Optimization and Engineering (2021) 22:1389–1418
https://doi.org/10.1007/s11081-020-09576-x

RESEARCH ARTICLE

On decomposition and multiobjective-based column and disjunctive cut generation for MINLP

Pavlo Muts[1] · **Ivo Nowak**[1] · **Eligius M. T. Hendrix**[2]

Received: 6 February 2020 / Revised: 23 September 2020 / Accepted: 27 October 2020 /
Published online: 11 November 2020

Abstract
Most industrial optimization problems are sparse and can be formulated as block-separable mixed-integer nonlinear programming (MINLP) problems, defined by linking low-dimensional sub-problems by (linear) coupling constraints. This paper investigates the potential of using decomposition and a novel multiobjective-based column and cut generation approach for solving nonconvex block-separable MINLPs, based on the so-called resource-constrained reformulation. Based on this approach, two decomposition-based inner- and outer-refinement algorithms are presented and preliminary numerical results with nonconvex MINLP instances are reported.

Keywords Decomposition method · Parallel computing · Column generation · Nonconvex optimization · Global optimization · Mixed-integer nonlinear programming

1 Introduction

Most real-world Mixed Integer Nonlinear Programming (MINLP) models are sparse, e.g. instances of the MINLPLib (Vigerske 2018). These models can be reformulated as block-separable problems, defined by low-dimensional sub-problems, which are coupled by a moderate number of linear global constraints. In this paper

✉ Ivo Nowak
ivo.nowak@haw-hamburg.de

Pavlo Muts
pavlo.muts@haw-hamburg.de

Eligius M. T. Hendrix
eligius@uma.es

[1] Hamburg University of Applied Sciences, Hamburg, Germany

[2] University of Malaga, Málaga, Spain

we investigate the potential of solving block-separable MINLPs using a decomposition multi-tree approach.

1.1 Global optimization and decomposition methods

Most global optimization methods can be divided into three approaches:

1. *Sampling* (point-based; Shapiro 2003), stochastic like Evolutionary Algorithms (Bäck 1996) or deterministic like DIRECT (Kolda et al. 2003);
2. *Branch-and-Bound* and variants (tree-based; Tawarmalani and Sahinidis 2005);
3. *Successive Approximation* (model-based), e.g. LP/MIP approximation (Kronqvist et al. 2016; Muts et al. 2020).

Our focus is on *Decomposition-based Successive Approximation*, i.e. an approximation is improved by solving low-dimensional sub-problems. Decomposition is a very general approach that can be applied to convex optimization, as well as non-convex optimization and discrete optimization. These methods are based on dividing a model into smaller sub-problems defined by local constraints, which can be solved in parallel. Then the results are used for updating a global master problem defined by global constraints, which is typically an LP, a MIP or a QP problem. The sub-problems and the global master problem are alternatingly solved until some termination criterion is met. In the following, an overview on (deterministic) decomposition methods is given.

1.1.1 Lagrangian decomposition

Many MINLP decomposition methods are based on solving a Lagrangian relaxation. It has been proven that Lagrangian relaxation regarding the global constraints is equivalent to a convex relaxation regarding the local constraints of the original problem when we are dealing with a linear objective function (Nowak 2005). Lagrangian Decomposition (LD) (Feltenmark and Kiwiel 2000; Geoffrion 1974; Lemaréchal and Renaud 2001) solves the dual problem by exactly solving subproblems. Column Generation (CG) methods also solve the dual (Lübbecke and Desrosiers 2005). However, in contrast to LD, CG does not require solving subproblems exactly. It is sufficient to compute feasible points of sub-problems with negative reduced cost. Note that it is possible to solve the LP master problem inexactly, e.g. using subgradient or bundle methods. Other decomposition algorithms for solving a Lagrangian relaxation are cutting plane methods or Frank–Wolfe decomposition, see Nowak (2005) for an overview.

1.1.2 Alternating direction methods

The Alternating Direction Method (ADM) computes local solutions of a MINLP by alternately solving a QP master problem and MINLP sub-problems. Originally, ADMs were developed for finite element problems (Gabay and Mercier 1976) and

are based on Uzawas algorithm (Uzawa 1958). A review of ADMs including a convergence proof for convex problems with separable objective function is given in Boyd et al. (2011). An ADM for solving MIPs is presented in Geissler et al. (2014) and for MINLPs in Nowak (2015).

1.1.3 Multi-tree decomposition methods

This type of decomposition strategy uses a MIP master problem. It is called multi-tree, because an individual branch-and-bound tree is built for each MIP instance. Using one global master problem which is updated during the solution process, i.e. new constraints are added during the solution process in order to improve the master problem, is called single-tree strategy. Both approaches solve easier sub-problems, in order to update the global master problem. More discussion on single-tree and multi-tree approaches can be found in Lundell et al. (2018). Rapid Branching is a CG-based multi-tree successive fixing heuristic, described in Borndörfer et al. (2013) and Nowak (2014). For MINLP problems with a small duality gap, like many transport planning problems, this method can be used to compute near globally optimal solutions of problems with millions of variables. The MOP-CG approach of Bodur et al. (2016) is a multi-tree decomposition algorithm for solving loosely coupled IPs by alternately solving CG lower and upper bounding master problems and Multi-Objective Programming (MOP) sub-problems. It is based on a resource-constrained reformulation of the MINLP. We investigate the potential of this approach combining them with Decomposition-based Inner- and Outer-Refinement (DIOR), see Nowak et al. (2018).

1.2 Investigating potential of decomposition multi-tree approaches

To investigate the potential of decomposition in contrast to using one single search tree in MINLP, we develop Decomposition-based Inner-and Outer-Refinement (DIOR) algorithms. We also investigate the idea of dimension reduction following the concept of a resource constraint program as introduced by Bodur et al. (2016). To do so, we follow the following steps:

1. Reformulate the MINLP as a *resource-constrained program (RCP)* in a reduced space
2. Compute an *LP approximation* of the RCP regarding *nondominated* columns using a two-phase approach:
   (a) Subgradient method, (b) Column generation.
3. Compute a *MIP approximation* of the RCP by adding *disjunctive cuts*, using a *multi-objective-based line-search*

Section 2 first focuses on the problem definition and the resource-constraint approach. Section 3 describes a column generation algorithm for computing an initial inner and outer LP approximation. Section 4 presents a DIOR algorithm for computing an outer MIP approximation. The convergence of this algorithm is shown

in Sect. 4.6. A faster DIOR algorithm for computing an inner MIP approximation is presented in Sect. 5. The numerical evaluation in Sect. 6 shows the potential of the new decomposition-based approximation approach. We conclude in Sect. 7 outlining steps for future investigation.

2 Problem formulation and reformulations

We consider *block-separable* (or *quasi-separable*) MINLP problems of the form

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

with

$$P := \{x \in \mathbb{R}^n :\ a_i^T x \le b_i,\ i \in M_1,\ a_i^T x = b_i,\ i \in M_2\}, \tag{2}$$

$|M_1| + |M_2| = m$, and

$$X_k := \{y \in [\underline{x}_k, \bar{x}_k] \subset \mathbb{R}^{n_{k1}} \times \mathbb{Z}^{n_{k2}} :\ g_{kj}(y) \le 0,\ j \in J_k\}. \tag{3}$$

The vector of variables $x \in \mathbb{R}^n$ is partitioned into $|K|$ disjoint blocks such that $n = \sum_{k \in K} n_k$, where $n_k = n_{k1} + n_{k2}$ is the dimension of block k and $x_k \in \mathbb{R}^{n_k}$ denotes the variables of block k. The vectors $\underline{x}, \bar{x} \in \mathbb{R}^n$ denote lower and upper bounds on the variables.

The linear constraints defining feasible set P are called *global*. The constraints defining feasible set X_k are called *local*. Set X_k is defined by linear and nonlinear local constraint functions, $g_{kj} : \mathbb{R}^{n_k} \to \mathbb{R}$, which are assumed to be bounded and continuously differentiable within the set $[\underline{x}_k, \bar{x}_k]$. Linear global constraints P are defined by $a_i \in \mathbb{R}^n, b_i \in \mathbb{R}, j \in [m]$. The linear objective function is defined by $c^T x := \sum_{k \in K} c_k^T x_k, c_k \in \mathbb{R}^{n_k}$.

The blocks $k \in K$ can be computed based on connected components of the so-called 'sparsity graph' (Nowak 2005). The nodes and arcs of this graph correspond to variables and nonzero entries of the Hessian of constraint functions of the original MINLP, respectively. Interestingly, this procedure yields small blocks for most instances of the MINLPLib (Vigerske 2018). Note that it is possible to reformulate a MINLP with a given arbitrary maximum block-size n_k by adding new variables and copy-constraints (Nowak 2005; Tawarmalani and Sahinidis 2005; Vigerske 2012).

2.1 Resource-constrained reformulation

If the MINLP (1) has a huge number of variables, it can be difficult to solve it in reasonable time. In particular, if the MINLP is defined by discretization of some infinitely dimensional variables, like in stochastic programming or in differential equations, and in addition the sub-problems are reformulated in a lifted space using auxiliary variables, the number of variables can be significantly higher than the number of global constraints connecting sub-problems. For such problems, a

resource-constrained view can be promising. In this approach, the original problem (1) in n-dimensional space is viewed from the m global constraints. Define the matrix A_k by

$$A_{ki} = \begin{cases} c_k^T & : i = 0, \\ a_{ki}^T & : i \in [m]. \end{cases} \tag{4}$$

and consider the transformed feasible set:

$$W_k := \{A_k x_k \ : \ x_k \in X_k\} \subset \mathbb{R}^{m+1}. \tag{5}$$

The variables $w_k := A_k x_k$ are called *resources*. They describe how the objective value and the constraint values $a_i^T x$ are distributed among the blocks. Note that for sparse MINLPs the number of non-zero resources, for which $A_{ki} \neq 0$, can be much smaller than m, see Sect. 2.3. Let

$$H := \left\{ w \in \prod_{k \in K} \mathbb{R}^{m+1} \ : \ \sum_{k \in K} w_{ki} \leq b_i, \ i \in M_1 \sum_{k \in K} w_{ki} = b_i, \ i \in M_2 \right\} \tag{6}$$

denote the global constraints in the resource space. We then define the *resource-constrained formulation* of (1) as

$$\min \sum_{k \in K} w_{k0} \quad s.t. \ w \in H, w_k \in W_k, k \in K. \tag{7}$$

