DECOGO

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

Motivation

1. Parallel Column Generation (CG) (global optimization)
   Experience with very large crew scheduling problems:
   - 100.000.000 variables, but small duality gap
   - Good solutions with a perturbation heuristic (Rapid branching)

Motivation

1. Parallel Column Generation (CG) (global optimization)
   Experience with very large crew scheduling problems:
   - 100.000.000 variables, but small duality gap
   - Good solutions with a perturbation heuristic (Rapid branching)
2. Parallel Alternating Direction (AD) (local optimization):
   Houska, Frasch and Diehl, *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization* (2014)
   Newton-type decomposition-method ($\rightarrow$ fast convergence)

Motivation

1. Parallel Column Generation (CG) (global optimization)
   Experience with very large crew scheduling problems:
   - 100.000.000 variables, but small duality gap
   - Good solutions with a perturbation heuristic (Rapid branching)
2. Parallel Alternating Direction (AD) (local optimization):
   Houska, Frasch and Diehl, *An Augmented Lagrangian based Algorithm for Distributed Non-Convex Optimization* (2014)
   Newton-type decomposition-method ($\rightarrow$ fast convergence)
3. Branch&Bound (B&B)
   - Tree-based search method ($\rightarrow$ many branching steps)
   - Current B&B-solvers solve problems up to $<$ 1000 variables (MINLPlib2 Benchmark Vigerske 2015)
   - ex5_2_5 example from MINLPlib2: bilinear, 33 variables,
     solvers: ANTIGONE, BARON, COUENNE, LINDO and SCIP
     all solutions are different with objective values in [-3500, -7629] !!

GO without B&B

Goal: Decomposition method which is not based on B&B

GO without B&B

Goal: Decomposition method which is not based on B&B

Generate&Refine (G&R): Parallel decomposition method combining

1. Column-Generation $\to$ Inner/Outer-Approximation (IA/OA)
2. Alternating Direction (Predictor) $\to$ local solution
3. IA/OA Refinement (Corrector) $\to$ global solution

GO without B&B

Goal: Decomposition method which is not based on B&B

Generate&Refine (G&R): Parallel decomposition method combining

1. Column-Generation $\rightarrow$ Inner/Outer-Approximation (IA/OA)
2. Alternating Direction (Predictor) $\rightarrow$ local solution
3. IA/OA Refinement (Corrector) $\rightarrow$ global solution

DECOGO (Decomposition-based Global Optimizer)

- Python framework for experimenting with decomposition methods, CG, AD and Piecewise-Linear (PL) OA methods
- Meta-solver using black-box sub-solvers
  (Assumption: sub-solvers can solve sub-problems quickly)
- development started in 2016, not finished

Comparison of exact GO approaches

method	approximation	improvement
Branch&Cut	LP-OA	branching, OA-refine
Branch&Price	IA	branching, IA-refine
Branch&Refine	MIP-OA	branching, OA-refine
Sequential MI(NL)P	MI(NL)P-OA	OA-refine
Generate&Refine	IA, MIP-OA	IA/OA-refine

Block-Separable MINLPs (Sparse Optimization Problems)

Block-separable (modular) MINLP:

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

Block-Separable MINLPs (Sparse Optimization Problems)

Block-separable (modular) MINLP:

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

with linear global constraints: $P := \{x \in \mathbb{R}^n : Ax \leq b\}$ (polytope)

and (nonconvex) nonlinear local constraints:

$$G_k := \{y \in \mathbb{R}^{n_k} : y_i \in \{0,1\},\, i \in I_k^{\mathrm{int}},\, g_j(y) \leq 0,\, j \in J_k\}$$

Block-Separable MINLPs (Sparse Optimization Problems)

Block-separable (modular) MINLP:

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

with linear global constraints: $P := \{x \in \mathbb{R}^n : Ax \le b\}$ (polytope)

and (nonconvex) nonlinear local constraints:

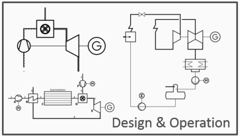
$$G_k := \{y \in \mathbb{R}^{n_k} : y_i \in \{0,1\}, i \in I_k^{\mathrm{int}}, g_j(y) \le 0, j \in J_k\}$$

