Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Technische Universität Berlin

pyADCG

A preliminary implementation of a new parallel solver for nonconvex MINLPs in Pyomo/Python

---

Ivo Nowak and Norman Breitfeld

ICMS 2016, Berlin

Table of contents

Introduction

Branch & Cut, Column Generation and ADCG

Branch & Cut:

- standard global solution approach for MIPs and MINLPs
- **However**: many branching steps necessary for large MINLPs (Benchmark Vigerske 2015, most problems have < 1000 variables)

Branch & Cut, Column Generation and ADCG

Branch & Cut:

- standard global solution approach for MIPs and MINLPs
- **However**: many branching steps necessary for large MINLPs (Benchmark Vigerske 2015, most problems have < 1000 variables)

Column Generation based perturbation heuristics (rapid branching):

- can solve crew scheduling problems with 100.000.000 variables
- **However**: duality gap should be small

Branch & Cut, Column Generation and ADCG

Branch & Cut:

- standard global solution approach for MIPs and MINLPs
- **However**: many branching steps necessary for large MINLPs (Benchmark Vigerske 2015, most problems have < 1000 variables)

Column Generation based perturbation heuristics (rapid branching):

- can solve crew scheduling problems with 100.000.000 variables
- **However**: duality gap should be small

ADCG (Alternating Direction Column Generation Method):

- exact parallel MINLP decomposition method
- combines AD and CG without using branching

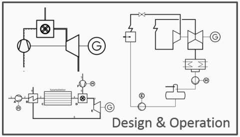
ADCG solves modular optimization problems

ADCG is a meta-solver, which uses sub-solvers for solving sub-problems.

ADCG solves modular optimization problems

ADCG is a meta-solver, which uses sub-solvers for solving sub-problems.

Planning (energy systems)



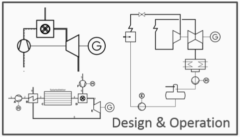
Engineering design (glider, Airbus)

sub-problems: MINLPs

ADCG solves modular optimization problems

ADCG is a meta-solver, which uses sub-solvers for solving sub-problems.

Planning (energy systems)



Engineering design (glider, Airbus)

sub-problems: MINLPs

Transport optimization (crew scheduling)

sub-problems: constrained shortest path

The quasi-separable MINLP

Nonconvex quasi-separable MINLP:

$$\min\{c^T x : x \in P \cap G\} \qquad \text{with} \quad G := \bigtimes_{k \in K} G_k$$

The quasi-separable MINLP

Nonconvex quasi-separable MINLP:

$$\min\{c^T x : x \in P \cap G\} \qquad \text{with} \quad G := \bigtimes_{k \in K} G_k$$

where G_k defines the feasible set of a (MINLP) sub-problem:

$$G_k := \{y \in [\underline{x}_{I_k}, \overline{x}_{I_k}] : y_i \in \{0,1\},\ i \in I_k^{\text{int}},\ g_j(y) \leq 0,\ j \in J_k\}$$

and $P := \{x \in [\underline{x}, \overline{x}] : \ Ax \leq b\}$ defines the coupling constraints

The quasi-separable MINLP

Nonconvex quasi-separable MINLP:

$$\min\{c^T x : x \in P \cap G\} \qquad \text{with} \quad G := \bigtimes_{k \in K} G_k$$

where G_k defines the feasible set of a (MINLP) sub-problem:

$$G_k := \{y \in [\underline{x}_{I_k}, \overline{x}_{I_k}] : y_i \in \{0,1\},\, i \in I_k^{\mathrm{int}},\, g_j(y) \leq 0,\, j \in J_k\}$$

and $P := \{x \in [\underline{x}, \overline{x}] : \; Ax \leq b\}$ defines the coupling constraints

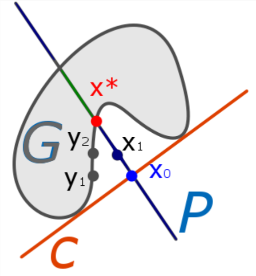
(MINLPs can be transformed into quasi-separable MINLPs by adding new variables and constraints)

Assumption: MINLP sub-problems can be solved quickly

The Alternating Direction Method

Basic steps of an ADM for solving $\min\{c^T x : x \in P \cap \bigtimes_{k\in K} G_k\}$:

