Decomposition-based Successive Approximation Methods for Global Optimization

Ivo Nowak[1,a)] and Pavlo Muts[1,b)]

[1]*Hamburg University of Applied Sciences, Berliner Tor 21, 20099 Hamburg, Germany*

[a)]Corresponding author: ivo.nowak@haw-hamburg.de
[b)]pavlo.muts@haw-hamburg.de

Abstract. Traditional deterministic global optimization methods are often based on a Branch-and-Bound (BB) search tree, which may grow rapidly, preventing the method to find a good solution. Motivated by decomposition-based inner approximation (column generation) methods for solving transport scheduling problems with over 100 million variables, we present two new deterministic decomposition-based successive approximation methods for general modular and/or sparse MINLPs. The first algorithm, called DECOA, is a successive MIP-outer-approximation algorithm based on refining nonconvex polyhedral underestimators of nonlinear functions. The second algorithm, called DIOR, is based on successively improving inner and outer approximations by solving column and cut generation sub-problems using DECOA. Both algorithms are part of Decogo, a new parallel MINLP solver. We describe the basic ideas of both algorithms, and present numerical results with Decogo for instances of the MINLPlib2.

INTRODUCTION

We assume that a MINLP is reformulated as a *block-separable* optimization problem, of the form

$$\min\ c^T x \quad \text{s.t.} \quad x \in P,\ x_k \in G_k,\ k \in K \tag{1}$$

with

$$P := \{x \in [\underline{x}, \overline{x}] :\ a_j^T x \le b_j, j \in J\} \quad \text{and} \quad G_k := G_k^{NLP} \cap G_k^{LP} \cap G_k^{INT}$$

where the n variables are partitioned into $|K|$ blocks of dimension n_k, and $x_k \in \mathbb{R}^{n_k}$ denotes the variables of the k-th block. The constraints defining G_k are called *local*, and the constraints defining P are called *global*. The linear objective function is defined by $c \in \mathbb{R}^n$, and the global linear constraints are defined by $a_j \in \mathbb{R}^n$ and $b_j \in \mathbb{R}$ for $j \in J$. The nonlinear, linear and mixed-integer local constraint sets are defined by $G_k^{NLP} := \{y \in \mathbb{R}^{n_k} : g_{kj}(y) \le 0,\ j \in [m_k]\}$ with $g_{kj} : \mathbb{R}^{n_k} \to \mathbb{R}$, $G_k^{LP} := \{y \in [\underline{x}_k, \overline{x}_k] :\ a_{kj}^T y \le b_{kj}, j \in J_k\}$, with $a_{kj} \in \mathbb{R}^{n_k}$, $b_{kj} \in \mathbb{R}$, and $G_k^{INT} := \mathbb{R}^{n_{k1}} \times \mathbb{Z}^{n_{k2}}$, respectively. The vectors $\underline{x}_k, \overline{x}_k \in \mathbb{R}^{n_k}$ determine the lower and upper bounds on the variables. Note that if some variable bounds are not finite and G_k is bounded, finite lower and upper bounds $\underline{x}_k, \overline{x}_k \in \mathbb{R}^{n_k}$ can be computed. The restriction to inequality constraints is only for notational simplicity. Furthermore, we use the notation $G := \prod_{k \in K} G_k$.

Most exact algorithms for solving the MINLP (1) are based on the spatial branch-and-bound approach and variants like branch-cut-and-price [5] or branch-decompose-and-cut [10], see [4, 3] for an overview of MINLP-solvers. A main difficulty of this approach is a possibly rapidly growing branch-and-bound tree, which makes it difficult to solve large-scale models in reasonable time.

Motivated by the excellent performance of the Column Generation (CG) based Rapid Branching (RB) approach [2, 8] for solving huge network optimization problems with small duality gaps, the Decomposition-based Inner and Outer Refinement (DIOR) algorithm was proposed in [9], which does not use a single potentially large BB search tree. In this paper, a variant of this approach is presented. It uses the new Decomposition-based Outer Approximation (DECOA) algorithm for quickly solving MINLP sub-problems, in order to compute solution candidates, generate cuts and reduce the duality gap.