- Almost all MINLPs of the MINLPlib2 are block-separable with small block-sizes ($n_k \le 10$)
- The block-size n_k can be reduced by adding new variables and constraints

Examples for Modular/Sparse Optimization Problems

Examples for Modular/Sparse Optimization Problems

Planning (energy systems)

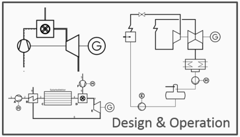


Engineering design (glider, Airbus)

Sub-problems: MINLPs

Examples for Modular/Sparse Optimization Problems

Planning (energy systems)



Engineering design (glider, Airbus)

Sub-problems: MINLPs

Transport optimization (crew scheduling)

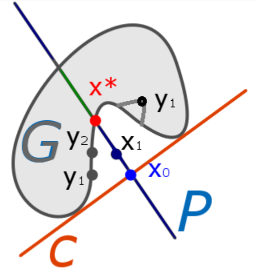
Sub-problems: constrained shortest path

The Alternating-Direction Column-Generation Method

Alternating Direction Method (AD) for Local Optimization

Basic steps of traditional AD for solving $\min\{c^T x : x \in P \cap G\}$:

1. $y^{i+1} = \operatorname{argmin}\{L^G_{x^i,\lambda^i}(y) : y \in G\}$ (G-project)
2. $x^{i+1} = \operatorname{argmin}\{L^P_{y^{i+1},\lambda^i}(x) : x \in P\}$ (P-project)
3. $\lambda^{i+1} \leftarrow$ update λ^i

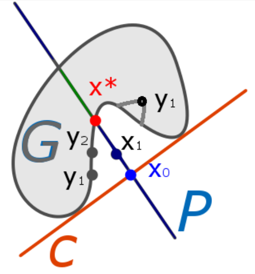


regarding the augmented Lagrange-functions:
$L^G_{x^i,\lambda^i}(y) := (c + A^T\lambda^i)^T y + \rho \sum_{k\in K} \|y_{I_k} - x^i_{I_k}\|^2$ and
$L^P_{y^{i+1},\lambda^i}(x) := (c - A^T\lambda^i)^T x + \rho \sum_{k\in K} \|x_{I_k} - y^{i+1}_{I_k}\|^2$

Alternating Direction Method (AD) for Local Optimization

Basic steps of traditional AD for solving $\min\{c^T x : x \in P \cap G\}$:

1. $y^{i+1} = \text{argmin}\{L^G_{x^i,\lambda^i}(y) : y \in G\}$ (G-project)
2. $x^{i+1} = \text{argmin}\{L^P_{y^{i+1},\lambda^i}(x) : x \in P\}$ (P-project)
3. $\lambda^{i+1} \leftarrow$ update λ^i



regarding the augmented Lagrange-functions:
$L^G_{x^i,\lambda^i}(y) := (c + A^T\lambda^i)^T y + \rho \sum_{k\in K} \|y_{I_k} - x^i_{I_k}\|^2$ and
$L^P_{y^{i+1},\lambda^i}(x) := (c - A^T\lambda^i)^T x + \rho \sum_{k\in K} \|x_{I_k} - y^{i+1}_{I_k}\|^2$

Traditional AD does not converge always towards the solution point x^*.

(AD is similar to Progressive Hedging in stochastic programming)

Target-projection based AD

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

- In order to enforce convergence, evaluate the penalty function $PenF(x^i) := c^T x^i + \gamma \cdot (Viol(x^i, P) + Viol(x^i, G))$ (minimizing $PenF(x) \Leftrightarrow$ solving MINLP) $(Viol(x^i, P) = slackSum(x^i))$

Target-projection based AD

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

- In order to enforce convergence, evaluate the penalty function $PenF(x^i) := c^T x^i + \gamma \cdot (Viol(x^i, P) + Viol(x^i, G))$ (minimizing $PenF(x) \Leftrightarrow$ solving MINLP) $(Viol(x^i, P) = slackSum(x^i))$

