



<b>Titre:</b> Title:	On generalized surrogate duality in mixed-integer nonlinear programming
<b>Auteurs:</b> Authors:	Benjamin Müller, Gonzalo Muñoz, Maxime Gasse, Ambros Gleixner, Andrea Lodi et Felipe Serrano
<b>Date:</b>	2021
<b>Type:</b>	Article de revue / Journal article
<b>Référence:</b> Citation:	Müller, B., Muñoz, G., Gasse, M., Gleixner, A., Lodi, A. & Serrano, F. (2021). On generalized surrogate duality in mixed-integer nonlinear programming. <i>Mathematical Programming</i> , 2021, p. 1-30. doi: <a href="https://doi.org/10.1007/s10107-021-01691-6">10.1007/s10107-021-01691-6</a>



### Document en libre accès dans PolyPublie

Open Access document in PolyPublie

<b>URL de PolyPublie:</b> PolyPublie URL:	<a href="https://publications.polymtl.ca/9256/">https://publications.polymtl.ca/9256/</a>
<b>Version:</b>	Version officielle de l'éditeur / Published version Révisé par les pairs / Refereed
<b>Conditions d'utilisation:</b> Terms of Use:	CC BY



### Document publié chez l'éditeur officiel

Document issued by the official publisher

<b>Titre de la revue:</b> Journal Title:	Mathematical Programming (vol. 2021)
<b>Maison d'édition:</b> Publisher:	Springer Nature
<b>URL officiel:</b> Official URL:	<a href="https://doi.org/10.1007/s10107-021-01691-6">https://doi.org/10.1007/s10107-021-01691-6</a>
<b>Mention légale:</b> Legal notice:	

**Ce fichier a été téléchargé à partir de PolyPublie,  
le dépôt institutionnel de Polytechnique Montréal**

This file has been downloaded from PolyPublie, the  
institutional repository of Polytechnique Montréal

<http://publications.polymtl.ca>



# On generalized surrogate duality in mixed-integer nonlinear programming

Benjamin Müller<sup>1</sup> · Gonzalo Muñoz<sup>2</sup> · Maxime Gasse<sup>3</sup> · Ambros Gleixner<sup>4</sup> · Andrea Lodi<sup>3</sup> · Felipe Serrano<sup>1</sup>

Received: 29 June 2020 / Accepted: 28 June 2021  
© The Author(s) 2021

## Abstract

The most important ingredient for solving mixed-integer nonlinear programs (MINLPs) to global  $\epsilon$ -optimality with spatial branch and bound is a tight, computationally tractable relaxation. Due to both theoretical and practical considerations, relaxations of MINLPs are usually required to be convex. Nonetheless, current optimization solvers can often successfully handle a moderate presence of nonconvexities, which opens the door for the use of potentially tighter nonconvex relaxations. In this work, we exploit this fact and make use of a nonconvex relaxation obtained via aggregation of constraints: a *surrogate* relaxation. These relaxations were actively studied for linear integer programs in the 70s and 80s, but they have been scarcely considered since. We revisit these relaxations in an MINLP setting and show the computational benefits and challenges they can have. Additionally, we study a generalization of such relaxation that allows for multiple aggregations simultaneously and present the first algorithm

---

✉ Benjamin Müller  
benjamin.mueller@zib.de

Gonzalo Muñoz  
gonzalo.munoz@uoh.cl

Maxime Gasse  
maxime.gasse@polymtl.ca

Ambros Gleixner  
gleixner@zib.de

Andrea Lodi  
andrea.lodi@polymtl.ca

Felipe Serrano  
serrano@zib.de

- <sup>1</sup> Zuse Institute Berlin, Berlin, Germany
- <sup>2</sup> Universidad de O'Higgins, Rancagua, Chile
- <sup>3</sup> CERC, Polytechnique Montréal, Montreal, Canada
- <sup>4</sup> HTW Berlin and Zuse Institute Berlin, Berlin, Germany

that is capable of computing the best set of aggregations. We propose a multitude of computational enhancements for improving its practical performance and evaluate the algorithm's ability to generate strong dual bounds through extensive computational experiments.

**Keywords** Surrogate relaxation · MINLP · Nonconvex optimization

**Mathematics Subject Classification** 90-08 · 90C27 · 90C26

## 1 Introduction

We consider a mixed-integer nonlinear program (MINLP) of the form

$$\min_{x \in X} \left\{ c^\top x \mid g_i(x) \leq 0 \text{ for all } i \in \mathcal{M} \right\}, \quad (1)$$

where  $X := \{x \in \mathbb{R}^{n-p} \times \mathbb{Z}^p \mid Ax \leq b\}$  is a compact mixed-integer linear set, each  $g_i : \mathbb{R}^n \rightarrow \mathbb{R}$  is a factorable continuous function [41], and  $\mathcal{M} := \{1, \dots, m\}$  denotes the index set of nonlinear constraints. Such a problem is called *nonconvex* if at least one  $g_i$  is nonconvex, and *convex* otherwise. Many real-world applications are inherently nonlinear, and can be formulated as a MINLP. See, e.g., [28] for an overview.

The state-of-the-art algorithm for solving nonconvex MINLPs to global  $\epsilon$ -optimality is spatial branch and bound, see, e.g., [30,51,52], whose performance highly depends on the tightness of the relaxations used. Those relaxations are typically convex, and are iteratively refined by branching, cutting planes, and variable bound tightening, e.g., feasibility- and optimality-based bound tightening [8,50].

As a result of the rapid progress during the last decades, current solvers can often handle a moderate presence of nonconvex constraints efficiently. This progress opens the door for practical use of potentially tighter *nonconvex* relaxations in MINLP solvers. One example are MILP relaxations, see [11,43,68]. In this paper, we go one step further and explore a nonconvex relaxation referred to as *surrogate relaxation* [21].

**Definition 1** (*Surrogate relaxation*) For a given  $\lambda \in \mathbb{R}_+^m$ , we call the following optimization problem a *surrogate relaxation* of (1)

$$S(\lambda) := \min_{x \in X} \left\{ c^\top x \mid \sum_{i \in \mathcal{M}} \lambda_i g_i(x) \leq 0 \right\}. \quad (2)$$

Let  $\mathcal{F} \subseteq \mathbb{R}^n$  be the feasible region of the original problem (1), and let  $S_\lambda \subseteq \mathbb{R}^n$  be the feasible region of a surrogate relaxation given in (2). Throughout the whole paper we assume that  $\mathcal{F}$  is not empty. Clearly,  $\mathcal{F} \subseteq S_\lambda$  holds for every  $\lambda \in \mathbb{R}_+^m$ , and as such (2) provides a valid lower bound of (1). Moreover, solving (2) might be computationally more convenient than solving the original problem (1), since there is only one nonconvex constraint in  $S(\lambda)$ . Note that one may turn  $S_\lambda$  into a continuous

relaxation of (1) by removing the integrality restrictions. However, in this work we purposely choose to retain integrality in the relaxation and compare directly to the optimal value of (1) as opposed to its continuous relaxation.

The quality of the bound provided by  $S(\lambda)$  may be highly dependent on the value of  $\lambda$ , and therefore it is natural to consider the *surrogate dual* problem in order to obtain the tightest surrogate relaxation.

**Definition 2** (*Surrogate dual*). The following optimization problem is the *surrogate dual* of (1):

$$\sup_{\lambda \in \mathbb{R}_+^m} S(\lambda). \tag{3}$$

The function  $S$  is only lower semi-continuous [27], thus, there might be no  $\lambda$  such that  $S(\lambda)$  is equal to (3). The surrogate dual is closely related to the well-known *Lagrangian dual*  $\max_{\lambda \in \mathbb{R}_+^m} L(\lambda)$ , where

$$L(\lambda) := \min_{x \in X} \left\{ c^T x + \sum_{i \in \mathcal{M}} \lambda_i g_i(x) \right\}, \tag{4}$$

but always results in a bound that is at least as good [27,35], i.e.,

$$S(\lambda) \geq L(\lambda) \quad \text{for all } \lambda \in \mathbb{R}_+^m. \tag{5}$$

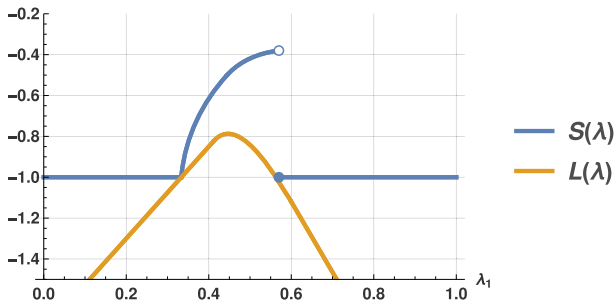
Figure 1 shows the difference between  $S(\lambda)$  and  $L(\lambda)$  on the two-dimensional instance of Example 1 below. In contrast to  $L : \mathbb{R}_+^m \rightarrow \mathbb{R}$ , which is a continuous and concave function,  $S : \mathbb{R}_+^m \rightarrow \mathbb{R}$  is only quasi-concave [27] and is, in general, discontinuous. As shown in Fig. 1, the main difficulty in optimizing  $S(\lambda)$  is that the function is most of the time “flat”, and leads to nontrivial dual bounds for only a small subset of the  $\lambda$ -space.