Proposition 1 *Problem* (1) *and* (7) *are equivalent to the following two-level program:*

$$\min \sum_{k \in K} w_{k0}^* \quad s.t. \ w \in H, \tag{8}$$

where w_{k0}^* *is the optimal value of the sub-problem* RCP_k *given by*

$$\begin{aligned} w_{k0}^* := \min \ & c_k^T x_k \\ s.t. \ & A_{ki} x_k \leq w_{ki}, \ i \in M_1, \\ & A_{ki} x_k = w_{ki}, \ i \in M_2, \ x_k \in X_k. \end{aligned} \tag{9}$$

Proof Problem (1) can be formulated as

$$\begin{aligned} \min \ & \sum_{k \in K} c_k^T x_k \\ s.t. \ & A_{ki} x_k \leq w_{ki}, \ i \in M_1, \\ & A_{ki} x_k = w_{ki}, \ i \in M_2, \ x_k \in X_k, \ k \in K, \\ & w \in H. \end{aligned} \tag{10}$$

This shows that (1) and (7) are equivalent. For a given solution (w^*, x^*) of (10), it follows that x^* fulfills (9). Hence, (8) is equivalent to (10). □

2.2 Multi-objective reformulation

The multi-objective (MO) view on (7) changes the focus from the complete image set W_k to the relevant set of Pareto optimal points. A similar reformulation is presented in Bodur et al. (2016). Consider the following MO sub-problem of block k, called MOP_k, where we aim to minimize $|M_1| + 1$ resources simultaneously

$$\min (A_{ki}x_k)_{i\in M_1\cup\{0\}} \quad s.t.\ x_k \in X_k. \tag{11}$$

The nondominated (Pareto-optimal or efficient) front of MO sub-problem (11) is defined to be the set of vectors, $w_k = A_k x_k$ with $x_k \in X_k$, with the property that there does not exist any other feasible solution, $v_k = A_k y_k$ with $y_k \in X_k$, which dominates w_k, i.e., for which $v_{ki} \le w_{ki}$ for all $i \in M_1 \cup \{0\}$, and $v_{ki} < w_{ki}$ for at least one index $i \in M_1 \cup \{0\}$. An element of the nondominated front is known as a nondominated point (NDP). In other words, a NDP is a feasible objective vector for which none of its components can be improved without making at least one of its other components worse. A feasible solution $x_k \in X_k$ is efficient (or Pareto optimal) if its image $w_k = A_k x_k$ is a NDP, i.e. it is nondominated. Define the set of NDPs of (11) by

$$W_k^* := \{w \in W_k : (w_i)_{i\in M_1\cup\{0\}} \text{ is a NDP of (11)}\}$$

and correspondingly, set

$$W^* := \prod_{k\in K} W_k^*$$

Proposition 2 *The solution of problem* (7) *is attained at* $w^* \in W^*$.

$$\min \sum_{k\in K} w_{k0} \quad s.t.\ w \in H, \quad w_k \in W_k^*,\ k \in K. \tag{12}$$

Proof Assume there exist parts $\hat{w}_k^* \notin W^*$ of a solution w^*, i.e the parts are dominated. This means $\exists \hat{w}_k \in W_k^*$ which dominates w_k^*, i.e. $\hat{w}_{ki} \le w_{ki}^*$ for $i \in \{0\} \cup M_1$. Consider $\hat{w}$ the corresponding solution where in w^* the parts w_k^* are replaced by $\hat{w}_k$. Then $\hat{w}$ is feasible for RCP given

$$\sum_{k\in K} \hat{w}_{ki} \le \sum_{k\in K} w_{ki}^* \le b_i,$$

for $i \in M_1$, and its objective function value is at least as good as that of w^* as

$$\sum_{k\in K} \hat{w}_{k0} \le \sum_{k\in K} w_{k0}^*.$$

which means that the optimum is attained at a NDP point $\hat{w} \in W^*$. □

2.3 Reducing the dimension of the resources

In many practical problems, some of the blocks may not appear in a global constraint. To make use of this characteristic in terms of dimension reduction, we should consider the exact properties. Consider the index set of *relevant resources*

$$M_{1k} := \{i \in \{0\} \cup M_1 : A_{ki} \neq 0\}, \quad M_{2k} := \{i \in M_2 : A_{ki} \neq 0\}. \tag{13}$$

Then RCP (7) is equivalent to

$$\begin{aligned} \min \ & \sum_{k \in K,\, 0 \in M_{1k}} w_{k0}, \\ s.t. \ & \sum_{k \in K,\, i \in M_{1k}} w_{ki} \leq b_i,\ i \in M_1, \\ & \sum_{k \in K,\, i \in M_{2k}} w_{ki} = b_i,\ i \in M_2, \\ & w_k \in \Pi_k(W_k), \quad k \in K, \end{aligned} \tag{14}$$

where the projection operator $\Pi_k : \mathbb{R}^{m+1} \to \mathbb{R}^{|M_{1k}|+|M_{2k}|}$ is defined by

$$\Pi_k(w) := (w_i)_{i \in M_{1k} \cup M_{2k}}. \tag{15}$$

Similarly, following Proposition 2, (12) is equivalent to

$$\begin{aligned} \min \ & \sum_{k \in K,\, 0 \in M_{1k}} w_{k0}, \\ s.t. \ & \sum_{k \in K,\, i \in M_{1k}} w_{ki} \leq b_i,\ i \in M_1, \\ & \sum_{k \in K,\, i \in M_{2k}} w_{ki} = b_i,\ i \in M_2, \\ & w_k \in \Pi_k(W_k^*),\ k \in K. \end{aligned} \tag{16}$$

Formulations (14) and (16) are of interest, because the number $\sum_{k \in K} |M_{1k}| + |M_{2k}|$ of relevant resources is usually significantly smaller than the number n of variables in the original problem. This is the case for sparse optimization models, for which the model components are coupled by a moderate number of global constraints.

2.4 Supported nondominated points

A common method to compute a NDP is the weighted sum method (Miettinen 1998), which solves an optimization problem with a single objective obtained as a positive (convex) combination of the objective functions of the multiobjective problem:

$$\min d^T A_{M_{1k}} x_k \quad s.t.\ x_k \in X_k. \tag{17}$$

For a positive weight vector $d \in \mathbb{R}_+^{|M_{1k}|}$, any optimal solution of (17) is an efficient solution of (11), i.e., its image is nondominated. Such a solution and its image are called a *supported efficient solution* and a *supported NDP*, respectively. Thus, an efficient solution x_k is supported if there exists a positive vector d for which x_k is an optimal solution of (17), otherwise x_k is called unsupported.

Example 1 We introduce a numerical example which can be calculated by hand to illustrate the introduced concepts. Let $n = 4$, $K = \{1, 2\}$, $c = (-1, -2, -1, -1)$, $A = (2, 1, 2, 1)$, $b = 10$, $\underline{x} = (0, 0, 2, 1)$ and $\overline{x} = (5, 1.5, 5, 3)$. Integer variables are $I_1 = \{1\}, I_2 = \{3\}$ and the local constraints $g_{11}(x_1, x_2) = 3x_2 - x_1^3 + 6x_1^2 - 8x_1 - 3$ and $g_{21}(x_3, x_4) = x_4 - \frac{5}{x_3} - 5$. One can verify that the optimal solution is $x = (1, 1.5, 2, 2.5)$ with objective value -8.5. In the resource space, this corresponds to the points $w_1 = (-4, 3.5)$ in space W_1 and $w_2 = (-4.5, 6.5)$ in space W_2.

Figure 1 sketches the image spaces W_1 and W_2 with the corresponding Pareto front. For block $k = 2$, the image nearly coincides with the Pareto front W_2^*, although the number of supported points is limited. For block $k = 1$, one can observe more clearly parts of the feasible space that are dominated.

3 Column generation

In this section we describe a column generation algorithm for computing initial inner and outer LP approximations. We use the notation $[S] := \{1, \ldots, |S|\}$ for the index set of a discrete set S, i.e. $S = \{y_j : j \in [S]\}$. For a subset $T \subseteq S$, $[T]$ is defined by $[T] := \{j \in [S] : y_j \in T\}$.

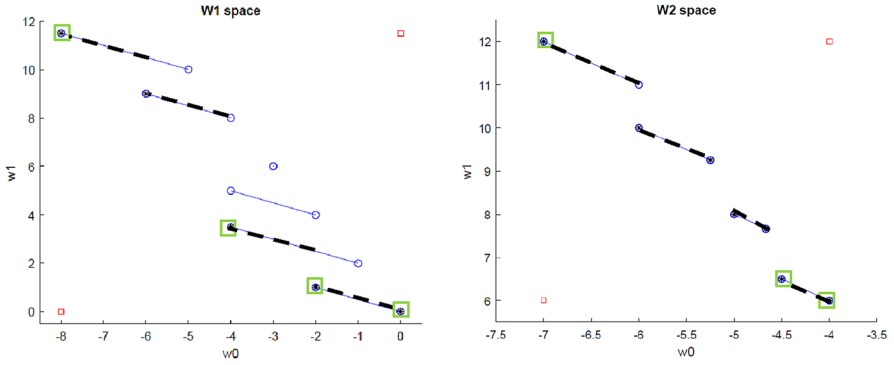

Fig. 1 Resource constraint space of blocks $k = 1, 2$. Image spaces W_1 and W_2 in blue with extreme points as circles. Pareto front in black, with extreme points as a star. Supported Pareto points are marked with a green square. As a red square, the ideal $\underline{w}_k$ (left under) and the Nadir point (right-up). (Color figure online)

3.1 Inner approximation

Column generation (CG) is a decomposition method for solving the following *convex relaxation* of (1)

$$\min c^T x \quad s.t.\ x \in P,\ x_k \in \text{conv}\,(X_k),\ k \in K. \tag{18}$$

Note that the convex relaxation (18) of problem (1) is equivalent to the *Lagrangian relaxation* of problem (1) regarding the global constraints, see Lemma 3.7 of Nowak (2005) for the proof.