THE MIP OUTER APPROXIMATION PROBLEM

DECOA successively refines a MIP Outer Approximation (MIPOA) problem by adding cuts and refining nonconvex polyhedral underestimators, presented in [9], of nonlinear functions. A MIPOA is defined by

$$\min c^T x \quad \text{s.t.} \quad x \in \check{P} \cap \check{G}, \tag{2}$$

where $\check{P}$ and $\check{G} := \prod_{k\in K} \check{G}_k$ are convex and nonconvex polyhedral approximations, respectively, defined by

$$\check{P} := \{x \in [\underline{x}, \overline{x}] : A_j x \le b_j,\, j \in \check{J}\} \quad \text{and} \quad \check{G}_k := \{y \in G_k^{INT} \cap \check{G}_k^{LP} : \breve{g}_{kj}(y) \le 0,\, j \in J_k\}$$

with $\check{G}_k^{LP} := \{y \in [\underline{x}_k, \overline{x}_k] : a_{kj}^T y \le b_{kj},\, j \in \check{J}_k\}$, $\check{J} \supseteq J$, $\check{J}_k \supseteq J_k$, and $\breve{g}_{kj}$ denotes a piecewise linear underestimator of the nonlinear function g_{kj}, which is constructed using the DC formulation (Difference of Convex functions)

$$g_{kj}(x) = h_{kj}(x) - \sigma_{kj} \sum_{i\in I_{kj}} q_{ki}(x),$$

defined by the convexified nonlinear and quadratic functions

$$h_{kj}(y) := g_{kj}(y) + \sigma_{kj} \sum_{i\in[n_k]} q_{ki}(y_i) \quad \text{and} \quad q_{ki}(y_i) := (y_i - \underline{x}_{ki})(y_i - \overline{x}_{ki}). \tag{3}$$

The convexification parameters are computed by $\sigma_{kj} = \max\{0, -v_{kj}\}$ and v_{kj} is a lower bound of the optimimal value of the nonlinear eigenvalue problem

$$\min y^T H_{kj}(x)\, y \quad \text{s.t.} \quad x \in [\underline{x}_k, \overline{x}_k],\; y \in \mathbb{R}^{n_k},\; \|y\|^2 = 1 \tag{4}$$

with $H_{kj} = \nabla^2 g_{kj}$.

A convex polyhedral underestimator of h_{kj} is defined by

$$\check{h}_{kj}(y) = \max_{\hat{y}\in T_k} \bar{h}_{kj,\hat{y}}(y), \qquad \text{where} \qquad \bar{h}_{kj,\hat{y}}(y) := h_{kj}(\hat{y}) + \nabla h_{kj}(\hat{y})^T (y - \hat{y}), \tag{5}$$

regarding a sample set $T_k \subset \check{G}_k^{LP}$. A piecewise linear overestimator of q_{ki} is defined by

$$\check{q}_{ki}(y) := q_{ki}(\hat{y}_{ki,v}) \frac{\hat{y}_{ki,v+1} - y}{\hat{y}_{ki,v+1} - \hat{y}_{ki,v}} + q_{ki}(\hat{y}_{ki,v+1}) \frac{y - \hat{y}_{ki,v}}{\hat{y}_{ki,v+1} - \hat{y}_{ki,v}}$$

for $y \in [\hat{y}_{ki,v}, \hat{y}_{ki,v+1}]$, $v \in \{1, \ldots, |B_{ki}|-1\}$, regarding breakpoints $B_{ki} := \{\hat{y}_{ki,1}, \ldots, \hat{y}_{ki,|B_{ki}|}\}$. Then a nonconvex polyhedral underestimator $\breve{g}_{kj}$ of g_{kj} is given by

$$\breve{g}_{kj}(x) := \check{h}_{kj}(x) - \sigma_{kj} \sum_{i\in I_{kj}} \check{q}_{ki}(x).$$

THE DECOMPOSITION-BASED OUTER APPROXIMATION (DECOA) ALGORITHM

Algorithm 1 describes DECOA in a simplified form. Given an initial upper bound $\overline{v}$ on the optimal value of (1), and an initial MIPOA problem (2), defined by $\check{P}$ and $\check{G}$, it solves the MINLP (1) by successively refining the MIPOA using *decomposition-based* cut and breakpoint generation, see methods addCutsPoints and addBlockRefineCutsPoints.