**Example 1** Consider the following nonconvex problem

$$\begin{aligned} \min \quad & -y \\ \text{s.t.} \quad & g_1(x, y) := 2xy + x^2 - y^2 - x \leq 0, \\ & g_2(x, y) := -xy - 0.3x^2 - 0.2y^2 - 0.5x + 1.5y \leq 0, \\ & (x, y) \in [0, 1]^2, \end{aligned}$$

which attains its optimal value  $-0.37$  at  $(x^*, y^*) \approx (0.52, 0.37)$ . The surrogate dual is

$$\sup_{(\lambda_1, \lambda_2) \in \mathbb{R}_+^2} \left\{ \begin{array}{l} \min \quad -y \\ \text{s.t.} \quad \lambda_1 g_1(x, y) + \lambda_2 g_2(x, y) \leq 0 \\ (x, y) \in [0, 1]^2 \end{array} \right\},$$



**Fig. 1** Plot of the surrogate and Lagrangian relaxations values for the MINLP detailed in Example 1. For display purposes, we plot both relaxations with respect to  $\lambda_1$  only. Note that since the surrogate relaxation is invariant to scaling, we can add a normalization constraint which makes  $\lambda_1$  the only free parameter

and the Lagrangian dual is

$$\max_{(\lambda_1, \lambda_2) \in \mathbb{R}_+^2} \left\{ \begin{array}{l} \min \quad -y + \lambda_1 g_1(x, y) + \lambda_2 g_2(x, y) \\ \text{s.t.} \quad (x, y) \in [0, 1]^2 \end{array} \right\}.$$

The value of the surrogate dual is  $S(0.56, 0.44) \approx -0.38$ , and the value of the Lagrangian dual is  $L(0.67, 0.82) \approx -0.78$ . Neither dual proves optimality of  $(x^*, y^*)$ .

*Contribution* In this paper, we revisit surrogate duality in the context of mixed-integer nonlinear programming. To the best of our knowledge, surrogate relaxations have never been considered for solving general MINLPs. The first contribution of the paper is an algorithm capable of solving a *generalized* surrogate dual that allows for multiple aggregations of the nonlinear constraints simultaneously. We also prove its convergence guarantees. Secondly, we present computational enhancements to make the algorithm practical. Our developed algorithm allows us to compare the performance of the classical surrogate relaxation with the generalized one. Finally, we provide an exhaustive computational analysis using publicly available benchmark instances, and we show the practical utility of our proposed approach.

*Structure* The rest of the paper is organized as follows. In Sect. 2, we present a literature review of surrogate duality. Sect. 3 discusses an algorithm from the literature for solving the classic surrogate dual problem and our new computational enhancements. In Sect. 4, we review a generalization of surrogate relaxations from the literature. Afterward, in Sect. 5, we adapt an algorithm for the classic surrogate dual problem to the general case and prove its convergence. An exhaustive computational study on publicly available benchmark instances is given in Sect. 6. Afterward, Sect. 7 presents ideas for future work on how to include surrogate relaxations in the tree search of spatial branch and bound. Section 8 presents concluding remarks.

## 2 Background

Surrogate constraints were first introduced by Glover [21] in the context of zero-one linear integer programming problems. He defined the *strength* of a surrogate constraint according to the dual bound achieved by it—the same notion we use. Extensions were provided by Balas [5] and Geoffrion [20], although under an alternative notion of strength. Glover [22] provided a unified view on these approaches and proposed a generalization where only a subset of constraints are used in an aggregation, leaving the rest explicitly enforced.

A theoretical analysis of surrogate duality in a nonlinear setting was first presented by Greenberg and Pierskalla [27]. This includes results such as quasi-concavity of the surrogate relaxation and a comparison with the Lagrangian dual. They also proposed a generalization using multiple *disjoint* aggregation constraints. A similar generalization allowing multiple aggregations was later proposed by Glover [23]. These were proposed without computational evaluation. Regarding the link with Lagrangian duality, Karwan and Rardin [35] presented necessary conditions for having no gap between the Lagrangian and surrogate duals. Constraint qualification conditions for the surrogate dual have been exhaustively studied, see [23,49,58] and references therein.

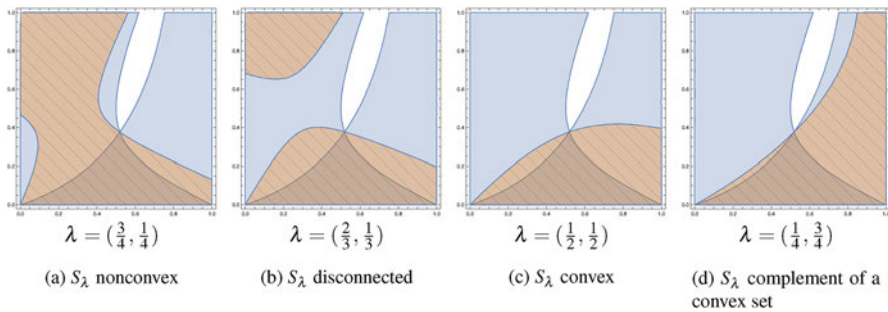
The first algorithmic method for solving (3) is attributed to Banerjee [7]. In the context of integer linear programming, he proposed a Benders' approach similar to the one considered by us in order to find the best multipliers. Karwan [34] expanded on this approach, including a refinement of that of Banerjee and subgradient-based methods. Independently, Dyer [17] proposed similar methods to those of Karwan. Karwan and Rardin [35,38] further developed both Benders- and subgradient-based approaches. Other search procedures for solving (3) involve consecutive Lagrangian dual searches [39,53] and heuristics [19].

From a different perspective, Karwan and Rardin [37] described the interplay between the branch-and-bound trees of an integer programming problem and its surrogate relaxations. Later on, Sarin et al. [54] showed how to integrate their Lagrangian-based multiplier search proposed in [53] into branch and bound.

Surrogate constraints have been used in various applications: in primal heuristics for IPs [24], knapsack problems with a quadratic objective function [15], the job shop problem [18], generalized assignment problems [46], among others. We refer the reader to [3,25] for reviews on surrogate duality methods including other applications and alternative methods for generating surrogate constraints not based on aggregations.

To the best of our knowledge, the efforts for practical implementations of multiplier search methods have mainly focused on *linear* integer programs. This focus can be explained by the maturity of computational optimization tools available during the time where most of these implementations have been developed. We are only aware of two exceptions: the entropy approach to nonlinear programming (see [59,67]) which uses an entropy-based surrogate reformulation instead of a weighted sum of the constraints; and the work by Nakagawa [45], who considered *separable* nonlinear integer programming and presented a novel, albeit expensive, algorithm for solving the surrogate dual.

Regarding the generalization of the surrogate dual that considers multiple aggregated constraints, we are not aware of any work considering a multiplier search method



**Fig. 2** Surrogate relaxations for the nonconvex optimization problem in Example 1. The feasible region defined by each nonlinear constraint is blue and the set  $S_\lambda$  is orange (line shaded).  $S_\lambda$  can be nonconvex, disconnected, convex, or reverse convex (color figure online)

with provable guarantees or a computational implementation of a heuristic approach. We are only aware of the discussion by Karwan and Rardin [36] where they argue that the lack of desirable structures (such as quasi-concavity) may impair some multiplier searches for the surrogate dual generalizations proposed by Greenberg and Pierskalla [27] and Glover [23].