The quality of convex relaxation (18) of MINLP (1) depends strongly on the *duality gap*, defined by

$$\text{gap} := \text{val}\,(1) - \text{val}\,(18).$$

Given a finite set of feasible points

$$S_k := \{y_{kj}\}_{j\in[S_k]} \subset X_k,$$

we have that

$$\min c^T x \quad s.t.\ x \in P,\ x_k \in \text{conv}\,(S_k),\ k \in K \tag{19}$$

is an inner LP approximation of (18). The resource constrained formulation of (19) is given by

$$\min \sum_{k\in K} w_{k0} \quad s.t.\ w \in H, \quad w_k \in \text{conv}\,(R_k),\ k \in K, \tag{20}$$

where $R_k := \{A_k y : y \in S_k\}$ is called a *set of columns* $r_{kj} \in \mathbb{R}^{m+1}$. An LP formulation of the inner approximation (20), called LP-IA, is given by

$$\min \sum_{k\in K} w_{k0}(z_k) \quad s.t.\ w(z) \in H, \quad z_k \in \Delta_{|R_k|},\ k \in K, \tag{21}$$

where

$$w(z) := \sum_{k\in K} w_k(z_k), \quad w_k(z_k) := \sum_{j\in[R_k]} z_{kj} r_{kj}, \quad r_{kj} \in R_k, \quad z_k \in \Delta_{|R_k|}. \tag{22}$$

$\Delta_{|R_k|}$ denotes the standard simplex

$$\Delta_{|R_k|} = \{z \in \mathbb{R}^{|R_k|} : \sum_{j\in[R_k]} z_j = 1, z_j \geq 0, j \in [R_k]\}.$$

CG generates columns $r_{kj} \in \mathbb{R}^{m+1}$ and adds them to the column set $R_k, k \in K$, by alternately computing a dual solution μ of (21), solving Lagrangian sub-problems

$$y_k(\mu) := \text{argmin}\{(1, \mu^T) A_k x : x \in X_k\}, \tag{23}$$

and adding $y_k(\mu)$ to R_k for $k \in K$. In general, sub-problem (23) is a nonconvex MINLP problem since it is defined over nonlinear feasible set X_k. However, it is smaller than the original problem (1) and its size depends on the block-size $n_k, k \in K$.

The computation of convex relaxation (18) is based on solving large easy LP master problems (19) and small difficult MINLP sub-problems (23). In order to be efficient, a fast sub-solver for Lagrangian sub-problem (23) is necessary. Examples of fast sub-solvers are (truncated) branch-and-cut, local search, dynamic programming-based constrained shortest path (Engineer et al. 2008) or MIP-based OA (Kronqvist et al. 2018; Nagarajan et al. 2019).

3.2 Initializing LP-IA

Algorithm 1 computes initial columns $R_k, k \in K$, by performing a *subgradient method* for maximizing the dual function of problem (1) regarding the global constraints:

$$L(\mu) := \sum_{k \in K} \min_{y_k \in X_k} (1, \mu^T) A_k y_k - \mu^T b. \tag{24}$$

We compute the step length α^p by comparing the values of the function $L(\mu)$ defined in (24) at different iterations p of Algorithm 1, similarly as in Shor (1985):

$$\alpha^{p+1} = \begin{cases} 0.5\alpha^p & : L(\mu^p) < L(\mu^{p-1}), \\ 2\alpha^p & : L(\mu^p) > L(\mu^{p-1}) > L(\mu^{p-2}), \\ \alpha^p & : \text{otherwise.} \end{cases} \tag{25}$$

The method SOLVESUBPROBLEM(d) solves MINLP sub-problem (23) for a given search direction $d \in \mathbb{R}^{m+1}$

$$\begin{aligned} y_k = \text{argmin} & d^T A_k x, \\ & s.t.\ x \in X_k, \end{aligned} \tag{26}$$

and computes reduced cost δ_k of the new point y_k. The columns $w_k = A_k y_k$ are added to the column set R_k, $k \in K$. The reduced cost δ_k is computed by taking the difference between the cost of new column w_k regarding direction d and minimum cost of existing columns R_k regarding direction d, i.e.

$$\delta_k = d^T w_k - \min_{r_k \in R_k} d^T r_k.$$

After computing minimizers y_k of (26) for $d = (1, \mu^T)$, one can easily compute the value of dual function (24). Note that if y_k is a global minimizer and d_k is a non-negative search direction, then $w_k = A_k y_k$ is a supported NDP.

Algorithm 1 Initialization of LP-IA

function INITIA
  $R \leftarrow \emptyset$, $p \leftarrow 0$, $\mu^p \leftarrow 0, \alpha^p = 1$
  for $k \in K$ **do**
    $(y_k, \delta_k) \leftarrow$ SOLVESUBPROBLEM$(1, \mathbf{0}^T)$, $R_k \leftarrow R_k \cup \{A_k y_k\}$
  repeat
    $p \leftarrow p + 1$, $\mu^p \leftarrow \mu^{p-1} + \alpha^p (Ay - b)$
    for $k \in K$ **do** $(y_k, \delta_k) \leftarrow$ SOLVESUBPROBLEM$(1, (\mu^p)^T)$, $R_k \leftarrow R_k \cup \{A_k y_k\}$
  until $p = p_{\max}$
  return R

3.3 A column generation algorithm

Algorithm 2 describes a Column Generation algorithm for computing LP-IA (21). In the beginning of the algorithm, the feasible set of LP-IA (21) may be empty. Therefore, the following master problem with slacks $s \in \mathbb{R}^m_+$ is solved by SOLVESLACKMASTERPROBLEM(R)

$$\begin{aligned} \min & \sum_{k \in K} w_{k0}(z_k) + \theta \sum_{i \in M_1} s_i + \theta \sum_{i \in M_2} s_{i1} + s_{i2} \\ s.t. & \sum_{k \in K} w_{ki}(z_k) \le b_i + s_i \,, \; i \in M_1, \\ & \sum_{k \in K} w_{k\ell}(z_k) = b_i + s_{\ell 1} - s_{\ell 2} \,, \; \ell \in M_2, \\ & z_k \in \Delta_{|R_k|}, \; k \in K, \; s_i, s_{\ell 1}, s_{\ell 2} \ge 0, \quad i \in M_1, \ell \in M_2, \end{aligned} \tag{27}$$

where penalty $\theta > 0$ is sufficiently large. If the slack variables are nonzero, i.e. $s \neq 0$, in order to eliminate nonzero slack variables, the method SLACKDIRECTIONS computes a new search direction $d \in \mathbb{R}^m$ for the subproblems (23) in the following way

$$d := \sum_{\substack{s_i > 0.1 \max(s), \\ i \in M_1}} e_i + \sum_{\substack{s_{i1} > 0.1 \max(s), \\ i \in M_2}} e_i - \sum_{\substack{s_{i2} > 0.1 \max(s), \\ i \in M_2}} e_i,$$

with $e_i \in \mathbb{R}^m$ the vector with a one at coordinate i and zeros elsewhere.

Algorithm 2 Column Generation

```
1: function colGen(R)
2:     R ← initIA
3:     M ← ∅
4:     repeat
5:         (z, μ, s) ← solveSlackMasterProblem(R)
6:         M ← M ∪ μ
7:         if s > 0 then
8:             d ← slackDirections(s)
9:             for k ∈ K do
10:                (y_k, δ_k) ← solveSubProblem(0, d^T),    R_k ← R_k ∪ {A_k y_k}
11:        for k ∈ K do
12:            (y_k, δ_k) ← solveSubProblem(1, μ^T),    R_k ← R_k ∪ {A_k y_k}
13:    until ∀δ_k ≥ 0
14:    return (z, R, M)
```

4 A DIOR algorithm for computing an outer MIP approximation

In this section, we present a DIOR algorithm for computing an exact outer MIP approximation of the resource-constrained problem (7). It consists of an LP phase (using an LP master problem) and a MIP phase (using a MIP master problem). The LP phase generates supported NDPs and is intended to speed up the convergence of the algorithm, like in Muts et al. (2020). The MIP phase generates also non-supported NDPs. For the sake of simplicity, we consider only global inequality constraints, i.e.

$$M_2 = \emptyset. \tag{28}$$

4.1 Outer LP approximation

An outer LP approximation of (7), called LP-OA, is given by

$$\min \sum_{k \in K} w_{k0} \quad s.t.\ w \in H, \quad w_k \in P_k,\ k \in K, \tag{29}$$

with

$$P_k := \{w_k \in \mathbb{R}^{m+1} : (1, \mu^T) A_k y_k(\mu) \leq (1, \mu^T) w_k,\ \forall \mu \in \mathcal{M}\}, \tag{30}$$

where $y_k(\mu)$ is the solution of the Lagrange sub-problem (23) regarding a dual point μ and $\mathcal{M}$ is a set of dual solution points, computed by Algorithm 2. Note that P_k consists of a set of supporting hyperplanes of set W_k. In other words, set P_k is defined by valid linear constraints, since they are constructed using optimal solution point $y_k(\mu)$ of the Lagrange sub-problem (23) regarding dual direction $\mu \in \mathcal{M}$. Therefore,

$$W_k \subset P_k, \quad k \in K. \tag{31}$$

4.2 Outer MIP approximation

We construct an outer approximation of W_k defined by polyhedral subdivision elements D_{ku}, called cells

$$D_k := \bigcup_{u \in U_k} D_{ku} \supset W_k, \tag{32}$$

where U_k is the index set of cell subdivision elements. We define a cell $D_{ku}, u \in U_k, k \in K$ in the following way

$$D_{ku} = \{w \in R^{m+1} : d_{kj}^T w \leq \beta_{kj}, j \in J_{ku}\}, \tag{33}$$

where J_{ku} denotes an index set of constraints defining the u-*th* cell. A nonconvex outer MIP approximation problem of (7) is given by

$$\begin{aligned} &\min \sum_{k \in K} w_{k0} \\ &s.t.\ w \in H, \quad w_k \in D_k, w_k \in P_k,\ k \in K, \end{aligned} \tag{34}$$

where binary variables select an index $u \in U_k$, such that $w_k \in D_{ku}$. Note that (34) does not consider the integer constraints of the original problem (1), since it is defined in transformed feasible set as in (5). Problem (34) contains only binary constraints which indicate whether cell $D_{ku}, u \in U_k$ is active or inactive. A MIP formulation of (34) is given by

$$\begin{aligned} &\min \sum_{k \in K} w_{k0} \\ &s.t.\ w \in H, \quad w_k \in P_k, \\ &\quad d_{kj}^T w_k \leq M(1 - t_u) + \beta_{kj}, j \in J_{ku}, u \in U_k, \\ &\quad t \in \{0,1\}^{|U_k|} \cap \Delta_{|U_k|}, \quad k \in K, \end{aligned} \tag{35}$$

where d_{kj} and $\beta_{kj}, j \in J_{ku}$, describe the polyhedral set corresponding to cell D_{ku} described by (33) and selected when $t_u = 1$. The "big" $M > 0$ should be sufficiently large. A restricted LP master problem regarding the selected cells $D_{ku_k}, k \in K$, is defined by

$$\min \sum_{k \in K} w_{k0} \quad s.t.\ w \in H, \quad w_k \in D_{ku_k},\ k \in K. \tag{36}$$