- If $PenF(x^i) - PenF(x^{i+1})$ is not large enough: make dual line search for Target-Projection Problem (TPP) :

$$x^{i+1} = \text{ approx argmin}\{c^T y + \rho \sum_{k \in K} \|y_{I_k} - x^i_{I_k}\|^2 : y \in G \cap P\}$$

Target-projection based AD

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

- In order to enforce convergence, evaluate the penalty function $PenF(x^i) := c^T x^i + \gamma \cdot (Viol(x^i, P) + Viol(x^i, G))$
(minimizing $PenF(x) \Leftrightarrow$ solving MINLP)
($Viol(x^i, P) = slackSum(x^i)$)

- If $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
make dual line search for Target-Projection Problem (TPP) :

$$x^{i+1} = \text{ approx argmin}\{c^T y + \rho \sum_{k \in K} \|y_{I_k} - x^i_{I_k}\|^2 : y \in G \cap P\}$$

Theorem: If ρ is large, then (TPP) has a zero duality gap in a neighborhood $U(x^{OPT}) \to$ (quadratic) convergence of AD

The Alternating-Direction Column-Generation Method

(TPP2) (formulation with a penalty function):

$$\min\{PenF(y) + \rho \sum_{k \in K} \|y_{I_k} - x^i_{I_k}\|^2 : y \in G^{OA} \cap P\}$$

$\rightarrow$ easy to generate points $y_k \in G_k^{OA}$

(TPP2) has a zero duality gap, if $x^i \in U(x^{OPT}) \rightarrow$ can be solved by CG

The Alternating-Direction Column-Generation Method

(TPP2) (formulation with a penalty function):

$$\min\{PenF(y) + \rho \sum_{k \in K} \|y_{I_k} - x^i_{I_k}\|^2 : y \in G^{OA} \cap P\}$$

$\rightarrow$ easy to generate points $y_k \in G_k^{OA}$

(TPP2) has a zero duality gap, if $x^i \in U(x^{OPT}) \rightarrow$ can be solved by CG

ADCG-LocalOpt(x^0):

1. $i \leftarrow 0$ and **repeat**:
2. $y^{i+1} = \text{argmin}\{L^G_{x^i,\lambda^i}(y) : y \in G\}$ (in parallel)
3. $x^{i+1} = \text{argmin}\{L^P_{y^{i+1},\lambda^i}(x) : x \in P\}$
4. **if** $PenF(x^i) - PenF(x^{i+1})$ is not large enough:
   $(x^{i+1}, y^{i+1}) \leftarrow$ CG-steps of (TPP2)
5. **if** $\|x^{i+1} - y^{i+1}\| < \epsilon$: stop, $x^{OPT} \leftarrow x^{i+1}$
   if no new points generated: stop $\rightarrow x^i \notin U(x^{OPT})$

Details in: I. Nowak, Column Generation based Alternating Direction Methods for solving MINLPs. 2015 (Optimization Online)

The Generate & Refine Method (GO without B&B)

Basic Steps of Generate&Refine

Generate (Predictor)

1. CG: generate an Inner-Approximation $G^{IA} \subset G$ and
   a (polyhedral) Outer-Approximations $G^{OA} \supset G$
   $x^{IA} := \text{argmin}\{c^T x : x \in P \cap G^{IA}\}$
   $x^{OA} := \text{argmin}\{c^T x : x \in P \cap G^{OA}\}$
2. ADCG: find local opt. x^{OPT} starting from x^i =Generate&Fix(x^{IA})
   (fails if $x^i \notin U(x^{OPT})$)

Basic Steps of Generate&Refine

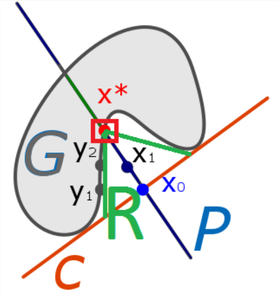
Generate (Predictor)

1. CG: generate an Inner-Approximation $G^{IA} \subset G$ and a (polyhedral) Outer-Approximations $G^{OA} \supset G$
   $x^{IA} := \text{argmin}\{c^T x : x \in P \cap G^{IA}\}$
   $x^{OA} := \text{argmin}\{c^T x : x \in P \cap G^{OA}\}$
2. ADCG: find local opt. x^{OPT} starting from x^i =Generate&Fix(x^{IA}) (fails if $x^i \notin U(x^{OPT})$)