### 3 Surrogate duality in MINLPs

While surrogate duality in its broader definition can be applied to any MINLP, to the best of our knowledge, only mixed-integer *linear* programming problems have been considered for practical applications. Much less attention has been given to the general MINLP case, due to the potential nonconvexity of the resulting problems. Figure 2 illustrates the possible drawbacks and benefits of a nonconvex surrogate relaxation, namely, potentially tight convex relaxations (Fig. 2c), nonconvex (Fig. 2a and 2d), and even disconnected (Fig. 2b) feasible regions.

In this section, we investigate the trade-off between the computational effort required to solve surrogate relaxations and the quality of the resulting bounds. We review how to solve the surrogate dual (3) via a Benders' algorithm (see [7,17,34]) and how to overcome some of the computational difficulties that arise. As we mentioned in Sect. 2, alternative algorithms for solving the surrogate dual exist. However, we use a Benders' approach because its extension to the generalized surrogate dual problem (Sect. 4) is more direct. It is unclear if, e.g., subgradient-based algorithms can be extended to work for the generalization.

#### 3.1 Solving the surrogate dual via Benders

The Benders' algorithm is an iterative approach that alternates between solving a master and a sub-problem. The master problem searches for an aggregation vector  $\lambda \in \mathbb{R}_+^m$  and the sub-problem solves (2) to evaluate  $S(\lambda)$ . Note that the value of an optimal solution  $\bar{x}$  of  $S(\lambda)$ , i.e.,  $c^T \bar{x}$ , is a valid dual bound for (1). To ensure that the

**Algorithm 1:** Algorithm for the surrogate dual.

**Input:** MINLP of the form (1), threshold  $\epsilon > 0$   
**Output:** optimal value  $D \in \mathbb{R}$  of the surrogate dual, subject to  $\epsilon$ -feasibility  
 1: initialize  $\lambda \leftarrow 0 \in \mathbb{R}_+^m, \Psi \leftarrow \infty, \mathcal{X} \leftarrow \emptyset, D \leftarrow -\infty$   
 2: **while**  $\Psi \geq \epsilon$  **do**  
 3:    $\bar{x} \leftarrow \operatorname{argmin}_x \{c^T x \mid x \in S_\lambda\}$   
 4:    $D \leftarrow \max\{D, c^T \bar{x}\}$   
 5:    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\bar{x}\}$   
 6:    $(\lambda, \Psi) \leftarrow$  optimal solution of (6) for  $\mathcal{X}$   
 7: **end while**  
 8: **return**  $D$

point  $\bar{x}$  is not considered in later iterations, i.e.,  $\bar{x} \notin S_{\lambda'}$ , the Benders' algorithm uses the master problem to compute a new vector  $\lambda' \in \mathbb{R}_+^m$  that ensures  $\sum_{i \in \mathcal{M}} \lambda'_i g_i(\bar{x}) > 0$ . This can be done by maximizing constraint violation. More precisely, given  $\mathcal{X}$  the set of previously generated points of the sub-problems, the master problem is the following linear program:

$$\begin{aligned} & \max_{\Psi, \lambda} \Psi \\ & \text{s.t. } \sum_{i \in \mathcal{M}} \lambda_i g_i(\bar{x}) \geq \Psi \quad \text{for all } \bar{x} \in \mathcal{X}, \\ & \quad \|\lambda\|_1 \leq 1, \\ & \quad \lambda \in \mathbb{R}_+^m. \end{aligned} \tag{6}$$

The normalization constraint  $\|\lambda\|_1 \leq 1$  can be added without loss of generality, and it is used to avoid unboundedness of (6). The resulting scheme, formalized in Algorithm 1, terminates once the solution value of (6) is smaller than a fixed value  $\epsilon \geq 0$ . Note that, with this stopping criterion, the best bound  $D$  found by the algorithm is “subject to  $\epsilon$ -feasibility”. This means that when the algorithm stops, it holds that

$$\forall \lambda \in \mathbb{R}_+^m, \|\lambda\|_1 \leq 1, \exists \bar{x} \in \mathcal{X}, c^T \bar{x} \leq D \wedge \sum_{i \in \mathcal{M}} \lambda_i g_i(\bar{x}) < \epsilon.$$

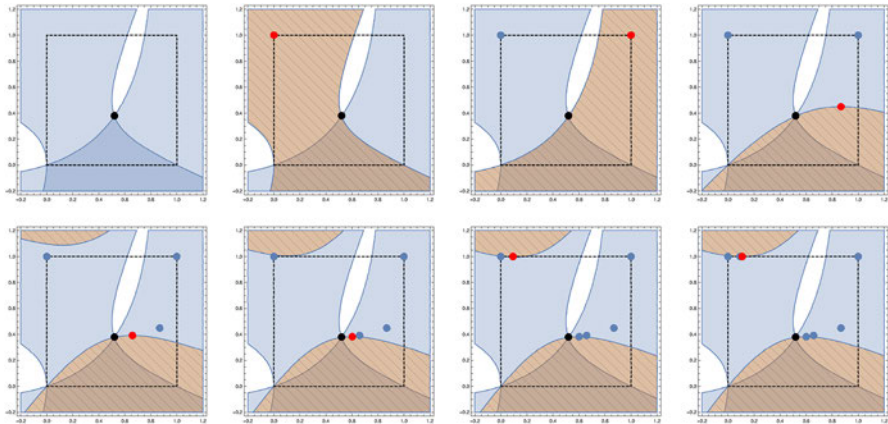
If a numerical feasibility tolerance of  $\epsilon$  is used in a solver (which is usually the case), then for every surrogate relaxation, some  $\bar{x} \in \mathcal{X}$  would be considered feasible and thus prevent the dual bound from improving. The consideration of  $\epsilon$  here is meant mostly for accommodating these numerical tolerances.

An illustration of the algorithm for Example 1 is given in Fig. 3.

**Remark 1** Instead of finding an aggregation vector that maximizes the violation of all points in  $\mathcal{X}$ , Dyer [17] uses an interior point for the polytope in (6). This can be achieved by scaling  $\Psi$  in each constraint of (6) depending on the values  $g_i(\bar{x})$  for each  $i \in \mathcal{M}$ . In our experiments, however, we have observed that maximizing the violation significantly improves the quality of the computed dual bounds.

Using the analysis in [7], Karwan [34] proved the following in the linear case.





**Fig. 3** Algorithm 1 applied to Example 1. The black point is the optimal solution to the original problem, the dashed lines correspond to the variable bounds, the light blue regions are the feasible sets of each nonlinear constraint, and the orange (line shaded) region is  $S_\lambda$  for the different values of  $\lambda \in \mathbb{R}_+^2$  that are computed by the algorithm. The red points are the optimal solutions at each iteration and any red point becomes blue in the next iteration as part of the set  $\mathcal{X}$ . The algorithm converges after seven iterations, and the best multiplier  $\lambda^* \approx (0.56, 0.44)$  was found after five iterations (color figure online)

**Theorem 1** Denote by  $((\lambda^t, \Psi^t))_{t \in \mathbb{N}}$  the sequence of values obtained in Step 6 of Algorithm 1 for  $\epsilon = 0$ . Let  $OPT$  be the value of the surrogate dual (3). If all  $g_i$  are linear for all  $i \in \mathcal{M}$ , then Algorithm 1 either

- terminates in  $T$  steps, in which case  $\max_{1 \leq t \leq T} S(\lambda^t) = OPT$ , or
- the sequence  $(S(\lambda^t))_{t \in \mathbb{N}}$  has a sub-sequence converging to  $OPT$ .

We prove a stronger version of this theorem in Sect. 5 that also works for nonlinear constraints. This stronger result, in particular, shows finite convergence of the algorithm for  $\epsilon > 0$ . Note that the convergence of the algorithm only relies on the solution of an LP and a nonconvex problem  $S(\lambda)$ , and does not make any assumption on the nature of  $S(\lambda)$  besides the fact that it can be solved.

### 3.2 Algorithmic enhancements

In this section, we present computational enhancements that speed up Algorithm 1 and improve the quality of the dual bound that can be achieved from (3). For the sake of completeness, we also discuss techniques that have been tested but did not improve the quality of the computed dual bounds significantly.