4.3 Disjunctive cuts

A *p-disjunctive cut* is defined by removing an open polyhedron C_k from a cell D_{ku} defined by p linear inequalities, i.e.

$$C_k = \{w \in \mathbb{R}^{m+1} : d_j^T w < \beta_j, j \in [p]\}.$$

We generate a *cone cut* with respect to vertex $v \in \mathbb{R}^{m+1}$, which we call an $|M_{1k}|$-disjunctive cut, by removing the cone of dominated area

$$C_k(v) := \{w \in \mathbb{R}^{m+1} : \exists i \in M_{1k}, w_i < v_i\}. \tag{37}$$

In the outer approximation, we use this concept to remove a cone of dominated area given an NDP v. This means that MIP approximation (35) is refined and cuts off parts of MIP solution $\hat{w}$ by removing cones $C_k(v_k)$ that contain $\hat{w}_k \in C_k(v_k)$ for those subproblems $k \in K$, where $\hat{w}_k \notin W_k$. In order to remove an cone $C_k(v_k)$, cell D_{ku} is divided into $|M_1|$ new cells

$$\{w \in D_{ku} : w_i \geq v_{ki}\}, \quad i \in M_{1k}. \tag{38}$$

4.4 Pareto line search

We describe a line search procedure for constructing a disjunctive cut for cutting off a solution $(u, \hat{w})$ of MIP-OA (35) by removing cone $C_k(v_k)$, defined in (37). Consider the ideal point

$$\underline{w}_{ki} = \min\{A_{ki}x_k : x_k \in X_k\}, \quad i \in M_{1k}. \tag{39}$$

We compute a disjunctive cut by removing the largest cone $C_k(v_k)$, such that $C_k(v_k) \cap W_k = \emptyset$, and v_k is on the line connecting ideal point $\underline{w}_k$ and solution point $\hat{w}_k$ of MIP-OA (34). Given the line

$$v_i(\lambda) := \underline{w}_{ki} + \lambda_k(\hat{w}_{ki} - \underline{w}_{ki}), \quad i \in M_{1k}, \quad \lambda \in \mathbb{R}, \tag{40}$$

the line search step size λ_k is computed by solving *Pareto line search sub-problem*

$$\begin{aligned}(y_k, \lambda_k) = \operatorname{argmin}\ & \lambda \\ s.t.\ & A_{ki}x \leq v_i(\lambda),\ i \in M_{1k}, \\ & x \in X_k,\ \lambda \in \mathbb{R}.\end{aligned} \tag{41}$$

Then the cone tip of $C_k(v_k)$ is $v_{ki} = v_i(\lambda_k)$, $i \in M_{1k}$.

4.5 DIOR using Pareto line search

Algorithm 4 describes an inner–outer refinement algorithm for solving (1) by iteratively adding disjunctive cone cuts based on Pareto line search. It maintains a set of sample points R_k and cells D_k on local level an a set $\mathcal{M}$ of dual vectors on global level. In Sect. 6, we will illustrate the algorithm numerically and also provide a sketch of the idea of the cone points, outer solutions and outer approximation in Fig. 2. It uses the methods:

- INITOA$(R, \mathcal{M})$ for initializing D by $D_{k1} = P_k, k \in K$, where P_k is defined by the local constraints of the OA-LP (29).
- SOLVEOUTERLP(D) for computing $(\hat{w}, \mu)$ of the OA-LP (29).
- IDEALPOINT for computing an *ideal point* $\underline{w}_k$ defined in (39).
- PARETOLINESEARCH$(\underline{w}_k, \hat{w}_k, \mu)$ for computing a possibly nonsupported NDP v_k by solving (41).
- CONESUBDIV(u_k, v_k, D_k) for removing cone $C_k(v_k)$ from set D_k by dividing D_{ku_k} into new overlapping cells 38.
- SOLVEOUTERMIP(D) for solving (34).

Algorithm 3 Intialize DIOR

```
1: function INITDIOR
2:     (ẑ, R, M) ← COLGEN                          # inner refine
3:     (u, D) ← INITOA(R, M)                       # init cells
4:     return (ẑ, u, D, R)
```

Algorithm 4 DIOR for computing an outer MIP approximation

```
1: function DIOR1
2:     (ẑ, u, D, R) ← INITDIOR,   ŵ ← w(ẑ)        # LP-IA refine
3:     for k ∈ K do
4:         w_k ← IDEALPOINT, add cuts w_k ≥ w_k to D_k
5:     repeat                                      # MIP-OA refine
6:         for k ∈ K do
7:             (v_k, λ_k) ← PARETOLINESEARCH(w_k, ŵ_k)
8:             if λ_k > 1 then D_k ← CONESUBDIV(u_k, v_k, D_k)   # D_k ← D_k \ C_k
9:         (u, ŵ) ← SOLVEOUTERMIP(D)               # MIP-OA solution
10:    until stopping criterion
11:    return Σ_{k∈K} w_{k0}
```

The algorithm starts by generating columns and a first outer approximation via Algorithm 3. A reduced version of Algorithm 4 is illustrated with instances having one global constraint in Muts et al. (2020).

4.6 Proof of convergence

In this section, we prove that Algorithm 4 computes an ϵ-global optimum of problem (1) in finitely many iterations. Note that we assume (28), i.e. $m = |M_1|$. Let

$$f(w) := \sum_{k \in K} w_{k0}.$$

Denote by $\hat{w}^p, \lambda^p, v^p$ the solution of MIP-OA (34), the dual solution of restricted LP-OA (36), and the solution of the Pareto line-search sub-problem (41) in iteration p, respectively. Furthermore, denote by $D_k^p \supset W_k, k \in K$, the outer approximation, which is refined by Algorithm 4 in iteration p by adding cone cuts. In particular, we have

$$D_k^{p+1} = D_k^p \backslash C_k(v_k^p), \quad \text{if } \lambda_k^p > 1, \quad D_k^{p+1} = D_k^p, \quad \text{if } \lambda_k^p = 1. \tag{42}$$

This process creates a sequence of enclosure sets

$$\widehat{W}^p := \prod_{k \in K} D_k^p,$$

with the following property

$$\widehat{W}^0 \supset \cdots \supset \widehat{W}^{p-1} \supset \widehat{W}^p \supset W. \tag{43}$$

In fact, it is sufficient to have $\widehat{W}^p$ enclose W^* as Property (2) details. In order to prove the main convergence result, we present intermediate results in Lemmas 1–7. It is assumed that MIP-OA master problem (34) and line-search sub-problems (41) are solved to global optimality.

Lemma 1 *Let $\widehat{W}_k^{*^p}$ be the Pareto front of $\widehat{W}_k^p$, i.e.*

$$\widehat{W}_k^{*^p} := \{w \in \widehat{W}_k^p : w \text{ is a NDP of } \min v \text{ s.t. } v \in \widehat{W}_k^p\}.$$

MIP-OA (34) *is equivalent to*

$$\min f(w) \quad s.t.\ w \in H, w_k \in \widehat{W}_k^{*^p}. \tag{44}$$

Proof This can be proved exactly as in Proposition 2. □

For the sequel of the proof, we introduce the extended resource set as a complement of the dominated area

$$\overline{W}_k = \mathbb{R}^{m+1} \backslash \{w \in \mathbb{R}^{m+1}, \exists v \in W_k^*, \exists i \in M_{1k}, w_i < v_i\}.$$

The *extended Pareto frontier* is defined as

$$\overline{W}_k^* := \{w \in \overline{W}_k : w \text{ is a NDP of } \min v \text{ s.t. } v \in \overline{W}_k\}.$$

Notice that $\overline{W}_k^*$ not only includes the Pareto front W_k^*, but also covers the gaps in the Pareto front, which are also sketched in Fig. 1.

Lemma 2 *Problem*

$$\min f(w) \quad s.t.\ w \in H, w_k \in W_k, k \in K,$$

is equivalent to

$$\min f(w) \quad s.t.\ w \in H, \quad w_k \in \overline{W}_k^*,\ k \in K. \tag{45}$$

Proof This can be proven as in Proposition 2. Assume that $\hat{w}_k^* \notin \overline{W}^*$ for some $k \in K$ of a solution w^*. This means $\exists \hat{w}_k \in W_k^*$ which dominates w_k^*, i.e. $\hat{w}_{ki} \leq w_{ki}^*$ for $i \in \{0\} \cup M_1$. Consider $\hat{w}$ the corresponding solution where in w^* the parts w_k^* are replaced by $\hat{w}_k$. As in the proof of Proposition 2 it follows that the optimum is attained at a NDP point $\hat{w} \in W^* \subseteq \overline{W}^*$. □

Considering $\overline{W}_k$ is relevant when we have a look at the line search (41). Now focusing on this step, notice that if λ takes a value of 1, the outer approximation sub-solution $\hat{w}_k$ is feasible and in that respect optimal for this part of the master problem. So, if for all sub-problems we have a value of $\lambda = 1$ we are done and the algorithm converged.