Refine (Corrector)

1. IA-Refine: if ADCG fails, improve x^i by generating new points near x^i and $x^{i+1} \leftarrow$ Generate&Fix(x^{IA})
2. OA-Refine: if IA-Refine fails, set $G_k^{OA} \leftarrow G_k^{OA} \setminus R_k$ and $x^{i+1} \leftarrow x^{OA}$ ($\rightarrow$ reduce gap$:= c^T x^* - c^T x^{OA}$)

$\rightarrow x^{OPT} = x^*$ in finitely many steps



CG for generating Inner- and Outer Approximations

CG: Solve alternately **LP-master-problem** with slacks:

$$x^{CG} = \text{argmin}\{c^T x + e^T s : x \in P_s \cap \text{conv}(G^{IA})\}, \quad \text{where}$$

$P_s := \{x \in [\underline{x}, \overline{x}] : \ Ax \le b + s\}$ and $G^{IA} := \prod_{k \in K} G_k^{IA} \subseteq G$

CG for generating Inner- and Outer Approximations

CG: Solve alternately **LP-master-problem** with slacks:

$$x^{CG} = \operatorname{argmin}\{c^T x + e^T s : x \in P_s \cap \operatorname{conv}(G^{IA})\}, \quad \text{where}$$

$P_s := \{x \in [\underline{x}, \overline{x}] : \; Ax \leq b + s\}$ and $G^{IA} := \prod_{k \in K} G_k^{IA} \subseteq G$

and **MINLP sub-problems** :

$$y_k^i = \operatorname{argmin}\{(d_k^i)^T y : \; y \in G_k\}, \quad k \in K$$

regarding search directions

- $d_k^i = \pm c_{I_k}$ for initializing G_k^{IA}
- $d_k^i = A_{k,j}$ for eliminating slacks $s_j > 0$
- $d_k^i = c_{I_k} + A_k^T \lambda$ for solving the Lagrangian relaxation

The result is an IA defined by points $G_k^{IA} := \{y_k^i\}_{i \in I}$
and an OA defined by cuts: $(d_k^i)^T y \geq (d_k^i)^T y_k^i$

IA-Refinement using Generate&Fix

If $x^i \notin U(x^{OPT})$ **or** $x^i = x^{OPT}$:

Find a trial point x^{i+1} near x^i with $PenF(x^{i+1}) < PenF(x^i)$

by generating point sets $T_k(x^i) \subset G_k^{OA} \cap U(x^i)$

IA-Refinement using Generate&Fix

If $x^i \notin U(x^{OPT})$ **or** $x^i = x^{OPT}$:
Find a trial point x^{i+1} near x^i with $PenF(x^{i+1}) < PenF(x^i)$
by generating point sets $T_k(x^i) \subset G_k^{OA} \cap U(x^i)$

Generate&Fix: (successive fixing and generating points)

1. $K_{fix} \leftarrow \emptyset$
2. solve MIP master problem

$$x^{i+1} \leftarrow \operatorname{argmin}\{PenF(x) : x \in P \cap \prod_{k \in K_{fix}} G_k^{IA} \times \prod_{k \in K \setminus K_{fix}} \operatorname{conv}(G_k^{IA})\}$$

3. **if** $x^{i+1} \in G^{IA}$: stop
4. choose $k \in K \setminus K_{fix}$ with $x_k^{i+1} \notin G_k^{IA}$, $K_{fix} \leftarrow K_{fix} \cup \{k\}$
5. generate $T_k(x^i)$, $G_k^{IA} \leftarrow G_k^{IA} \cup T_k(x^i)$ and goto 2

(In crew scheduling $T_k(x^i)$ is defined by variations of a duty plan x^i)