#### 3.2.1 Refined MILP relaxation

Instead of only using the initial linear constraints  $Ax \leq b$  to define  $X$  in (1), we exploit a linear programming (LP) relaxation of (1) that is available in LP-based spatial branch and bound. This relaxation contains  $Ax \leq b$  but also linear constraints that have been derived from, e.g., integrality restrictions of variables (e.g., MIR cuts

[48] and Gomory cuts [26]), gradient cuts [32], RLT cuts [56], SDP cuts [57], or other valid underestimators for each  $g_i$  with  $i \in \mathcal{M}$ . Using a linear relaxation  $A'x \leq b'$  with

$$X' := \{x \in X \mid A'x \leq b'\} \tag{7}$$

in the definition of  $S_\lambda$  improves the value of (3) because a relaxed version of the nonlinear constraint  $g_i(x) \leq 0$  is captured in  $S_\lambda$  even if  $\lambda_i$  is zero.

Another way to strengthen the linear relaxation is to make use of objective cutoff information. If there is a feasible, but not necessarily optimal, solution  $x^*$  to (1), the linear relaxation can be strengthened with the inequality  $c^\top x \leq c^\top x^*$ . This inequality preserves all optimal solutions of (1) and can improve the performance when solving (3) (for example, an objective cutoff can lead to dual reductions found by reduced-cost strengthening).

### 3.2.2 Dual objective cutoff in the sub-problem

An undesired phenomenon in Algorithm 1 is that the sequence of dual bounds provided by  $c^\top \bar{x}$  in Step 3 might not be monotone. One way to overcome this problem is to add a *dual objective cutoff*  $c^\top x \geq D$  to the sub-problem  $S(\lambda)$ , thus enforcing monotonicity in the sequence of dual bounds. This does not change the convergence/correctness guarantees of Algorithm 1 and it can improve the progress of the subsequent dual bounds. Moreover, such cutoff can be used to filter the set  $\mathcal{X}$  and thus reduce the size of the LP (6). For example, in Fig. 3 the two last iterations could be avoided.

However, this cutoff has an unfortunate drawback: it increases degeneracy, which affects essential components of a branch-and-bound solver, e.g., pseudocost branching [9]. In the case of Algorithm 1, adding this cutoff significantly increases the time for solving the sub-problem, resulting in an overall negative effect on the algorithm. We confirmed this with extensive computational experiments and decided not to include this feature in our final implementation.

Fortunately, we can still carry dual information through different iterations and improve the performance of the algorithm.

### 3.2.3 Early stopping in the sub-problem

One important ingredient to speed up Algorithm 1, proposed by Karwan [34] and Dyer [17] independently, is an early stopping criterion while solving  $S(\lambda)$ . In our setting, problem  $S(\lambda)$  is the bottleneck of Algorithm 1.

Assume that Algorithm 1 proved a dual bound  $D$  in some previous iteration. It is possible to stop the solving process of  $S(\lambda)$  if a point  $\bar{x} \in S_\lambda$  with  $c^\top \bar{x} \leq D$  has been found. The point  $\bar{x}$  both provides a new inequality for (6) violated by  $\lambda$  (as  $\bar{x} \in S_\lambda$ ) and shows  $S(\lambda) \leq D$ , i.e.,  $\lambda$  will not lead to a better dual bound. All convergence and correctness statements regarding Theorem 1 remain valid after this modification.

Furthermore, we can apply the same idea for any choice of  $D$ . In this scenario,  $D$  would act as a *target* dual bound that we want to prove. Since the Benders' algorithm is computationally expensive, one might require a minimum improvement in the dual

bound. Empirically, we observed that solving  $S(\lambda)$  to global optimality for difficult MINLPs requires a lot of time. However, finding a good quality solution for  $S(\lambda)$  is usually fast. This allows us to early stop most of the sub-problems and only spend time on those sub-problems that will likely result in a dual bound that is at least as good as the target value  $D$ .

### 3.3 Initial empirical observations

For the implementation of Algorithm 1, we use the MINLP solver SCIP

1. to construct a linear relaxation  $A'x \leq b'$  for (1),
2. to find a feasible solution  $x^*$  to (1) to be used as an objective cutoff  $c^T x \leq c^T x^*$  when solving  $S(\lambda)$ , and
3. to use it as a black box to solve each  $S(\lambda)$  sub-problem.

We provide extensive details of our implementation and the results in Sect. 6, but we would like to summarize some important observations to the reader.

Our proposed algorithmic enhancements proved to be key for obtaining a practical algorithm for the surrogate dual, especially the use of a refined MILP relaxation. The achieved dual bounds by only using the initial linear relaxation in Algorithm 1 were almost always dominated by the dual bounds obtained by the refined MILP relaxation. Thus, utilizing the refined MILP relaxation seems mandatory for obtaining strong surrogate relaxations. Our computational study in Sect. 6 shows that our algorithmic enhancements of Algorithm 1 allows us to compute dual bounds that close on average 35.0% more gap (w.r.t. the best known primal bound) than the dual bounds obtained by the refined MILP relaxations, i.e.,  $S(0)$ , on 469 affected instances.

While the overall impact of this “classic” surrogate duality is positive, we observed that the dual bound deteriorates with increasing number of nonlinear constraints; aggregating a *large* number of nonconvex constraints into a single constraint may not capture the structure of the underlying MINLP. For this reason, in the next Section we propose the use of a generalized surrogate relaxation for solving MINLPs, which includes multiple aggregation constraints.

## 4 Generalized surrogate duality

In the following, we discuss a generalization of surrogate relaxations that has been introduced by [23]. Instead of a single aggregation, it allows for  $K \in \mathbb{N}$  aggregations of the nonlinear constraints of (1). The nonnegative vector  $\lambda = (\lambda^1, \lambda^2, \dots, \lambda^K) \in \mathbb{R}_+^{Km}$  encodes these  $K$  aggregations

$$\sum_{i \in \mathcal{M}} \lambda_i^k g_i(x) \leq 0, \quad k \in \{1, \dots, K\}$$

of the nonlinear constraints. Note that, for convenience, we are denoting as  $\lambda_i^k$  the component  $\lambda_{(k-1)m+i}$  of the  $\lambda$  vector, as it can also be seen as the  $i$ -th component

of the  $\lambda^k$  sub-vector. For a vector  $\lambda \in \mathbb{R}_+^{Km}$ , the feasible region of the  $K$ -surrogate relaxation is given by the intersection

$$S_\lambda^K := \bigcap_{k=1}^K S_{\lambda^k},$$

where  $S_{\lambda^k}$  is the feasible region of the surrogate relaxation  $S(\lambda^k)$  for  $\lambda^k \in \mathbb{R}_+^m$ . It clearly follows that  $S_\lambda^K$  is a relaxation for (1). The best dual bound for (1) generated by a  $K$ -surrogate relaxation is given by

$$\sup_{\lambda \in \mathbb{R}_+^{Km}} S^K(\lambda), \tag{8}$$

which we call the  $K$ -surrogate dual. Note that scaling each  $\lambda^k \in \mathbb{R}_+^m$  individually by a positive scalar does not affect the value of  $S^K(\lambda)$ , i.e.,

$$S^K(\dots, \lambda^k, \dots) = S^K(\dots, \alpha \lambda^k, \dots)$$

for any  $\alpha > 0$ . Therefore, it is possible to impose additional normalization constraints  $\|\lambda^k\|_1 \leq 1$  for each  $k \in \{1, \dots, K\}$ .

In [27], a related generalization was proposed, although not computationally tested. The paper considers a partition of constraints which are aggregated; equivalently, the support of sub-vectors  $\lambda^k$  are assumed to be fixed and disjoint. Glover’s generalization [23] does not make any assumption on the structure of the  $\lambda^k$  sub-vectors. As we argue below, this makes a significant difference for two reasons: (a) selecting the “best” partition of constraints *a-priori* is a challenging task and (b) restricting the support of sub-vectors  $\lambda^k$  to be disjoint can weaken the bound given by (8).

The function  $S^K$  remains lower semi-continuous for any choice of  $K$ . The proof idea is similar to the one given by Glover [23] for the case of  $K = 1$ .