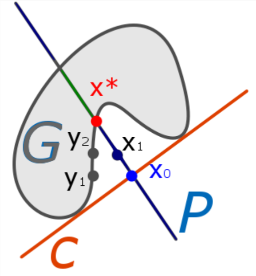
1. $y^{i+1} \leftarrow$ G-project $x^i \in P$ regarding $c + A^T\lambda^i$
2. $x^{i+1} \leftarrow$ P-project $y^{i+1} \in G$ regarding $c - A^T\lambda^i$
3. $\lambda^{i+1} \leftarrow$ update λ^i



The Alternating Direction Method

Basic steps of an ADM for solving $\min\{c^T x : x \in P \cap \bigtimes_{k\in K} G_k\}$:

1. $y^{i+1} \leftarrow$ G-project $x^i \in P$ regarding $c + A^T\lambda^i$
2. $x^{i+1} \leftarrow$ P-project $y^{i+1} \in G$ regarding $c - A^T\lambda^i$
3. $\lambda^{i+1} \leftarrow$ update λ^i



where

- G-project by (parallel) solving MINLP sub-problems
- P-project by solving a (large) QP master-problem
- the starting point $x^0 \in P$ is the solution of a convex relaxation

Traditional ADMs do not converge always towards the solution point x^*.

The Alternating Direction Column Generation Method

(Global optimization without branching)

Penalty function, full-projection problem, global convergence

Motivation: Houska, Frasch and Diehl, *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization* (2014), **ALADIN**:

- Check quality of x^i using the penalty function
  $PenF(x^i) := c^T x^i + \gamma \cdot Viol(x^i, P \cap G)$
  ($\rightarrow$ min $PenF(x) \Leftrightarrow$ MINLP)

Penalty function, full-projection problem, global convergence

Motivation: Houska, Frasch and Diehl, *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization* (2014), **ALADIN**:

- Check quality of x^i using the penalty function
  $PenF(x^i) := c^T x^i + \gamma \cdot Viol(x^i, P \cap G)$
  ($\rightarrow$ min $PenF(x) \Leftrightarrow$ MINLP)

- If $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
  solve approximately (via a dual line search)
  a full-projection problem ($x^i \in P \rightarrow P \cap G$)

$$
\begin{aligned}
x^{i+1} = \text{ approx argmin} \quad & c^T x + \rho \sum_{k \in K} \|x_{I_k} - x^i_{I_k}\|^2 \\
s.t. \quad & x \in G \cap P
\end{aligned}
$$

Penalty function, full-projection problem, global convergence

Motivation: Houska, Frasch and Diehl, *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization* (2014), **ALADIN**:

- Check quality of x^i using the penalty function
  $PenF(x^i) := c^T x^i + \gamma \cdot Viol(x^i, P \cap G)$
  $(\rightarrow \min PenF(x) \Leftrightarrow \text{MINLP})$

- If $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
  solve approximately (via a dual line search)
  a full-projection problem $(x^i \in P \rightarrow P \cap G)$

$$
\begin{aligned}
x^{i+1} = \text{ approx argmin} \quad & c^T x + \rho \sum_{k \in K} \|x_{I_k} - x^i_{I_k}\|^2 \\
s.t. \quad & x \in G \cap P
\end{aligned}
$$

Theorem: If ρ is large, then the full-projection problem has a zero duality gap $\rightarrow$ global convergence of ALADIN

ADCG: target constraint and CG-based full-projection

- Use a target constraint: $P' = \{x \in P : c^T x \leq tarVal\}$ for escaping from a non-global minimizer

ADCG: target constraint and CG-based full-projection

- Use a target constraint: $P' = \{x \in P : c^T x \leq tarVal\}$ for escaping from a non-global minimizer
- Solve the full-projection problem with slacks (approximately) by CG ($x^i \in P' \to P' \cap G$):

$$\begin{aligned} \min \quad & c^T x + \rho \sum_{k \in K} \|x_{I_k} - x^i_{I_k}\|^2 + \gamma \cdot s \\ s.t. \quad & x \in G \cap P \quad c^T x \leq tarVal + s, \quad s \geq 0 \end{aligned}$$