Generate&Fix Strategies

Large Neighborhood Search (Target-space enumeration):
Enumerate new points in $G_k^{OA} \cap U(x^i)$
After some experiments:

$$T_k(x^i) := \{x_k^i + e_j d_{k,j} + e_l d_{k,l}, \quad j, l \in [n_k]\}$$

where $d_k := x_k^{max} - x_k^i$ with $x_k^{max} = \mathrm{argmax}\{L_k(y, \lambda) : y \in G_k^{IA}\}$
(More experiments needed)

Generate&Fix Strategies

Large Neighborhood Search (Target-space enumeration):
Enumerate new points in $G_k^{OA} \cap U(x^i)$
After some experiments:

$$T_k(x^i) := \{x_k^i + e_j d_{k,j} + e_l d_{k,l}, \quad j, l \in [n_k]\}$$

where $d_k := x_k^{max} - x_k^i$ with $x_k^{max} = \text{argmax}\{L_k(y, \lambda) : y \in G_k^{IA}\}$
(More experiments needed)

Rapid-Branching (Perturbation-Fixing) (only if duality gap is small)

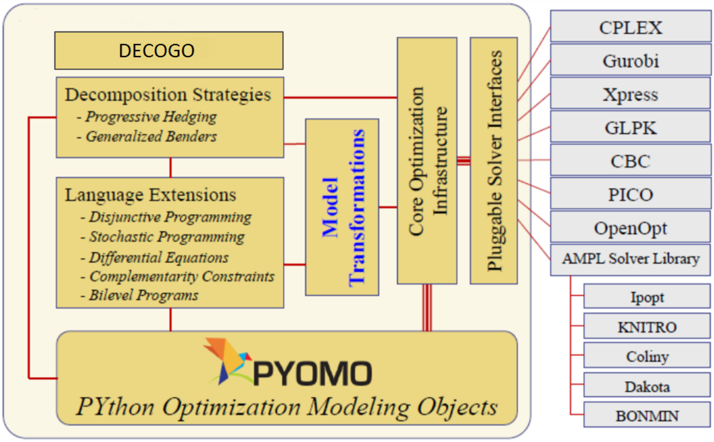
1. **if** $dist(x_k^{i+1}, G_k^{IA})$ is small, soft-fix x_k^{i+1} by reducing its column cost
2. **if** after soft-fixing the objective value increases strongly, generate new points

DECOGO (Decomposition-based Global Optimizer)

DECOGO - Status

- **DECOGO**(**DECO**mposition-based **G**lobal **O**ptimizer) is an object-oriented implementation of **Generate&Refine** in Python/Pyomo
- Meta-solver consisting of sub-solvers:
  - IpOpt for master-problems and local optimization
  - SCIP for MINLP sub-problems
- Development started in beginning of 2016
- Not finished
  - Only for nonconvex QQPs
  - Generate&Fix partly implemented, no OA-refinement

Pyomo



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

Preliminary Results

Time measurements with random QQPs

- 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

Preliminary Results

Time measurements with random QQPs

- 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

Small test examples from MINLPlib2 with large duality gap

- Perturbation-Fixing does not work (because of gap)
- CG+AD +search-space enumeration is fast, but does not generate a global solution for all examples
- Small blocks easier than big blocks $\rightarrow$ automatic decomposition

Final Remarks

Remarks

Generate&Refine:

- new exact GO approach, not based on B&B
- motivated by CG for large-scale transport optimization
- proximal-point method, which successively finds better points by improving inner- and outer-approximations

Remarks

Generate&Refine:

- new exact GO approach, not based on B&B
- motivated by CG for large-scale transport optimization
- proximal-point method, which successively finds better points by improving inner- and outer-approximations

Advantages:

- early solution candidates and parallel sub-problem solving
- general approach for modular/sparse problems
- number of nonconvex cuts in OA is related to visited local minimizers

Remarks

Generate&Refine:

- new exact GO approach, not based on B&B
- motivated by CG for large-scale transport optimization
- proximal-point method, which successively finds better points by improving inner- and outer-approximations

Advantages:

- early solution candidates and parallel sub-problem solving
- general approach for modular/sparse problems
- number of nonconvex cuts in OA is related to visited local minimizers

Next steps:

- efficient search-space enumeration method: exhaustive search in low dimensional sub-spaces (similar to path enumeration using DP)
- finish a preliminary version of DECOGO

Questions?