**Proposition 1** *If  $g_i$  is continuous for every  $i \in \mathcal{M}$  and  $X$  is compact, then  $S^K : \mathbb{R}_+^{Km} \rightarrow \mathbb{R}$  is lower semi-continuous for any choice of  $K$ .*

**Proof** Let  $(\lambda^t)_{t \in \mathbb{N}} \subseteq \mathbb{R}_+^{Km}$  a sequence that converges to  $\lambda^*$ . We denote the  $k$ -th sub-vector of  $\lambda^t$  as  $\lambda^{t,k}$ . Denote with  $x^t \in X$  an optimal solution of  $S^K(\lambda^t)$ .

We need to show that  $S^K(\lambda^*) \leq \liminf_{t \rightarrow \infty} S^K(\lambda^t)$ .

By definition, there exists a subsequence  $(\lambda^\tau)_{\tau \in \mathbb{N}}$  of  $(\lambda^t)_{t \in \mathbb{N}}$  such that  $S^K(\lambda^\tau) \rightarrow \liminf_{t \rightarrow \infty} S^K(\lambda^t)$ .

Since  $X$  is compact, there exists a subsequence  $(x^l)_{l \in \mathbb{N}}$  of  $(x^\tau)_{\tau \in \mathbb{N}}$  such that  $\lim_{l \rightarrow \infty} x^l = x^*$ . As  $(\lambda^l)_{l \in \mathbb{N}}$  is a subsequence of  $(\lambda^\tau)_{\tau \in \mathbb{N}}$ , we have that  $\lim_{l \rightarrow \infty} S^K(\lambda^l) = \liminf_{t \rightarrow \infty} S^K(\lambda^t)$ . From  $x^l \in S_{\lambda^l}^K$  it follows that

$$\sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0$$

for every  $k \in \{1, \dots, K\}$ , which is equivalent to

$$\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0.$$

Because the  $g_i$  are continuous and the maximum of continuous functions is still continuous, it follows that

$$\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} (\lambda^*)^k_i g_i(x^*) = \lim_{l \rightarrow \infty} \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0.$$

Hence,  $x^*$  is feasible but not necessarily optimal for  $S^K(\lambda^*)$ . Therefore,

$$S^K(\lambda^*) \leq c^T x^* = \lim_{l \rightarrow \infty} c^T x^l = \lim_{l \rightarrow \infty} S^K(\lambda^l) = \liminf_{l \rightarrow \infty} S^K(\lambda^l).$$

□

One important difference to the classic surrogate dual is that  $S^K(\lambda)$  is no longer quasi-concave, even for the case of  $K = 2$  and two linear constraints, as the following example shows.

**Example 2** Let  $K = 2$  and consider the linear program

$$\begin{aligned} \min \quad & y \\ \text{s.t.} \quad & 4x - 8y + 3.2 \leq 0, \\ & 5x - y - 1.5 \leq 0, \\ & (x, y) \in [0, 1]^2. \end{aligned}$$

Due to the symmetry of the generalized surrogate dual,  $S^2(\lambda) = S^2(\mu) \approx 0.30$  holds for the aggregation vectors  $\lambda := ((0.7, 0.3), (0.3, 0.7))$  and  $\mu := ((0.3, 0.7), (0.7, 0.3))$ . However, using the convex combination  $\lambda/2 + \mu/2$  we have that  $S^2(\lambda/2 + \mu/2) \approx 0.19$ , which is smaller than  $S^2(\lambda)$  and  $S^2(\mu)$  and thereby shows that  $S^2$  is not quasi-concave.

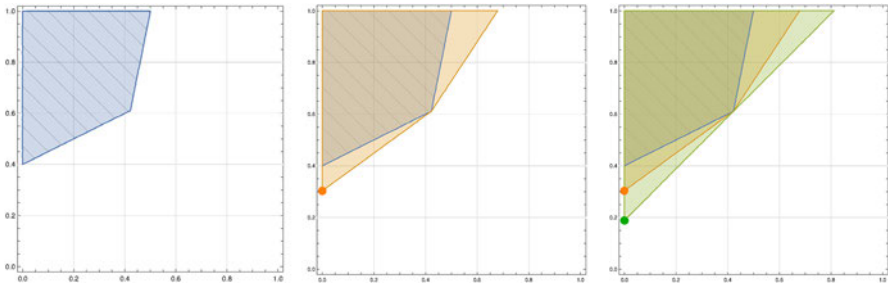
See Fig. 4 for an illustration of the counterexample.

Since  $S^K$  may not be quasi-concave, gradient descent-based algorithms for optimizing (3), as in [34], become inadequate for solving (8). Even though (8) is substantially more difficult to solve than (3), it might be beneficial to consider larger  $K$  to obtain tight relaxations for (1). The following proposition formalizes this. We omit its proof as it is elementary.

**Proposition 2** *The inequality*

$$\sup_{\lambda \in \mathbb{R}_+^{Km}} S^K(\lambda) \leq \sup_{\lambda \in \mathbb{R}_+^{(K+1)m}} S^{K+1}(\lambda) \tag{9}$$

holds for any  $K \in \mathbb{N}$ . Furthermore,  $\sup_{\lambda \in \mathbb{R}_+^{m^2}} S^m(\lambda)$  is equal to the value of (1).



**Fig. 4** A visualization of Example 2 that shows that  $S^K$  is in general not quasi-concave for  $K > 1$ . The blue line shaded region (left) is the feasible set defined by two original inequalities. The orange region (center) depicts both  $S^2_\lambda$  and  $S^2_\mu$ , while the green region (right) is their convex combination  $S^2_{(\lambda+\mu)/2}$ . The example shows  $S^2(\lambda) = S^2(\mu) > S^2((\lambda + \mu)/2)$ , which proves that  $S^2$  is not quasi-concave (color figure online)

The following example shows that going from  $K = 1$  to  $K = 2$  can have a tremendous impact on the quality of the surrogate relaxation:

**Example 3** Consider the following NLP with four nonlinear constraints and four unbounded variables:

$$\begin{aligned} \min \quad & -x - y \\ \text{s.t.} \quad & x^3 - z \leq 0 \\ & x^3 + z \leq 0 \\ & y^2 + w \leq 0 \\ & y^2 - w \leq 0 \end{aligned}$$

It is easy to see that  $(0, 0, 0, 0)$  is the optimal solution.

First, we show that the classic surrogate dual is unbounded. For an aggregation  $\lambda \in \mathbb{R}^4$ , the sole constraint in the corresponding surrogate relaxation is

$$(\lambda_1 + \lambda_2)x^3 + (-\lambda_1 + \lambda_2)z + (\lambda_3 + \lambda_4)y^2 + (\lambda_3 - \lambda_4)w \leq 0.$$

If either  $\lambda_1 \neq \lambda_2$  or  $\lambda_3 \neq \lambda_4$ , then the relaxation is clearly unbounded, as  $z$  and  $w$  are free variables with coefficient 0 in the objective. If  $\lambda_1 = \lambda_2$  and  $\lambda_3 = \lambda_4$ , the aggregation reads  $2\lambda_1x^3 + 2\lambda_3y^2 \leq 0$ . We split this case in two: (1) if  $\lambda_1 = 0$ , then  $(x, y, w, z) = (M, 0, 0, 0)$  is feasible for any  $M$ . (2) Otherwise, we can set  $(x, y, w, z) = (-M, 2M, 0, 0)$  and  $2\lambda_1x^3 + 2\lambda_3y^2 = -2\lambda_1M^3 + 8\lambda_3M^2$ . The last expression is negative for large enough  $M$ .

In both cases the objective evaluates to  $-M$ . Letting  $M \rightarrow \infty$  shows unboundedness.

Now consider the two aggregations  $\lambda = (\lambda^1, \lambda^2)$  with  $\lambda^1 = (1/2, 1/2, 0, 0)$  and  $\lambda^2 = (0, 0, 1/2, 1/2)$ . Using the 2-surrogate relaxation obtained from  $\lambda$  immediately implies  $x \leq 0$  and  $y = 0$ , which proves optimality of  $(0, 0, 0, 0)$ .

### 5 An algorithm for the $K$ -surrogate dual

Even though (8) yields a strong relaxation for large  $K$ , it is computationally more challenging to solve than (3). To the best of our knowledge, there is no algorithm in the literature known that can solve (8). Due to the missing quasi-concavity property of  $S^K$ , it is not possible to adjust each aggregation vector independently; an alternating-type method based on the  $K = 1$  case could provide weak bounds.