Lemma 3 *If after $p < \infty$ iterations of Algorithm 4, $\lambda_k^p = 1$ for all $k \in K$, then $\hat{w}^p$ is an optimal solution of problem (7).*

Proof Since $\hat{w}^p$ is an optimal solution of MIP-OA master problem (34), it is in $H \cap \widehat{W}^p$. From property (43) it is clear that $\widehat{W}^p$ also includes W. Since $\hat{w}_k^p \in \widehat{W}_k^{*p}$ from Lemma 1, it follows $\text{int}\,[\underline{w}_k, \hat{w}_k^p] \cap \widehat{W}_k^p = \emptyset$, and hence $\lambda_k \geq 1$ for all $k \in K$. If $\lambda_k^p = 1$, $v_k \in [\underline{w}_k, \hat{w}_k^p]$. Therefore, no cone $C_k(v_k)$ with $v_{ki} > \hat{w}_{ki}^p$ for $i \in \{0, \ldots, m\}$ and $\hat{w}_{ki}^p > \underline{w}_{ki}$ has to be removed from $\widehat{W}_k^p$. Hence, $\hat{w}_k^p \in \overline{W}_k^*$ for all $k \in K$. From Lemma 2 it follows that $\hat{w}^p$ minimizes the objective function within $H \cap W$. Since $\hat{w}^p \in H$, it follows that it is also an optimal solution of (7). □

Lemma 4 *If $\lambda_k^p \neq 1$ for some $k \in K$, Algorithm 4 excludes $\hat{w}^p$ from set $\widehat{W}^{p+1}$, i.e. $\hat{w}^p \notin \widehat{W}^{p+1}$.*

Proof If $\lambda_k^p \neq 1$, then $\lambda_k^p > 1$ from Lemma 1 (as above) and $C_k(v_k^p)$ is removed from $\widehat{W}^p$. If $\lambda_k^p > 1$, then $\exists i \in \{0, \ldots, m\}$ with $v_{ki}^p > \hat{w}_{ki}^p$. Hence, $\hat{w}_k^p \in C_k(v_k^p)$ and $\hat{w}^p \notin \widehat{W}^{p+1}$. □

In Lemma 5 we show that if Algorithm 4 does not stop in a finite number of iterations, the sequence of primal solution points contains at least one convergent subsequence $\{\hat{w}^{p_j}\}_{j=1}^{\infty}$, where

$$\{p_1, p_2, \ldots\} \subseteq \{1, 2, \ldots\} \quad \text{and} \quad \{\hat{w}^{p_j}\}_{j=1}^{\infty} \subseteq \{\hat{w}^p\}_{p=1}^{\infty}.$$

Since subsequence $\{\hat{w}^{p_j}\}_{j=1}^{\infty}$ is convergent, there exists a limit it has a limit $\lim_{j\to\infty} \hat{w}^{p_j} = w^*$. In Lemmas 6 and 7, we show that w^* is in the extended Pareto frontier $\overline{W}^*$ and therefore an optimal solution of (7), where $\overline{W}^* := \prod_{k\in K} \overline{W}_k^*$.

Lemma 5 *If Algorithm 4 does not stop in a finite number of iterations, it generates a convergent subsequence* $\{\hat{w}^{p_j}\}_{j=1}^{\infty}$.

Proof Since the algorithm has not terminated, for all $p = 1, 2, \ldots$ there exists a $k \in K$ such that $\lambda_k^p > 1$. Therefore, all the points in the sequence $\{\hat{w}^p\}_{p=1}^{\infty}$ will be distinct as shown in Lemma 4. Since $\{\hat{w}^p\}_{p=1}^{\infty}$ contains an infinite number of different points, and all are in a compact set, since MIP-OA is bounded, according to the Bolzano-Weierstrass Theorem, the sequence contains a convergent subsequence. □

Lemma 6 *The limit* w_k^* *for any convergent subsequence* $\{\hat{w}^{p_j}\}_{j=1}^{\infty}$ *generated in Algorithm 4 belongs to* $\overline{W}_k^*$.

Proof Let $\hat{w}_k^{p_j}$ and $\hat{w}_k^{p_{j+1}}$ be points from sequence $\{\hat{w}_k^{p_j}\}_{j=1}^{\infty}$. From Lemma 4 follows that in each iteration a cone $C_k(v_k^p)$ with $v_k^p = \underline{w}_k + \lambda_k^p(\hat{w}_k^p - \underline{w}_k)$ for some $k \in K$ is removed. This is also the case for iterate p_j, such that for future iterates $\exists i,\ \hat{w}_{ki}^p \geq v_{ki}^{p_j}$. This means

$$\exists i \in \{0, \ldots, m\},\ \hat{w}_{ki}^{p_{j+1}} - \underline{w}_i \geq \lambda_k^{p_j}(\hat{w}_{ki}^{p_j} - \underline{w}_i).$$

Assume that $\lambda_k^{p_j}$ does not converge to 1 and there is a value $\tau > 1$, such that in each iterate $\lambda_k^{p_j} > \tau > 1$. This leads to a contradiction, because the iterates v_k^p are in the bounded set D_k^0. From the proof of Lemma 3 we have $\lambda_k^{p_j} \geq 1$. Hence, $\lambda_k^{p_j} \to 1$ and $|\hat{w}_k^{p_j} - v_k^{p_j}| \to 0$. This implies $\lim_{j\to\infty} \hat{w}_k^{p_j} \in \overline{W}_k^*$. □

Lemma 7 *The limit point of a convergent subsequence is a global minimum point of* (7).

Proof Because each set $\widehat{W}^p$ is an outer approximation of the feasible set W, $f(\hat{w}^{p_j})$ gives a lower bound on the optimal value of the objective function. Due to property (43), sequence $\{f(\hat{w}^{p_j})\}_{j=1}^{\infty}$ is nondecreasing and since the objective function is continuous, we get $\lim_{j\to\infty} f(\hat{w}^{p_j}) = f(w^*)$. According to Lemma 6, limit point w_k^* is within the set $\overline{W}^*$. From Lemma 2 follows that w^* minimizes the objective function within $H \cap W$. Because $w^* \in H$, it is also an optimal solution of (7). □

Since Lemmas 6 and 7 apply to all convergent subsequences generated by solving the MIP-OA master problems (34), any limit point of such sequence will be a global optimum. We summarize the convergence results in the following theorem.

Theorem 1 *Algorithm 4 either finds a global optimum of* (7) *in a finite number of iterations or generates a sequence* $\{\hat{w}^{p_j}\}_{j=1}^{\infty}$ *converging to a global optimum.*

Proof Suppose the algorithm stops in a finite number of iterations. Then the last solution of the MIP-OA master problem (34) satisfies all constraints and according to Lemma 3 it is a global optimum of (7). In case the algorithm does not stop in a

finite number of iterations, it generates a sequence converging to a global optimum of (7) according to Lemmas 5 and 7. □

5 A DIOR algorithm for computing an inner MIP approximation

Motivated by the Rapid Branching approach (Borndörfer et al. 2013), we present in this section a heuristic DIOR algorithm for computing an inner MIP approximation of the resource-constrained problem (7). The goal of the algorithm is to compute a good primal solution point using the resources obtained by the inner MIP approximation. The approach is based on iteratively cutting off low-dimensional faces of conv $(R \cap H)$ containing the solution $\hat{w}$ of a MIP master problem. Note that the objective value of an inner MIP approximation does not provide a valid lower bound for problem (1).

5.1 Inner MIP approximation

Consider a partition of the relevant part of resource space $[\underline{w}, \overline{w}]$ using polyhedral partition elements D_{ku}, called cells, i.e.

$$[\underline{w}, \overline{w}] = \bigcup_{u \in U_k} D_{ku}, \quad \text{int}\,(D_{ku}) \cap \text{int}\,(D_{kv}) = \emptyset,\ \forall u, v \in U_k.$$

Implicitly, this partition also partitions the set of columns R_k into subsets $R_{ku} := R_k \cap D_{ku}$. An inner MIP approximation with slacks, called MIP-IA, is

$$\begin{aligned} \min\ & \sum_{k \in K} w_{k0} + \theta \sum_{i \in M_1} s_i + \theta \sum_{i \in M_2} s_{i1} + s_{i2} \\ s.t.\ & \sum_{k \in K} w_{ki} \le b_i + s_i, \quad i \in M_1, \\ & \sum_{k \in K} w_{ki} = b_i + s_{i1} - s_{i2}, \quad i \in M_2, \\ & w_k \in \bigcup_{u \in U_k} \text{conv}\,(R_{ku}), \quad k \in K, \\ & s_i \ge 0, \quad i \in [m]. \end{aligned} \tag{46}$$

A MIP formulation of (46) is given by

$$
\begin{aligned}
\min & \sum_{k\in K} w_{k0}(z_k) + \theta \sum_{i\in M_1} s_i + \theta \sum_{i\in M_2} s_{i1} + s_{i2} \\
s.t. & \sum_{k\in K} w_{ki}(z_k) \le b_i + s_i, \quad i \in M_1, \\
& \sum_{k\in K} w_{ki}(z_k) = b_i + s_{i1} - s_{i2}, \quad i \in M_2, \\
& z_k \in \Delta_{|R_k|}, \quad t \in \Delta_{|U_k|} \cap \{0,1\}^{|U_k|}, \\
& \sum_{j\notin[R_{ku}]} z_{kj} \le 1 - t_u, \quad u \in U_k, \quad k \in K, \\
& s_i \ge 0, \quad i \in [m],
\end{aligned}
\tag{47}
$$

where $[R_{ku}] \subset [R_k]$ denotes the indices of the columns R_{ku}. Note that replacing $\mathrm{conv}\,(R_{ku})$ in (46) by $\mathrm{conv}\,(W_k \cap D_{ku_k})$ defines a lower bounding program of the MINLP (1). By performing column generation regarding cells D_{ku}, the optimum value of (46) is converging to the optimum value of this lower bounding program.

5.2 Computing inner disjunctive cuts

MIP-IA (47) is refined by adding an *inner disjunctive cut*, defined by subdividing a cell D_{ku_k} into sub-cells D_{kv}, $v \in V_k(u_k)$ such that

$$
D_{ku_k} = \bigcup_{v\in V_k(u_k)} D_{kv}, \quad \mathrm{int}\,(D_{kv}) \cap \mathrm{int}\,(D_{kw}) = \emptyset,\ \forall v, w \in V_k(u_k),
$$

and replacing $\mathrm{conv}\,(R_{ku_k})$ by $\bigcup_{v\in V_k(u_k)} \mathrm{conv}\,(R_{ku_k} \cap D_{kv})$. In order to increase the optimum value of (46), it is necessary to cut off $w_k(\hat{z}_k)$ for some $k \in K$, where $\hat{z}$ is the solution of MIP-IA (47). This is equivalent to

$$
w_k(\hat{z}_k) \notin \mathrm{conv}\,(R_{ku_k} \cap D_{kv}), \quad \forall v \in V_k(u_k). \tag{48}
$$

Denote by $\hat{R}_k \subseteq R_k$ a set of supporting columns with $w_k(\hat{z}_k) \in \mathrm{int}\,(\mathrm{conv}\,(\hat{R}_k))$. We define the sub-cell D_{kv} such that it eliminates one supporting column from the set $\hat{R}_k$, i.e. $\hat{R}_k \not\subset D_{kv}, \forall v \in V_k(u_k)$. For that we set the point $\hat{w}_k(\hat{z})$ to be a vertex of $D_{kv}, \forall v \in V_k(u_k)$. Since $w_k(\hat{z}_k) \in \mathrm{int}\,(\mathrm{conv}\,(\hat{R}_k))$ and $w_k(\hat{z}_k) \in vert(D_{kv}), \forall v \in V_k(u_k)$, it cannot be expressed as a convex combination of points in $\hat{R}_k \cap D_{kv}$, i.e. (48) holds.