CG generates sample points $S_k \subseteq G_k$ (inner approximation)
$\to$ efficient warm-start possible

ADCG: target constraint and CG-based full-projection

- Use a target constraint: $P' = \{x \in P : c^T x \leq tarVal\}$
  for escaping from a non-global minimizer
- Solve the full-projection problem with slacks (approximately) by CG ($x^i \in P' \to P' \cap G$):

$$\begin{aligned} \min \quad & c^T x + \rho \sum_{k \in K} \|x_{I_k} - x^i_{I_k}\|^2 + \gamma \cdot s \\ s.t. \quad & x \in G \cap P \quad c^T x \leq tarVal + s, \quad s \geq 0 \end{aligned}$$

  CG generates sample points $S_k \subseteq G_k$ (inner approximation)
  $\to$ efficient warm-start possible
- ADCG performs CG-steps, until
  (i) $PenF$ is reduced sufficiently or
  (ii) $s + redCost(\lambda) > 0 \qquad \Rightarrow P' \cap G = \emptyset \Leftrightarrow tarVal < v^*$

The algorithm

ADCG (simplified):

1. compute $x^0 \in P$ using a start heuristic, $\mathit{tarVal} \leftarrow \infty$

The algorithm

ADCG (simplified):

1. compute $x^0 \in P$ using a start heuristic, $tarVal \leftarrow \infty$
2. **for** $i = 0, 1, \ldots$
   - 2.1 $y^{i+1} \leftarrow$ project x^i onto G by solving sub-problems (in parallel)
   - 2.2 $x^{i+1} \leftarrow$ project y^{i+1} onto P' by solving a QP master-problem
   - 2.3 **if** $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
     $(x^{i+1}, y^{i+1}) \leftarrow$ one CG-step (approx. full-project)

The algorithm

ADCG (simplified):

1. compute $x^0 \in P$ using a start heuristic, *tarVal* $\leftarrow \infty$
2. **for** $i = 0, 1, \ldots$
   2.1 $y^{i+1} \leftarrow$ project x^i onto G by solving sub-problems (in parallel)
   2.2 $x^{i+1} \leftarrow$ project y^{i+1} onto P' by solving a QP master-problem
   2.3 **if** $\mathit{PenF}(x^i) - \mathit{PenF}(x^{i+1})$ is not large enough:
   $(x^{i+1}, y^{i+1}) \leftarrow$ one CG-step (approx. full-project)
   if $s + \mathit{redCost}(\lambda) > 0$: $(\rightarrow \mathit{tarVal} = \text{lower bound})$
   increase *tarVal*

The algorithm

ADCG (simplified):

1. compute $x^0 \in P$ using a start heuristic, $tarVal \leftarrow \infty$
2. **for** $i = 0, 1, \ldots$
   2.1 $y^{i+1} \leftarrow$ project x^i onto G by solving sub-problems (in parallel)
   2.2 $x^{i+1} \leftarrow$ project y^{i+1} onto P' by solving a QP master-problem
   2.3 **if** $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
       $(x^{i+1}, y^{i+1}) \leftarrow$ one CG-step (approx. full-project)
       if $s + redCost(\lambda) > 0$: ($\rightarrow$ $tarVal$ = lower bound)
         increase $tarVal$
   2.4 **if** $\|x^{i+1} - y^{i+1}\| < \epsilon$: ($\rightarrow$ found local solution)
       decrease $tarVal$

The algorithm

ADCG (simplified):

1. compute $x^0 \in P$ using a start heuristic, $tarVal \leftarrow \infty$
2. **for** $i = 0, 1, \ldots$
   - 2.1 $y^{i+1} \leftarrow$ project x^i onto G by solving sub-problems (in parallel)
   - 2.2 $x^{i+1} \leftarrow$ project y^{i+1} onto P' by solving a QP master-problem
   - 2.3 **if** $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
     - $(x^{i+1}, y^{i+1}) \leftarrow$ one CG-step (approx. full-project)
     - **if** $s + redCost(\lambda) > 0$: ($\rightarrow tarVal =$ lower bound)
       - increase $tarVal$
   - 2.4 **if** $\|x^{i+1} - y^{i+1}\| < \epsilon$: ($\rightarrow$ found local solution)
     - decrease $tarVal$