In this section, we present the first algorithm for solving (8). The idea of the algorithm is the same as before: a master problem will generate an aggregation vector  $(\lambda^1, \dots, \lambda^K)$  and the sub-problem will solve the  $K$ -surrogate relaxation corresponding to  $(\lambda^1, \dots, \lambda^K)$ . The only differences to Algorithm 1 are that we replace the LP master problem by a MILP master problem and solve  $S^K(\lambda^1, \dots, \lambda^K)$  instead of  $S(\lambda)$ .

*Generalizing Algorithm 1* Assume that we have a solution  $\bar{x}$  of problem  $S^K(\lambda^1, \dots, \lambda^K)$ . In the next iteration, we need to make sure that  $\bar{x}$  is infeasible for *at least one* of the aggregated constraints. This can be written as a disjunctive constraint

$$\bigvee_{k=1}^K \left( \sum_{i \in \mathcal{M}} \lambda_i^k g_i(\bar{x}) > 0 \right). \tag{10}$$

As in (6), we replace the strict inequality by maximizing the activity of  $\sum_{i \in \mathcal{M}} \lambda_i^k g_i(\bar{x})$  for all  $k \in \{1, \dots, K\}$ . The master problem then reads as

$$\begin{aligned} \max_{\Psi, \lambda} \quad & \Psi \\ \text{s.t.} \quad & \bigvee_{k=1}^K \left( \sum_{i \in \mathcal{M}} \lambda_i^k g_i(\bar{x}) \geq \Psi \right) \quad \text{for all } \bar{x} \in \mathcal{X}, \\ & \|\lambda^k\|_1 \leq 1, \lambda^k \in \mathbb{R}_+^m \quad \text{for all } k \in \{1, \dots, K\}, \end{aligned} \tag{11}$$

where  $\mathcal{X}$  is the set of generated points of the sub-problems.

*Solving the master problem* The disjunction in the master problem (11) is encoding  $K^{|\mathcal{X}|}$  many LPs, thus an enumeration approach for tackling the disjunction is impractical. Instead, we use a, so-called, big-M formulation. This enables us to solve (11) with moderate running times. Such MILP formulation of (11) reads

$$\begin{aligned} \max_{\Psi, \lambda} \quad & \Psi \\ \text{s.t.} \quad & \sum_{i \in \mathcal{M}} \lambda_i^k g_i(\bar{x}) \geq \Psi - M(1 - z_k^{\bar{x}}) \quad \text{for all } k \in \{1, \dots, K\}, \bar{x} \in \mathcal{X}, \\ & \sum_{k=1}^K z_k^{\bar{x}} = 1 \quad \text{for all } \bar{x} \in \mathcal{X}, \end{aligned}$$

$$\begin{aligned}
 z_k^{\bar{x}} &\in \{0, 1\} && \text{for all } k \in \{1, \dots, K\}, \bar{x} \in \mathcal{X}, \\
 \|\lambda^k\|_1 &\leq 1, \lambda^k \in \mathbb{R}_+^m && \text{for all } k \in \{1, \dots, K\},
 \end{aligned}
 \tag{12}$$

where  $M$  is a large constant. A binary variable  $z_k^{\bar{x}}$  indicates if the  $k$ -th disjunction of (11) is used to cut off the point  $\bar{x} \in \mathcal{X}$ . Due to the normalization  $\|\lambda^k\|_1 \leq 1$ , it is possible to bound  $M$  by  $\max_{i \in \mathcal{M}} |g_i(\bar{x})|$ . Even more, since the optimal  $\Psi$  values of (12) are non-increasing, we could use the optimal  $\Psi_{prev}$  of the previous iteration as a bound on  $M$ . Thus, it is possible to bound  $M$  by  $\min\{\max_{i \in \mathcal{M}} |g_i(\bar{x})|, \Psi_{prev}\}$ .

**Remark 2** Big-M formulations are typically not considered strong in MILPs, given their usual weak LP relaxations. Other formulations in extended spaces can yield better theoretical guarantees (see, e.g., [6,10,62]). The drawback of these extended formulations is that they require to add copies of the  $\lambda$  variables depending on the number of disjunctions, which in our case is rapidly increasing.

In [61], the author proposes an alternative that does not create variable copies, but that can be costly to construct unless special structure is present.

In our case, as we discuss in Sect. 5.2.2, we do not require a tight LP relaxation of (11) and thus we opted to use (12).

The algorithm for the  $K$ -surrogate dual problem is stated in Algorithm 2. For details on the exact meaning of “subject to  $\epsilon$ -feasibility”, please see the discussion following Algorithm 1. The following example shows that Algorithm 2 can compute significantly better dual bounds than Algorithm 1.

---

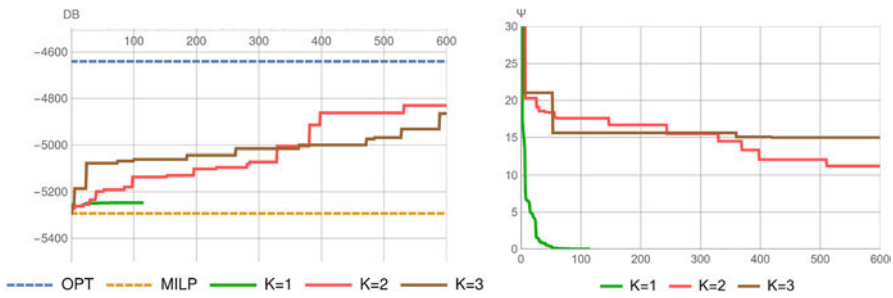
**Algorithm 2:** Algorithm for the  $K$ -surrogate dual.

---

- Input:** MINLP of the form (1), threshold  $\epsilon > 0$ ,  $K \in \mathbb{N}$  aggregations
  - Output:** optimal value  $D \in \mathbb{R}$  of the  $K$ -surrogate dual, subject to  $\epsilon$ -feasibility
  - 1: initialize  $\lambda \leftarrow 0 \in \mathbb{R}_+^{Km}$ ,  $\Psi \leftarrow \infty$ ,  $\mathcal{X} \leftarrow \emptyset$ ,  $D \leftarrow -\infty$
  - 2: **while**  $\Psi \geq \epsilon$  **do**
  - 3:    $\bar{x} \leftarrow \operatorname{argmin}_x \{c^T x \mid x \in S_\lambda^K\}$
  - 4:    $D \leftarrow \max\{D, c^T \bar{x}\}$
  - 5:    $\mathcal{X} \leftarrow \mathcal{X} \cup \{\bar{x}\}$
  - 6:    $(\lambda, \Psi) \leftarrow$  optimal solution of (11) for  $\mathcal{X}$
  - 7: **end while**
  - 8: **return**  $D$
- 

**Example 4** We briefly discuss the results of Algorithm 2 for the `genpooling_lee1` instance from MINLPLib. The instance consists of 20 nonlinear, 59 linear constraints, 9 binary, and 40 continuous variables after preprocessing. The classic surrogate dual, i.e.,  $K = 1$ , could be solved to optimality, whereas for  $K = 2$  and  $K = 3$  the algorithm hit the iteration limit. Nevertheless, the dual bound  $-4829.6$  achieved for  $K = 2$  and the dual bound  $-4864.87$  for  $K = 3$  are significantly better than the dual bound of  $-5246.0$  for  $K = 1$ , see Fig. 5.





**Fig. 5** Results of Algorithm 2 for the instance `genpooling_1ee1` for different choices of  $K$ . The first plot shows the progress of the proved dual bound and the second plot the value of  $\Psi$  for the first 600 iterations. The blue line is the optimal solution value of the MINLP and the yellow line that of the MILP relaxation (color figure online)

### 5.1 Convergence

In the following, we show that the dual bounds obtained by Algorithm 2 converge to the optimal value of the  $K$ -surrogate dual. The idea of the proof is similar to the one presented by [38] for the case of  $K = 1$  and linear constraints.