Algorithm 5 Project-and-branch method for inner disjunctive cut generation and local optimization

function INNERDISJUNCTCUT$(u, \hat{z}, D, R)$
  $V \leftarrow \emptyset, \quad L \leftarrow \{(\hat{z}, u, \emptyset)\}$ `# sub-path list`
  repeat
    $(\hat{z}, u, \tilde{K}) \leftarrow \operatorname{argmin}\{\underline{\nu}(\hat{z}) : (\hat{z}, u, \tilde{K}) \in L\}, \quad L \leftarrow L \setminus \{(\hat{z}, u, \tilde{K})\}$
    for $k \in K \setminus \tilde{K}$ **do** `# 1. find block for subdiv.`
      $\hat{R}_k \leftarrow$ GETSUPPORTCOLUMNS$(u_k, \hat{z}_k, D_k, R_k)$
    $\check{K} \leftarrow \{k \in K \setminus \tilde{K} : |\hat{R}_k| \geq 2 \wedge V_k = \emptyset\}$
    if $\check{K} \neq \emptyset$ **then**
      repeat
        $k \leftarrow \operatorname{argmin}\{|\max_{j \in [\hat{R}_\ell]} \hat{z}_{\ell j} - 0.5| : \ell \in \check{K}\}$
        $(\tilde{y}_k, \tilde{s}_k) \leftarrow$ SOLVERESPROJECTSUBPROBLEM$(u_k, \hat{z}_k)$ `# part. sol.`
        $R_k \leftarrow R_k \cup \{A_k \tilde{y}_k\}, \quad \tilde{w}_k \leftarrow A_k \tilde{y}_k + \tilde{s}_k$
        if $\tilde{s} = 0$ **then** $\check{K} \leftarrow \check{K} \setminus \{k\}$
      until $\tilde{s}_k > 0$ or $\check{K} = \emptyset$
      if $\tilde{s}_k > 0$ **then**
        $\tilde{K} \leftarrow \tilde{K} \cup \{k\}$ `# 2. subdivision`
        $\eta_k \leftarrow$ INNERSUBDIVCUTS$(\hat{z}_k, \tilde{w}_k, \hat{R}_k)$
        $(V_k, D_k) \leftarrow$ SUBDIV$(\eta_k, \hat{z}_k, D_k)$
        for $v \in V_k$ **do** `# 3. CG for sub-paths`
          $u_k \leftarrow v$
          $d_k \leftarrow$ GETSEARCHDIRECTION$(\tilde{w}_k, r_{kj_v})$ `# restricted CG`
          $y_k \leftarrow$ SOLVELAGSUBPROBLEM$(d_k, D_{ku_k}), \quad R_k \leftarrow R_k \cup \{A_k y_k\}$
          $(\hat{z}, \mu) \leftarrow$ SOLVERESTRICTIA(u, D, R)
          for $\ell \in K$ **do**
            $y_\ell \leftarrow$ SOLVELAGSUBPROBLEM$((1, \mu^T), D_{\ell u_\ell})$
            $R_\ell \leftarrow R_\ell \cup \{A_\ell y_\ell\}$
          if $|\tilde{K}| < |K|$ **then**
            $\hat{z} \leftarrow$ SOLVERESTRICTIA(u, D, R)
            $L \leftarrow L \cup \{(\hat{z}, u, \tilde{K})\}$ `# add new sub-path`
  until $L = \emptyset$ or stopping criterion
  return (V, D, R)

Algorithm 5 describes a project-and-branch procedure for refining MIP-IA by adding disjunctive cuts. It iteratively subdivides a cell D_{ku_k} into sub-cells $D_{kv}, v \in V_k(u_k)$. Denote the set of subdivided blocks by $\tilde{K} \subset K$. For each *sub-path of sub-cells* $D_{ku_k}, k \in \tilde{K}$, a lower bound $\underline{\nu}(\hat{z}) := \sum_{k \in K} \hat{w}_{k0}(\hat{z}_k)$ is computed, where $\hat{z}$ is the solution of the restricted LP-IA (54) regarding sub-cells D_{ku_k}. In order to prevent, that a cell D_{ku_k} is subdivided several times, the index set $V_k = V_k(u_k)$ is stored, and D_{ku_k} is only subdivided if $V_k = \emptyset$. The path data $(\hat{z}, u, \tilde{K})$ is stored in a list L. In each iteration, the following steps are performed:

1. In the first step, a block $k \in K \setminus \tilde{K}$ is determined, which will be later subdivided. Since the relative distance of $w_k(\hat{z}_k)$ to a column $r_{kj} \in \hat{R}_k$ is related to $1 - \hat{z}_{kj}$, a block is selected, for which $|\max_{j \in [\hat{R}_k]} \hat{z}_{\ell j} - 0.5|$ is small, where $\hat{R}_k \subseteq R_k$ is a set of supporting columns (with positive $\hat{z}_{kj}$-value).
   - The set $\hat{R}_k$ is computed by GETSUPPORTCOLUMNS$(u_k, \hat{z}_k, D_k, R_k)$ by first computing the p largest positive values $\hat{z}_{k1} \geq \hat{z}_{k2} \geq \cdots \geq \hat{z}_{kp}$ and setting

$$\hat{R}_k = \{r_{kj} : j \in [p],\ \hat{z}_{kj} > 0\}. \tag{49}$$

Then redundant columns $r_{kj} \in \hat{R}_k$ are removed, which can be represented as a convex combination of other columns of $\hat{R}_k$, such that $\text{conv}(\hat{R}_k) = \text{conv}(\hat{R}_k \backslash \{r_{kj}\})$ and $|\hat{R}_k| \leq |M_{1k}| + |M_{2k}|$.

- In order to check if $w_k(\hat{z}_k)$ is infeasible, the following resource constrained projection sub-problem with slacks, similar as RCP_k (9), is solved using SOLVERESPROJECTSUBPROBLEM$(u_k, \hat{z}_k)$:

$$\begin{aligned}(\tilde{y}_k, \tilde{s}_k) = \text{argmin} & \sum_{i \in M_{1k}} s_{ki} + \sum_{i \in M_{2k}} s_{ki1} - s_{ki2}, \\ s.t.\ & A_{ki} x_k \leq w_{ki}(\hat{z}_k) + s_{ki},\ i \in M_{1k}, \\ & A_{ki} x_k = w_{ki}(\hat{z}_k) + s_{ki1} - s_{ki2},\ i \in M_{2k}, \\ & s_{ki} \geq 0,\ i \in M_{1k} \cup M_{2k}, \\ & x_k \in X_k,\ A_k x_k \in D_{ku_k}.\end{aligned} \tag{50}$$

If $\tilde{s}_k \neq 0$, then $w_k(\hat{z}_k) \notin W_k$. The point $\tilde{y}_k \in X_k$ is a partial solution estimate, which is used in Algorithm 6 for computing a solution candidate.

2. In the second step, D_{ku_k} is subdivided into the sub-cells

$$D_{ku_j} = \{w \in D_{ku_k} : \eta_{kji}^T (w - w_k(\hat{z}_k)) \geq 0,\ i \in [\hat{R}_k] \backslash \{j\}\}, \tag{51}$$

for $u_j \in V_k(u_k)$. Sub-cells D_{kv} are defined by $|\hat{R}_k| - 1$ cut directions $\eta_{kji} \in \mathbb{R}^{|\hat{R}_k|}$ separating columns $r_{kj} \in \hat{R}_k$, and fulfilling (48). They are computed using the methods:

- INNERSUBDIVCUTS$(\hat{z}_k, \tilde{w}_k, \hat{R}_k)$ for computing cut directions η_{kji} of sub-cells D_{kv_j} defined in (51), by solving for all $j \in [\hat{R}_k]$, $i \in [\hat{R}_k] \backslash \{j\}$ the following system of equations

$$\begin{aligned}&\eta_{kji} \in \text{span}\{(r - \tilde{w}_k)\}_{r \in \hat{R}_k}, \\ &\eta_{kji}^T (r - w_k(\hat{z}_k)) = 0, \quad r \in \hat{R}_k \cup \{\tilde{w}_k\} \backslash \{r_{kj}, r_{ki}\}.\end{aligned} \tag{52}$$

If $\eta_{kji}^T (r_{kj} - w_k(\hat{z}_k)) \geq 0$, η_{kji} is multiplied by -1. Despite the removal of redundant columns in method GETSUPPORTCOLUMNS, there is no guarantee that system (52) is solvable. In order to be able always to compute a cut η_{kji}, we constructed it by computing the basis of the null space of (52).

- SUBDIV$(\eta_k, \hat{z}_k, D_k)$ for dividing D_{ku_k} into new cells D_{kv}, $v \in V_k(u_k)$, defined in (51) and updating $U_k \leftarrow U_k \backslash \{u_k\} \cup V_k(u_k)$.

3. In the third step, lower bounds $\underline{v}(\hat{z})$ of sub-paths D_{ku_k}, $k \in \tilde{K}$, are computed by performing a limited CG using

- GETSEARCHDIRECTION$(\tilde{w}_k, r_{kj_v})$ for setting the search direction

$$d_k = \tilde{w}_k - r_{kj_v}, \quad \text{where} \quad r_{kj_v} \notin D_{kv}.$$

- SOLVELAGSUBPROBLEM(d_k, D_{ku_k}) for solving

$$y_k = \text{argmin}\{d_k^T A_k x_k \,:\, x_k \in X_k, \quad A_k x_k \in D_{ku_k}\}. \tag{53}$$

- SOLVERESTRICTIA(u, D, R) for computing a primal and dual solution of

$$\begin{aligned} &\min \sum_{k\in K} w_{k0}(z_k), \\ &s.t.\ w(z) \in H,\ z_k \in \Delta_{|R_k|}, \\ &\quad z_{kj} = 0,\ j \notin [R_k \cap D_{ku_k}],\ k \in K. \end{aligned} \tag{54}$$

5.3 DIOR using an inner MIP approximation

Algorithm 6 describes an inner MIP refinement algorithm for computing a solution candidate of (1) by iteratively subdividing the feasible set and generating new columns using INNERDISJUNCTCUT$(u, \hat{z}, D, R)$ for computing (V, D, R). If $V = \emptyset$, the algorithm stops, since no cells were subdivided and no columns were generated. The algorithm uses the methods:

- SOLVEINNERMIP(R, D) for solving (47).
- LOCALSOLVE(y) for computing a solution candidate x^* of (1) by performing a local search starting from the point y.