Theorem: x^i converges towards a global solution of the MINLP (without branching)

Remarks

- ADCG is a proximal point method
  $\rightarrow$ trial-points in the neighborhood of the starting point

Remarks

- ADCG is a proximal point method
  $\rightarrow$ trial-points in the neighborhood of the starting point
- Based on point-based inner-approximations $S_k \subseteq G_k$
  $\rightarrow$ possible to use inexact sub-problem solutions, e.g. feasible points (of black-box models)

Remarks

- ADCG is a proximal point method
  $\rightarrow$ trial-points in the neighborhood of the starting point
- Based on point-based inner-approximations $S_k \subseteq G_k$
  $\rightarrow$ possible to use inexact sub-problem solutions, e.g. feasible points (of black-box models)

- CG-based start heuristic for starting point x^0 and local cuts

Remarks

- ADCG is a proximal point method
  $\rightarrow$ trial-points in the neighborhood of the starting point
- Based on point-based inner-approximations $S_k \subseteq G_k$
  $\rightarrow$ possible to use inexact sub-problem solutions, e.g. feasible points (of black-box models)

- CG-based start heuristic for starting point x^0 and local cuts

Remarks

- ADCG is a proximal point method
  $\rightarrow$ trial-points in the neighborhood of the starting point
- Based on point-based inner-approximations $S_k \subseteq G_k$
  $\rightarrow$ possible to use inexact sub-problem solutions, e.g. feasible points (of black-box models)

- CG-based start heuristic for starting point x^0 and local cuts

For details of ADCG see:

1. Nowak. *Column Generation based Alternating Direction Methods for solving MINLPs.* 2015 (Optimization Online)

2. Nowak, Breitfeld, *The Alternating Direction Column Generation Method (Global Optimization without Branching)* (in preparation)

Preliminary results with pyADCG

pyADCG

Implementation:

- object-oriented implementation in Python/Pyomo
- development started in beginning of 2016
- not finished, currently only for nonconvex QQPs

pyADCG

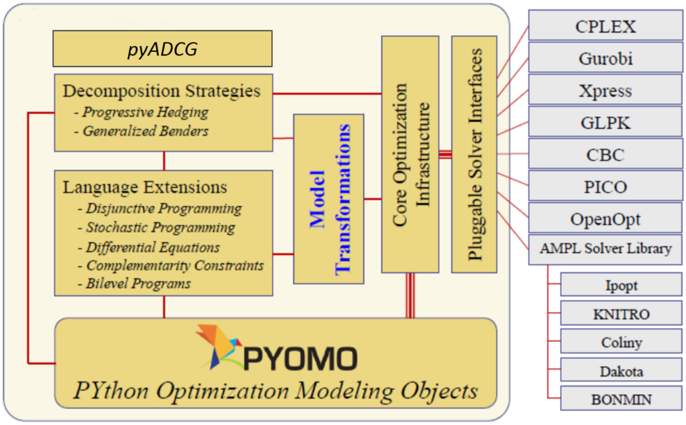
Implementation:

- object-oriented implementation in Python/Pyomo
- development started in beginning of 2016
- not finished, currently only for nonconvex QQPs

pyADCG is a meta-solver consisting of the sub-solvers:

- IpOpt for master-problems and local optimization
- SCIP for MINLP sub-problems

What is Pyomo?



Pyomo provides a modeling language and supports the development of meta-solvers based on sub-solvers via Python ($\rightarrow$ fast development)

Current status

Preliminary numerical results with random QQPs
(without parallelization):

- very few ADCG iterations (maybe because duality gap is small)
- time for solving master-problems 1%
- time for solving sub-problems 94%

$\Rightarrow$ significant potential for parallel solving the sub problems

- time for Python/Pyomo operations 5%

$\Rightarrow$ little influence of the implementation language

Current status

Preliminary numerical results with random QQPs
(without parallelization):

- very few ADCG iterations (maybe because duality gap is small)
- time for solving master-problems 1%
- time for solving sub-problems 94%

⇒ significant potential for parallel solving the sub problems

- time for Python/Pyomo operations 5%

⇒ little influence of the implementation language

Planned:

- solving MINLPlib2 models (translated to Pyomo) in development
- automatic decomposition and other enhancements
- application to energy conversion systems

Questions?