**Theorem 2** Denote by  $(\lambda^t, \Psi^t)_{t \in \mathbb{N}}$  the sequence of values obtained after solving (11) in Algorithm 2 for  $\epsilon = 0$ . Let  $OPT$  be the optimal value of the  $K$ -surrogate dual (8). The algorithm either

- (a) terminates in  $T$  steps, in which case  $\max_{1 \leq t \leq T} S^K(\lambda^t) = OPT$ , or
- (b)  $\sup_{t \geq 1} S^K(\lambda^t) = OPT$ .

**Proof** As in Proposition 1, we denote the  $k$ -th sub-vector of  $\lambda^t$  as  $\lambda^{t,k}$ . Let  $x^t \in X$  be an optimal solution obtained from solving  $S^K(\lambda^t)$  at iteration  $t$ .

(a) If the algorithm terminates after  $T$  iterations, i.e.,  $\Psi^T = 0$ , then for any choice  $\lambda \in \mathbb{R}^{Km}$ , there is at least one point  $x^1, \dots, x^T$  that is feasible for  $S^K(\lambda)$ . This implies  $OPT = \max_{1 \leq t \leq T} \{c^T x^t\}$ .

(b) Now assume that the algorithm does not converge in a finite number of steps, i.e.,  $\Psi^t > 0$  for all  $t \geq 1$ . Since  $(\Psi^t)_{t \in \mathbb{N}}$  defines a decreasing sequence which is bounded below by 0, it must converge to a value  $\Psi^* \geq 0$ . The same holds for any subsequence of  $(\Psi^t)_{t \in \mathbb{N}}$ . Furthermore, the sequence  $(\lambda^t, x^t)_{t \in \mathbb{N}}$  belongs to a compact set: indeed,  $\|\lambda^t\|_1 \leq 1$  for all  $t \in \mathbb{N}$  and  $X$  is assumed to be compact. Therefore, there exists a subsequence of  $(\Psi^t, \lambda^t, x^t)_{t \in \mathbb{N}}$  that converges. With slight abuse of notation, we denote this subsequence by  $(\Psi^l, \lambda^l, x^l)_{l \in \mathbb{N}}$ . To summarize, we have that

- $\lim_{l \rightarrow \infty} \Psi^l = \Psi^* \geq 0$ ,
- $\lim_{l \rightarrow \infty} \lambda^l = \lambda^*$ , and
- $\lim_{l \rightarrow \infty} x^l = x^*$ .

for some  $(\Psi^*, \lambda^*, x^*)$ .

First, we show  $\Psi^* = 0$ . Note that  $x^l$  is an optimal solution to  $S^K(\lambda^l)$ . This means that  $x^l$  satisfies all aggregation constraints, i.e.,  $\sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0$  for all  $k = 1, \dots, K$ , which is equivalent to the inequality  $\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0$ .

After solving (11), we know that  $\Psi^l$  is equal to the minimum violation of the disjunction constraints for the points  $x^1, \dots, x^{l-1}$ . This implies the inequality

$$\Psi^l = \min_{1 \leq t \leq l-1} \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^t) \leq \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^{l-1}),$$

which uses the fact that the minimum over all points  $x^1, \dots, x^{l-1}$  is bounded by the value for  $x^{l-1}$ . Both inequalities combined show that

$$\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^l) \leq 0 < \Psi^l \leq \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \lambda_i^{l,k} g_i(x^{l-1})$$

for all  $l \geq 0$ . Using the continuity of  $g_i$  and the fact that the maximum of finitely many continuous functions is continuous, we obtain

$$\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} (\lambda^*)^k_i g_i(x^*) \leq 0 \leq \Psi^* \leq \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} (\lambda^*)^k_i g_i(x^*),$$

which shows  $\Psi^* = 0$ .

Next, we show that  $\sup_{t \geq 1} S^K(\lambda^t) = OPT$ .

Clearly,  $\sup_{t \geq 1} S^K(\lambda^t) \leq OPT$ .

Let us now prove that  $\sup_{t \geq 1} S^K(\lambda^t) \geq OPT$ .

Take any  $\epsilon > 0$  and let  $\bar{\lambda}$  be such that  $S^K(\bar{\lambda}) \geq OPT - \epsilon$ . Such  $\bar{\lambda}$  always exists by definition of  $OPT$ . Furthermore, via a re-scaling we may assume  $\|\bar{\lambda}^k\| \leq 1$  for all  $k \in \{1, \dots, K\}$ .

By definition,

$$\Psi^l \geq \min_{1 \leq t \leq l-1} \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \bar{\lambda}_i^k g_i(x^t).$$

Computing the limit when  $l$  goes to infinity, we obtain

$$0 \geq \inf_{1 \leq t} \max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \bar{\lambda}_i^k g_i(x^t).$$

Let  $\bar{x}$  be  $x^{t_0}$  if the infimum is achieved at  $t_0$  or  $x^*$  if the infimum is not achieved. Then,

$$\max_{1 \leq k \leq K} \sum_{i \in \mathcal{M}} \bar{\lambda}_i^k g_i(\bar{x}) \leq 0.$$

This last inequality implies that  $\bar{x}$  is feasible for  $S^K(\bar{\lambda})$ .

Hence,

$$OPT - \epsilon \leq S^K(\bar{\lambda}) \leq c^T \bar{x} \leq \sup_{t \geq 1} S^K(\lambda^t).$$

Since  $\epsilon > 0$  is arbitrary, we conclude that  $\sup_{t \geq 1} S^K(\lambda^t) \geq OPT$ .  $\square$

The proof of Theorem 2 shows that  $(\Psi^t)_{t \in \mathbb{N}}$  always converges to zero. Therefore, the proof also establishes that Algorithm 2 terminates after a finite number of steps for any  $\epsilon > 0$ .

## 5.2 Computational enhancements

We now discuss computational enhancements additional to those discussed in Sect. 3.2 for improving the performance of Algorithm 2. Due to the increasing complexity of the master problem with each iteration, solving the MILP (11) is for most instances the bottleneck of Algorithm 2. For this reason, most of our computational enhancements are devoted to reduce the computational effort spent in the master problem.

As in the case  $K = 1$ , we also report techniques that we did not include in our final implementation.

### 5.2.1 Multiplier symmetry breaking

One difficulty of the  $K$ -surrogate dual is that (11) and (12) might contain many equivalent solutions. For example, any permutation  $\pi$  of the set  $\{1, \dots, K\}$  implies that the sub-problem  $S^K(\lambda)$  with  $\lambda = (\lambda^1, \dots, \lambda^K)$  is equivalent to  $S^K(\lambda^\pi)$  with  $\lambda^\pi = (\lambda^{\pi_1}, \dots, \lambda^{\pi_K})$ . This can heavily impact the solution time of the master problem. We refer to [40] for an overview of symmetry in integer programming. Ideally, in order to break symmetry, we would like to impose  $\lambda^1 \succeq_{\text{lex}} \lambda^2 \succeq_{\text{lex}} \dots \succeq_{\text{lex}} \lambda^K$  where  $\succeq_{\text{lex}}$  indicates a lexicographical order. Such order can be imposed using linear constraints with additional binary variables and big-M constraints. However, this yields prohibitive running times. Two more efficient alternatives that only partially break symmetries, are as follows. First, the constraints

$$\lambda_1^1 \geq \lambda_1^2 \geq \dots \geq \lambda_1^K \quad (13)$$

enforce that  $\lambda^1, \dots, \lambda^K$  are sorted with respect to the first component. The drawback of this sorting is that if  $\lambda_1^k = 0$  for all  $k \in \{1, \dots, K\}$ , then (13) does not break any of the symmetry of (12). Our second idea for breaking symmetry is to use

$$\lambda_j^j \geq \lambda_j^k \quad \text{for all } j \in \{1, \dots, K-1\} \text{ for all } k \in \{j+1, \dots, K\}. \quad (14)$$

In our experiments, we observed that slightly better dual bounds could be computed when using (13) instead of (14). However, the overall impact was not significant, and we decided to not include this in our final implementation.

### 5.2.2 Early stopping of the master problem

Solving (12) to optimality in every iteration of Algorithm 2 is computationally expensive for  $K \geq 2$ . On one hand, the true optimal value of  $\Psi$  is needed to decide whether

the algorithm terminated. On the other hand, to ensure progress of the algorithm it is enough to only compute a feasible point  $(\Psi, \lambda^1, \dots, \lambda^K)$  of (12) with  $\Psi > 0$ . We balance these two opposing forces with the following early stopping method.

