P})$
3. $(\tilde{P}, \tilde{C}) \leftarrow \text{reduceBox}(\tilde{P}, \tilde{C})$ \hspace{1cm} # reduce bounding box
4. $(z, S, \tilde{P}, \tilde{C}) \leftarrow \text{colGenStart}(\tilde{P}, \tilde{C})$ \hspace{1cm} # init IA/OA
5. $(v, \overline{v}, x^*) \leftarrow \text{outerMipHeu}(x(z), \tilde{P}, \tilde{C})$ \hspace{1cm} # compute sol.
6. Repeat
   6.1 $(\tilde{P}, \tilde{C}) \leftarrow \text{addOptCuts}(x^*, \tilde{P}, \tilde{C})$ \hspace{1cm} # cut off sol.
   6.2 $(z, S, \tilde{P}, \tilde{C}) \leftarrow \text{addLagCuts}(S, \tilde{P}, \tilde{C})$ \hspace{1cm} # improve IA/OA
   6.3 $(v, \overline{v}, x^*) \leftarrow \text{outerMipHeu}(x(z), \tilde{P}, \tilde{C})$ \hspace{1cm} # compute sol.
   6.4 Until $\overline{v} - v < \epsilon$ or stopping criterion
7. Return $(v, \overline{v}, x^*)$
Preliminary results with column generation
All experiments were performed on Windows-based computer with 16 GB RAM and Intel Core i7-7820HQ CPU running at 3.8 GHz

Reformulation:

- The problem has 3 blocks with size 9 in Hessian of Lagrangian.
- After reformulation: $K=4$ blocks with block sizes (21, 21, 21, 5).

Comparison of original problem and reformulated:

<table>
<thead>
<tr>
<th></th>
<th>original problem</th>
<th>after reformulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables</td>
<td>32</td>
<td>68</td>
</tr>
<tr>
<td>Linear constraints</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>Nonlinear constraints</td>
<td>11</td>
<td>36</td>
</tr>
</tbody>
</table>

\(^1\)http://www.minlplib.org/ex5_2_5.html
Example 1 from MINLPLib: pooling problem

Best known solutions:

<table>
<thead>
<tr>
<th>Primal bound</th>
<th>Dual bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3500</td>
<td>-3500.000004 (ANTIGONE)</td>
</tr>
<tr>
<td></td>
<td>-6233.265793 (BARON)</td>
</tr>
<tr>
<td></td>
<td>-7496.090988 (SCIP)</td>
</tr>
</tbody>
</table>

Our result:

- Estimate of the lower bound = \( \text{val}(\text{OA}) = -13330.3708 \) in 48s.
- Estimate of the upper bound = \( \text{val}(\text{IA}) = -3373.71873 \)
- Gap reported by \( \text{SCIP} = 8857.92 \)
- Convergence of \( \text{SCIP} \) is slow \( \Rightarrow \) termination after 10000 nodes
- \( \text{val}(\text{OA}) = -3671.3708 \) in \( \approx 9 \) hours.
Example 2 from MINLPLib: QCP\(^2\)

Reformulation:

- The problem has 16 blocks with average blocksize in Hessian of Lagrangian 7.5625.
- After reformulation: \(K=17\) blocks with average block size 23.

Comparison of original problem and reformulated:

<table>
<thead>
<tr>
<th></th>
<th>original problem</th>
<th>after reformulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of variables</td>
<td>126</td>
<td>391</td>
</tr>
<tr>
<td>Linear constraints</td>
<td>28</td>
<td>93</td>
</tr>
<tr>
<td>Nonlinear constraints</td>
<td>65</td>
<td>265</td>
</tr>
</tbody>
</table>

\(^2\)http://www.minlplib.org/ex8_3_8.html
### Example 2 from MINLPLib

**Best known solutions:**

<table>
<thead>
<tr>
<th>Primal bound</th>
<th>Dual bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3.25611909</td>
<td>-5.70013654 (ANTIGONE)</td>
</tr>
<tr>
<td></td>
<td>-10.00000000 (BARON)</td>
</tr>
<tr>
<td></td>
<td>-10.00000000 (SCIP)</td>
</tr>
</tbody>
</table>

**Our result:**

- Estimate of the lower bound $= \text{val(OA)} = -10.0$ in 369s.
- Estimate of the upper bound $= \text{val(IA)} = -9.6$
- Gap reported by SCIP $= 20676.41$
- Convergence of SCIP is slow $\Rightarrow$ termination after 10000 nodes
Final Remarks
Remarks

**Dior:**

- new exact MINLP approach, not based on B&B
- motivated by CG for huge transport optimization problems
- successive approximation method, which iteratively finds better points by improving inner- and outer-approximations
Remarks

**Dior:**

- new exact MINLP approach, not based on B&B
- motivated by CG for huge transport optimization problems
- successive approximation method, which iteratively finds better points by improving inner- and outer-approximations

**Advantages:**

- no search tree
- parallel solving (small difficult) sub-problems
- general approach for modular/sparse problems

Next steps:

- finish implementation of Decogo (new project)
- solve energy system planning and topology optimization problems
Dior:

- new exact MINLP approach, not based on B&B
- motivated by CG for huge transport optimization problems
- successive approximation method, which iteratively finds better points by improving inner- and outer-approximations

Advantages:

- no search tree
- parallel solving (small difficult) sub-problems
- general approach for modular/sparse problems

Next steps:

- finish implementation of Decogo (new project)
- solve energy system planning and topology optimization problems
Questions?