Algorithm 6 Heuristic DIOR for computing an inner MIP approximation

```
1: function DIOR2
2:     (ẑ, u, D, R) ← INITDIOR                                  # LP-IA refine
3:     repeat
4:         (V, D, R) ←INNERDISJUNCTCUT(u, ẑ, D, R)              # MIP-IA refine
5:         if V ≠ ∅ then
6:             (u, ẑ) ← SOLVEINNERMIP(R, D)                     # MIP-IA solution
7:     until V = ∅ or stopping criterion
8:     ν ← Σ_{k∈K} ŵ_{k0}(ẑ_k)                                  # estimated lower bound
9:     for k ∈ K do y_k ← SOLVERESPROJECTSUBPROBLEM(u_k, ẑ_k)    # partial sol.
10:    x* ← LOCALSOLVE(y)                                       # solution candidate
11:    return (ν, x*)
```

Since it is not guaranteed that method INNERDISJUNCTCUT$(u, \hat{z}, D, R)$ generates all possible columns, Algorithm 6 provides only estimated lower bound $\underline{\nu}$ of problem (1).

6 Numerical results

Algorithm 4 and 6 were implemented with Pyomo (Hart et al. 2017), an algebraic modelling language in Python, as part of the parallel MINLP-solver Decogo (Nowak et al. 2018). The implementation of Decogo is not finished, in particular parallel

solving of sub-problems has not been implemented yet. The solver utilizes SCIP 5.0.1 (Gleixner et al. 2017) for solving MINLP sub-problems, GUROBI 9.0.3 for for solving MIP/LP master-problems and IPOPT 3.12.13 (Wächter and Lorenz 2006) for performing a local search starting from the projected solution of the last MIP master problem. Note that it is possible to use other suitable solvers which can interface with Pyomo. All computational experiments were performed using a computer with Intel Core i7-7820HQ 2.9 GHz CPU and 16 GB RAM.

Since most MINLP models are not given in a block-separable form, block structure identification of the original problem and its automatic reformulation into a block-separable form have been implemented. The block structure identification is based on the idea of connected components of a Hessian adjacency graph. Consider a MINLP problem defined by n variables and by $|M|$ functions $h_m, m \in M$. Consider a Hessian adjacency graph $\mathcal{G} = (V, E)$ defined by the following vertex and edge sets

$$\begin{aligned} V &= \{1, \dots, n\}, \\ E &= \{(i,j) \in V \times V : \frac{\partial^2 h_m}{\partial x_i \partial x_j} \neq 0,\ m \in M\}. \end{aligned} \tag{55}$$

In order to subdivide the set of variables into $|K|$ blocks, we compute the connected components $V_k, k \in K$, of $\mathcal{G}$ with $\bigcup_{k \in K} V_k = V$. We obtain the list of variables $V_k \subset V,\ k \in K$, such that $n = \sum_{k \in K} n_k$, where $n_k = |V_k|$.

6.1 Experiment with DIOR1

In this section, we illustrate the results of Algorithm 4 with *Example 1* outlined in Sect. 2.4. The optimal value of the problem is -8.5 with the optimal resources $(-4, 3.5)$ in space W_1 and $(-4.5, 6.5)$ in space W_2. Note that we solve Pareto line search sub-problem (41) to global optimality, in order to guarantee that the cone point is an NDP. As a stopping criterion for the algorithm, we use a tolerance on improvement of MIP-OA objective value and set it to $\epsilon_{MIP} = 10^{-5}$.

For the line search, the algorithm computes the ideal point $\underline{w}_1 = (-8, 0)$ and $\underline{w}_2 = (-6.2, 5)$. Algorithm 3 computes an initial OA point $\hat{w}_1 = (-3.6, 3)$ and $\hat{w}_2 = (-5, 7)$. Figure 2 shows that the optimal value of the MIP-OA (34) converges to the global optimum of (1) in 20 iterations after 12.5 s. It is interesting to notice that in space W_2, for almost all OA solution points, the corresponding cone point v_2 is identical, i.e. the OA solution $\hat{w}_2$ belongs to the feasible set.

For the instances with more than one constraint, the convergence is much slower. In each iteration, the algorithm generates $m + 1$ disjunctive cuts for the selected cell. These cuts are weak, since the algorithm first selects cells which don't improve the OA objective value. The cells, which improve the OA objective value, are selected only when other resources cannot be improved anymore. Also, after several iterations, the MIP-OA master problem becomes more difficult to solve due to the huge amount of generated disjunctive cone cuts.

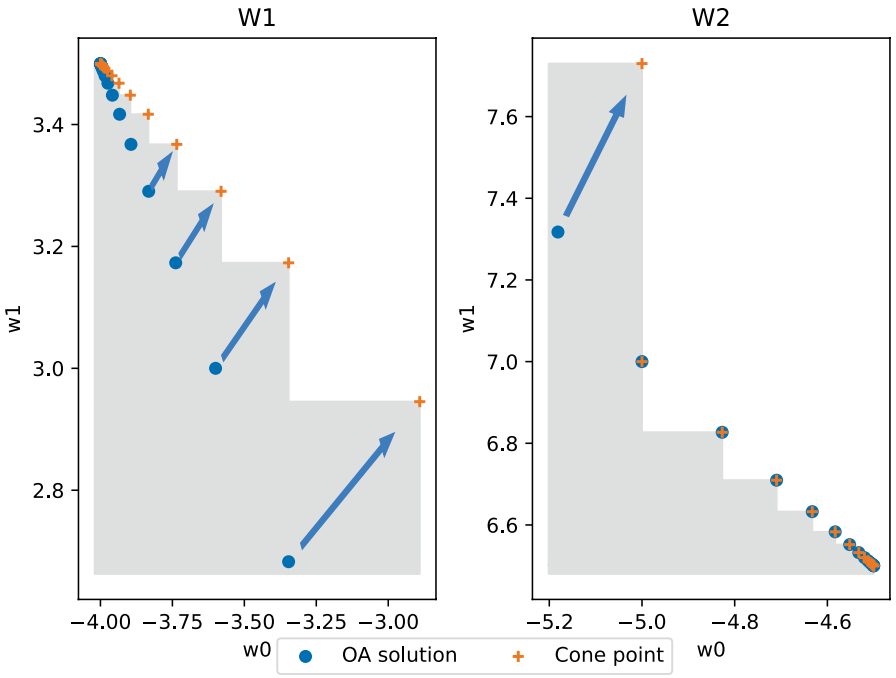

Fig. 2 Steps 5–10 of Algorithm 4 for *Example 1*. Blue arrows represent a line search towards the feasible set defined by OA solution $\hat{w}$ and ideal point $\underline{w}_1 = (-8, 0)$ and $\underline{w}_2 = (-6.2, 5)$. Grey shaded area represents eliminated cones. (Color figure online)

6.2 Experiments with DIOR2

In this section, we present the results for CG Algorithm 2 and DIOR2 Algorithm 6. For testing purpose, we selected several instances from MINLPLib (Vigerske 2018).

Table 1 Performance of CG Algorithm 2, CG

	Instances	n	$\|K\|$	m	N_{LP}	N_{sub}	$\|R\|$	$v^* - \underline{v}_{LP}$,%
1	alkyl	14	4	6	13	68	27	23.2
2	ex2_1_1	5	5	1	3	30	15	11.2
3	example 1	4	2	1	4	12	8	1.2
4	pooling_rt2tp	34	3	18	19	75	41	25.9
5	sep1	29	2	10	15	48	29	41.8
6	st_e05	5	3	2	4	39	9	78.3
7	st_glmp_kky	7	3	4	5	42	12	20.0
8	st_jcbpaf2	10	5	13	2	65	19	17.9
9	tln2	8	3	6	4	36	9	21.5
10	util	145	3	15	39	120	55	3.6

Table 2 Performance of Algorithm 6, DIOR2

	Instances	N_{sub}	$\|R\|$	N_{MIP}	$\underline{v}$	$\overline{v}$	v^*
1	alkyl	150	79	4	− 1.7	− 1.8	− 1.8
2	ex2_1_1	95	20	2	− 17.0	− 17.0	− 17.0
3	example 1	21	11	2	− 8.5	− 8.5	− 8.5
4	pooling_rt2tp	603	530	20	− 4647.6	− 3274.0	− 4391.8
5	sep1	366	334	20	− 499.3	− 510.1	− 510.1
6	st_e05	119	76	7	7049.3	7049.2	7049.2
7	st_glmp_kky	95	23	3	− 2.5	− 2.5	− 2.5
8	st_jcbpaf2	252	93	5	− 794.9	− 794.9	− 794.9
9	tln2	88	17	3	5.3	5.3	5.3
10	util	653	450	20	1115.3	1034.7	999.6

Table 3 Comparing Algorithm 6 with the SCIP solver. All values in seconds

	Instances	T_{dec}	T_{LP}	T_{MIP}	T_{MINLP}	T	T_{SCIP}
1	alkyl	1.2	0.1	1.3	22.1	26.7	2.02
2	ex2_1_1	0.5	0.01	0.4	6.8	8.7	0.45
3	example 1	0.1	0.01	0.4	0.9	1.7	0.01
4	pooling_rt2tp	2.1	0.1	12.5	163.1	203.0	1.84
5	sep1	0.9	0.1	11.9	80.0	104.4	1.81
6	st_e05	0.2	0.1	2.6	13.3	17.8	1.59
7	st_glmp_kky	0.7	0.1	0.9	8.5	11.4	1.8
8	st_jcbpaf2	0.7	0.1	1.7	22.2	28.3	0.89
9	tln2	0.5	0.01	0.8	4.7	7.1	0.02
10	util	18.7	0.1	12.0	88.3	143.0	2.28

More detailed statistics on the selected instances is given in Table 1.

Focusing on the setting for the Column Generation (Algorithm 2), notice that often smaller MINLP sub-problems can still be difficult to solve. Therefore, we set termination criteria for earlier stopping of SCIP for solving MINLP sub-problems, e.g. the maximum number of processed nodes after the last improvement of the primal bound or the relative gap tolerance. Those parameters are set to 500 and 0.01, respectively. Furthermore, Algorithm 2 solves the problem over its convex relaxation. In order to check convergence, we repeat one iteration of Algorithm 2 without early termination of sub-problem solving, i.e. we don't apply any stopping criteria for the sub-solver. The disadvantage of this strategy is that Column Generation needs more iterations to converge. Therefore, it solves more MINLP and LP sub-problems.