The method initCutsPoints computes an initial polyhedral relaxation $\check{G}$, a first solution candidate x^*, an upper bound $\overline{v}$ of the optimal value of the MINLP, and a solution $\hat{x}$ of the MIPOA. The method addCutsPoints computes for $k \in K$ a solution $\hat{y}_k$ of the k-th projection problem

$$\min \|y - \hat{x}_k\|^2 \quad \text{s.t.} \quad y \in G_k^{NLP} \cap \check{G}_k^{LP},$$

and $\check{G}_k$ is refined by cuts and breakpoints at $\hat{y}_k$. For $k \in K$, the method addBlockRefineCutsPoints first computes a solution $\hat{x}_k$ of the k-th partly fixed MIPOA regarding a solution candidate x^*

$$\min c^T x \quad \text{s.t.} \quad x \in P \cap \check{G},\; x_m = x_m^*,\; m \in K \setminus \{k\},$$

then calls addCutsPoints for generating cuts and breakpoints regarding $\hat{x}_k$, and repeats both steps until there is no improvement. The method addNlpCuts fixes all integer variables of the MINLP (1) using the solution $\hat{x}$ of the MIPOA. Then a solution x^* of the resulting NLP is computed, and $\check{G}_k$ is refined by cuts and points at x_k^*. Furthermore, the method tightenBounds adds the cut $c^T x \leq \overline{v} = c^T x^*$ to $\check{P}$, and reduces the intervals $[\underline{x}_{ki}, \overline{x}_{ki}]$ by *optiminality based bound tightening*, similar as in [7].

Algorithm 1 Decomposition-based MIP outer approximation method DECOA

function oaSolve($\overline{v}, \check{P}, \check{G}$)
  $(x^*, \hat{x}, \overline{v}, \check{G}) \leftarrow$ initCutsPoints($\overline{v}, \check{P}, \check{G}$) `# init the MIPOA`
  repeat
    if $|\hat{x} - x^*|$ is not reducing **then**
      $(x^*, \check{G}) \leftarrow$ addFixedNlpCutsPoints($\tilde{x}, \check{G}$) `# compute solution`
      if $c^T x^* < \overline{v}$ **then**
        $\overline{v} \leftarrow c^T x^*$
        $(\check{P}, \check{G}) \leftarrow$tightenBounds($x^*, \check{P}, \check{G}$) `# reduce [x, x]`
        $(\hat{x}, \check{G}) \leftarrow$addBlockRefineCutsPoints($x^*, \check{G}$) `# gen. cuts and points by block-refine`
    repeat
      $(\hat{x}, \check{G}) \leftarrow$addCutsPoints($\hat{x}, x^*, \check{G}$) `# gen. cuts and points by projection`
    until stopping criterion
  until $\overline{v} - c^T \hat{x} < \epsilon$
  return $(\overline{v}, \hat{x}, x^*, \check{P}, \check{G})$

THE DECOMPOSITION-BASED INNER AND OUTER REFINEMENT (DIOR) ALGORITHM

Algorithm 2 describes DIOR, a decomposition algorithm for solving a general MINLP. It uses the methods autoDecomp for transforming the given MINLP into the form (1), calcCurvature for calculating the convexification parameters σ as in (4), initOa for initializing $\check{G}$, computing the first solution candidate x^* and the related upper bound $\overline{v}$, and a solution $\hat{x}$ of the MIPOA, colGenStart for computing initial inner approximation points S and refining $\check{G}$, addOptCut for cutting off a solution candidate x^*, and addColGenCuts for reducing the duality gap by adding CG-based cuts. CG and cut generation subproblems are solved by a simplified version of the DECOA Algorithm 1.

Algorithm 2 Decomposition-based Inner- and Outer-Refinement

function dior
  autoDecomp
  $(\overline{v}, \hat{x}, x^*, \check{G}) \leftarrow$initOa `# init. MIPOA and first solution using DECOA`
  $(S, \check{G}) \leftarrow$ colGenStart($\check{G}$) `# init. inner and outer approximation`
  $\check{P} \leftarrow P$
  repeat
    $(\check{P}, \check{G}) \leftarrow$optCut $(x^*, S, \check{P}, \check{G})$ `# cut off solution`
    $(S, \check{G}) \leftarrow$addColGenCuts $(\hat{x}, S, \check{P}, \check{G})$ `# reduce duality gap`
    $(\overline{v}, \hat{x}, x^*, \check{P}, \check{G}) \leftarrow$oaSolve $(\overline{v}, \check{P}, \check{G})$ `# compute new solution using DECOA`
  until $\overline{v} - c^T \hat{x} < \epsilon$ or stopping criterion
  return x^*

NUMERICAL EXPERIMENTS USING DECOGO

A preliminary version of Algorithms 1 and 2 were implemented in Python/Pyomo [6] as part of a new MINLP-solver, called Decogo (Decomposition-based Global Optimizer). The MINLP and MIP subproblems are solved using SCIP

[1, 12], and the LP and NLP subproblems are solved using Ipopt [13]. The global MINLP solver is only used during preprocessing for computing the convexification paramters σ_{kj} by solving (4). After preprocessing, only MIP, LP and NLP subproblems are solved.

Since the implementation of Decogo is not finished, we present only results for convex MINLPs using DECOA. The numerical experiments were performed using 91 instances of the MINLPlib2 [11] with a maximum and average number of 1400 and 230 variables, respectively. The experiments show that the current version of DECOA needs on average 15 MIPOA solutions to solve these MINLP instances. Switching off the decomposition-based cut and point generation methods addCutsPoints and addBlockRefineCutsPoints significantly increases the number of MIPOA solutions.

ACKNOWLEDGMENTS

This work was funded by by grant 03ET4053B of the German Federal Ministry for Economic Affairs and Energy.

REFERENCES

[1] T. Achterberg. SCIP: solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, Jul 2009.

[2] R. Borndörfer, A. Löbel, M. Reuther, T. Schlechte, and S. Weider. Rapid branching. *Public Transport*, 5:3–23, 2013.

[3] S. Burer and A. Letchford. Non-convex mixed-integer nonlinear programming: A survey. *Surveys in Operations Research and Management Science*, 17 (2):97–106, 2012.

[4] M. R. Bussieck and S. Vigerske. *MINLP Solver Software*. www.math.hu-berlin.de/s̃tefan/minlpsoft.pdf, 2014.

[5] J. Desrosiers and M. Lübbecke. Branch-price-and-cut algorithms. In J. Cochran, L. Cox, P. Keskinocak, J. Kharoufeh, and J. Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010.

[6] W. Hart, C. Laird, J. Watson, and D. Woodruff. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, 2012.

[7] H. Nagarajan, M. Lu, S. Wang, R. Bent, and K. Sundar. *An Adaptive, Multivariate Partitioning Algorithm for Global Optimization of Nonconvex Programs*. https://arxiv.org/abs/1707.02514, 2017.

[8] I. Nowak. *Parallel Decomposition Methods for Nonconvex Optimization - Recent Advances and New Directions*, 2014. Proceedings of MAGO.

[9] I. Nowak, N. Breitfeld, E. M. T. Hendrix, and G. Njacheun-Njanzoua. Decomposition-based inner- and outer-refinement algorithms for global optimization. *Journal of Global Optimization*, 2018.

[10] T. Ralphs and M. Galati. Decomposition and dynamic cut generation in integer linear programming. *Mathematical Programming*, 106(2):261–285, 2006.

[11] S. Vigerske. *MINLP Library 2*. http://www.gamsworld.org/minlp/minlplib2/html, 2017.

[12] S. Vigerske and A. Gleixner. SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optimization Methods and Software*, 33(3):563–593, 2018.

[13] A. Wächter. *An Interior Point Algorithm for Large-Scale Nonlinear Optimization with Applications in Process Engineering*. PhD thesis, Carnegie Mellon University, Pittsburgh, USA, http://researcher.watson.ibm.com/researcher/files/us-andreasw/thesis.pdf, 2002.