Table 1 shows the results of Column Generation for the selected test set. The characteristics of each instance are reported, i.e. problem size n, number of blocks $|K|$ and number of global constraints m. The performance measures are given by the number of solved LP master problems N_{LP}, the number of solved MINLP

sub-problems N_{sub} and number of generated columns $|R|$. Note that N_{LP} also denotes the number of iterations of Algorithm 2. We also compute the relative duality gap $v^* - \underline{v}_{LP}$, where $\underline{v}_{LP}$ denotes the objective of the LP master problem (20) and v^* denotes the best known objective function value.

Table 1 illustrates that the number of generated columns $|R|$ in Algorithm 2 is smaller than the number of solved MINLP sub-problems N_{sub}. This indicates that the MINLP sub-problem may generate the same column several times. For some instances, it is necessary to solve relatively more LP master problems, but these sub-problems are relatively easy, as can be observed in Table 3.

For the experiments with DIOR2, the maximum number of supporting columns in (49) is set to $p = 3$, and the maximum number of MIP iterations (number of times MIP-IA master problem (47) is solved) to 20, i.e. $N_{MIP} \leq 20$. Some MINLP sub-problems in steps 3–10 of Algorithm 6 are solved to optimality. For example, sub-problem (50), in order to check whether the resources of MIP-IA are feasible. Other MINLP sub-problems are not solved to optimality, i.e. they were solved as in Algorithm 2.

In Table 2 about DIOR2, N_{sub} and $|R|$ include the number of solved MINLP sub-problems and number of generated columns from Algorithm 2, respectively. The indicators are the number of MIP iterations N_{MIP} (number of times MIP-IA (47) is solved), optimal value $\underline{v}$ of MIP-IA (47) and objective value $\overline{v}$ at the primal solution point computed by the local solver.

Table 2 illustrates that the new decomposition-based successive approximation approach is able to solve nonconvex MINLP models to global optimality. Notice that like for CG, the number of solved MINLP sub-problems N_{sub} is higher than the number of generated columns $|R|$. For *Example 1*, DIOR2 reduces the number of iterations from 20 to 2 compared to DIOR1.

Table 3 compares Algorithm 6 to branch-and-bound-based solver SCIP 5.0.1 (Gleixner et al. 2017) in terms of computing time. Notice that computing time not only depends on performance, but also on the handiness of the programmer and the used platform; python is not directly fast. All settings of SCIP were set to default. For each instance, we compare total solution time T of DIOR2 with time spent by SCIP T_{SCIP}. Note that T also includes time spent for decomposition T_{dec}. T_{LP}, T_{MIP} and T_{MINLP} denote the time spent on solving LP master problems, MIP master problems and MINLP sub-problems, respectively.

Table 3 shows that SCIP requires less total time than Algorithm 6. However, Algorithm 6 spends most of the running time T on solving MINLP sub-problems. This shows some potential for solving these sub-problems in parallel. T_{MIP} depends on N_{MIP}, i.e. more iterations requires more time spent on solving MIP master problems. The MIP master problem becomes more difficult to solve when a lot of cuts are generated. Interesting enough, T_{LP} is relatively small and it does not depend on the number of solved LP master problems. Note that for *Example 1,* DIOR2 improves the solution compared to DIOR1 from 12.5 to 1.7 s.

7 Conclusions

MINLP is a strong paradigm for modelling practical optimization problems. We introduced multi-tree decomposition-based methods for solving MINLP models (1), which avoid building one big search tree to solve the problem. Instead, it solves smaller MINLP sub-problems, of which the solutions are combined in a master LP and MIP problem up to convergence. Moreover, we investigate the potential of the so-called resource constrained formulation of the MINLP problem.

Both algorithms DIOR1 and DIOR2 use low-dimensional MINLP-sub-problem solutions for refining resource-constrained LP and MIP master problems. DIOR1 is slow, but proven to converge. We have shown that DIOR2 is a faster heuristic procedure. On the other hand, an established well implemented branch-and-bound algorithm is faster than DIOR2. In the future, we will work on improving the DIOR algorithm, e.g. by implementing parallel solving of sub-problems and reducing the number sub-problems to be solved.

An advantage of multi-tree decomposition algorithms is the possibility to modify the optimization model during the solution process. An example for this is an airline transportation network which is extended during the solution process by adding new transport options, like train or bus connections. Another example is a response surface or an artificial neural network of a black-box function of a sub-problem, which is iteratively improved regarding new sample/training points. The generated cuts and points can also be used for performing a warm-start if the model has been changed slightly, e.g. in a dynamic optimization model with a moving horizon.

Similarly, like in CG, it is possible to start a multi-tree algorithm with a restricted or simplified search space, which is improved iteratively. Examples for simplifying the search space are: reducing scenario trees, restricting transport networks, or discretizing differential equations.

Such decomposition-based successive approximation methods may be used to solve large-scale optimization problems, like complex multidisciplinary models. Moreover, the presented multi-tree algorithms can be extended to multi-objective optimization by modifying the master problems according to multiple objectives.

Acknowledgements This work has been funded by Grant RTI2018-095993-B-I00 from the Spanish Ministry in part financed by the European Regional Development Fund (ERDF) and by Grant 03ET4053B of the German Federal Ministry for Economic Affairs and Energy.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Bäck T (1996) Evolutionary algorithms in theory and practice. Oxford University Press, Oxford

Bodur M, Ahmed S, Boland N, Nemhauser GL (2016) Decomposition of loosely coupled integer programs: a multiobjective perspective. http://www.optimization-online.org/DB_FILE/2016/08/5599.pdf

Borndörfer R, Löbel A, Reuther M, Schlechte T, Weider S (2013) Rapid branching. Public Transport 5:3–23

Boyd S, Parikh N, Chu E, Peleato B, Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Found Trends Mach Learn 3:1–122

Engineer F, Nemhauser G, Savelsbergh M (2008) Shortest path based column generation on large networks with many resource constraints. Technical report, Georgia Tech

Feltenmark S, Kiwiel KC (2000) Dual applications of proximal bundle methods including Lagrangian relaxation of nonconvex problems. SIAM J Optim 10(3):697–721

Gabay D, Mercier B (1976) A dual algorithm for the solution of nonlinear variational problems via finite element approximations. Comput Math Appl 2:17–40

Geissler B, Morsi A, Schewe L, Schmidt M (2014) Solving power-constrained gas transportation problems using an MIP-based alternating direction method. www.optimization-online.org/DB_HTML/2014/11/4660.html

Geoffrion AM (1974) Lagrangian relaxation for integer programming. Math Program Stud 2:82–114

Gleixner A, Eifler L, Gally T, Gamrath G, Gemander P, Gottwald RL, Hendel G, Hojny C, Koch T, Miltenberger M, Müller B, Pfetsch ME, Puchert C, Rehfeldt D, Schlösser F, Serrano F, Shinano Y, Viernickel JM, Vigerske S, Weninger D, Witt JT, Witzig J (2017) The SCIP Optimization Suite 5.0. Technical report, www.optimization-online.org/DB_HTML/2017/12/6385.html

Hart WE, Laird CD, Watson JP, Woodruff DL, Hackebeil GA, Nicholson BL, Siirola JD (2017) Pyomo-optimization modeling in Python, vol 67, 2nd edn. Springer, Berlin

Kolda TG, Lewis RM, Torczon V (2003) Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev 45(3):385–482

Kronqvist J, Bernal DE, Lundell A, Grossmann IE (2018) A review and comparison of solvers for convex MINLP. Optim Eng 20(2):397–455

Kronqvist J, Lundell A, Westerlund T (2016) The extended supporting hyperplane algorithm for convex mixed-integer nonlinear programming. J Global Optim 64(2):249–272

Lemaréchal C, Renaud A (2001) A geometric study of duality gaps, with applications. Math Program 90:399–427

Lübbecke M, Desrosiers J (2005) Selected topics in column generation. Oper Res 53(6):1007–1023

Lundell A, Kronqvist J, Westerlund T (2018) The supporting hyperplane optimization toolkit. www.optimization-online.org/DB_HTML/2018/06/6680.html

Miettinen K (1998) Nonlinear multiobjective optimization. Springer, Berlin

Muts P, Nowak I, Hendrix EMT (2020) A resource constraint approach for one global constraint MINLP. In: Gervasi O et al (eds) Computational science and its applications—ICCSA 2020. Berlin, Springer, pp 590–605. https://doi.org/10.1007/978-3-030-58808-3_43

Muts P, Nowak I, Hendrix EMT (2020) The decomposition-based outer approximation algorithm for convex mixed-integer nonlinear programming. J Global Optim 77(1):75–96

Nagarajan H, Lu M, Wang S, Bent R, Sundar K (2019) An adaptive, multivariate partitioning algorithm for global optimization of nonconvex programs. J Glob Optim 74(4):639–675

Nowak I (2005) Relaxation and decomposition methods for mixed integer nonlinear programming. Birkhäuser, Berlin

Nowak I (2014) Parallel decomposition methods for nonconvex optimization—recent advances and new directions. In: Proceedings of MAGO

Nowak I (2015) Column generation based alternating direction methods for solving MINLPs. www.optimization-online.org/DB_HTML/2015/12/5233.html

Nowak I, Breitfeld N, Hendrix EMT, Njacheun-Njanzoua G (2018) Decomposition-based inner- and outer-refinement algorithms for global optimization. J Global Optim 72(2):305–321

Shapiro A (2003) Monte Carlo sampling methods. In: Ruszczyński A, Shapiro A (eds) Handbooks in operations research and management science. Elsevier, Amsterdam, pp 353–425

Shor N (1985) Minimization methods for non-differentiable functions. Springer, Berlin

Tawarmalani M, Sahinidis N (2005) A polyhedral branch-and-cut approach to global optimization. Math Program 103(2):225–249

Uzawa H (1958) Iterative methods for concave programming. Stanford University Press, Stanford, pp 154–165

Vigerske S (2012) decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming. Ph.D. thesis, Humboldt-Universität zu Berlin

Vigerske S (2018) MINLPLib. http://minlplib.org/index.html

Wächter A, Lorenz BT (2006) On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Math Program 106(1):25–57