While solving (12), we have access to both a valid dual bound  $\Psi_d$  and primal bound  $\Psi_p$  such that the optimal  $\Psi$  is contained in  $[\Psi_p, \Psi_d]$ . Note that the primal bound can be assumed to be nonnegative as the vector of zeros is always feasible for (12). Let  $\Psi_d^t$  and  $\Psi_p^t$  be the primal and dual bounds obtained from the master problem in iteration  $t$  of Algorithm 2. We stop the master problem in iteration  $t + 1$  as soon as  $\Psi_p^{t+1} \geq \alpha \Psi_d^t$  holds for a fixed  $\alpha \in (0, 1]$ . The parameter  $\alpha$  controls the trade-off between proving a good dual bound  $\Psi_d^{t+1}$  and saving time for solving the master problem. On the one hand,  $\alpha = 1$  implies

$$\Psi_p^{t+1} \geq \alpha \Psi_d^t \geq \alpha \Psi_d^{t+1} = \Psi_d^{t+1},$$

which can only be true if  $\Psi_p^{t+1} = \Psi_d^{t+1}$  holds. This equality proves optimality of the master problem in iteration  $t + 1$ . On the other hand, setting  $\alpha$  close to zero means that we would stop as soon as a non-trivial feasible solution to the master problem has been found. In our experiments, we observed that setting  $\alpha$  to 0.2 performs well.

### 5.2.3 Constraint filtering

Another potential way of alleviating the computational burden of solving the master problem, is to reduce the set of nonlinear constraints to only those that are needed for a good quality solution of (8). Of course, this set of constraints is unknown in advance and challenging to compute because of the nonconvexity of the MINLP.

We tested different heuristics to preselect nonlinear constraints. We used the violation of the constraints with respect to the LP, MILP, and convex NLP relaxation of the MINLP, as measures of “importance” of nonlinear constraints. We also used the connectivity of nonlinear constraints in the variable-constraint graph<sup>1</sup> for discarding some constraints. Unfortunately, we could not identify a good filtering rule that results in strong bounds for (8).

However, we were able to find a way of reducing the number of constraints considered in the master problem without imposing a strong *a-priori* filter: an adaptive filtering, which we call *support stabilization*. We specify this next.

### 5.2.4 Support stabilization

Direct implementations of Benders-based algorithms, much like column generation approaches, are known to suffer from convergence issues. Deriving “stabilization” techniques that can avoid oscillations of the  $\lambda$  variables and tailing-off effects, among others, are a common goal for improving performance, see, e.g., [4,16,60].

We developed a *support stabilization* technique to address the exponential increase in complexity of the master problem (12) and to prevent the oscillations of the  $\lambda$

<sup>1</sup> Bipartite graph where each variable and each constraint are represented as nodes, and edges are included when a variable appears in a constraint.

variables. Once Algorithm 2 finds a multiplier vector that improves the overall dual bound, we restrict the support to that of the improving dual multiplier. This restricts the search space and drastically improves solution times. Once stalling is detected (which corresponds to finding a local optimum of (8)), we remove the support restriction until another multiplier vector that improves the dual bound is found. This technique not only improves solution times, but also leads to better bounds on (8) in fewer iterations.

### 5.2.5 Trust-region stabilization

While the previous stabilization alleviates some of the computational burden in the master problem, the non-zero entries of the  $\lambda$  vectors can (and do, in practice) vary significantly from iteration to iteration. To remedy this, we incorporated a classic stabilization technique: a *box trust-region* stabilization, see [14]. Given a reference solution  $(\hat{\lambda}^1, \dots, \hat{\lambda}^k)$ , we impose the following constraint in (12)

$$\|(\lambda^1, \dots, \lambda^k) - (\hat{\lambda}^1, \dots, \hat{\lambda}^k)\|_\infty \leq \delta$$

for some parameter  $\delta$ . This prevents the  $\lambda$  variables from oscillating excessively. In addition, by removing the trust-region when stalling is detected, we are able to preserve the convergence guarantees of Theorem 2. In our implementation, we maintain a fixed  $(\hat{\lambda}^1, \dots, \hat{\lambda}^k)$  until we obtain a bound improvement or the algorithm stalls. When any of this happens, we remove the box and compute a new  $(\hat{\lambda}^1, \dots, \hat{\lambda}^k)$  with (12) without any stabilization added.

**Remark 3** We also tested another stabilization technique inspired by column generation's *smoothing* by [65] and [47]. Let  $\lambda^{best}$  be the best found primal solution so far and let  $\lambda^{new}$  be the solution of the current master problem. Instead of using  $\lambda^{new}$  as a new multiplier vector, we choose a convex combination between  $\lambda^{best}$  and  $\lambda^{new}$ .

While this stabilization technique improved the performance of Algorithm 2 with respect to the algorithm with no stabilization, it performed significantly worse than the trust-region stabilization. Therefore, we did not include it in our final implementation.

## 6 Computational experiments

In this section, we present a computational study of the classic and generalized surrogate duality on publicly available instances of the MINLPLib [42]. We conduct three main experiments to answer the following questions:

1. ROOTGAP: How much of the root gap with respect to the MILP relaxation can be closed by using the  $K$ -surrogate dual?
2. BENDERS: How much do the ideas of Sect. 5 improve the performance of Algorithm 2?
3. DUALBOUND: Can Algorithm 2 improve on the dual bounds obtained by the MINLP solver SCIP?

Our ideas are embedded in the MINLP solver SCIP [55]. We refer to [1,63,64] for an overview of the general solving algorithm and MINLP features of SCIP.

## 6.1 Experimental setup

In all experiments, Algorithm 2 is called after the root node has been completely processed by SCIP. All generated and initial linear inequalities are added to  $S_\lambda$ .

For the ROOTGAP experiment, we run Algorithm 2 for one hour for each choice of  $K \in \{1, 2, 3\}$ . To measure the improvement in using  $K + 1$  instead of  $K$ , we use the best found aggregation vector of  $K$  as an initial point for  $K + 1$ .

In the BENDERS experiment we focus on  $K = 3$  and do not start with an initial point for the aggregation vector. This allows us to better analyze the impact of each proposed enhancement. We compare the following settings:

- DEFAULT: Algorithm 2 with all enhancements described in Sects. 3.2 and 5.2.
- PLAIN: Plain version of Algorithm 2 with no enhancement.
- NOSTAB: Same as DEFAULT but without using the trust-region of Sect. 5.2.5 and support stabilization of Sect. 5.2.4.
- NOSUPP: Same as DEFAULT but without using the support stabilization.
- NOEARLY: Same as DEFAULT but without using early termination for the master problem, described in Sect. 5.2.2.

Each of the five settings uses a time limit of one hour.

Finally, in the DUALBOUND experiment we evaluate how much the dual bounds obtained by SCIP can be improved by Algorithm 2. First, we collect the dual bounds for all instances that could not be solved by SCIP within three hours. Afterward, we apply Algorithm 2 for  $K = 3$ , a time limit of three hours, and set a target dual bound (see Sect. 3.2.3) of

$$D + (P - D) \cdot 0.2,$$

where  $D$  is the dual bound obtained by default SCIP and  $P$  be the best known primal bound reported in the MINLPLib. This means that we aim for a gap closed reduction of at least 20% and early stop each sub-problem in Algorithm 2 that will provably lead to a smaller reduction.

During all three experiments, we use a gap limit of  $10^{-4}$  for each sub-problem of Algorithm 2. Additionally, we chose a dual feasibility tolerance of  $10^{-8}$  (SCIP's default is  $10^{-7}$ ) and a primal feasibility tolerance of  $10^{-7}$  (SCIP's default is  $10^{-6}$ ).

*Stabilization details* The trust-region and support stabilization have been implemented as follows. Both stabilization methods are applied once an improving aggregation  $\lambda^*$  could be found. Each entry  $\lambda_i$  with  $\lambda_i^* = 0$  is fixed to zero. Otherwise, the domain of  $\lambda_i$  is restricted to the interval

$$[\max\{0, \lambda_i^* - 0.1\}, \min\{1, \lambda_i^* + 0.1\}].$$

Once a new improving solution has been found, we update the trust region accordingly. We remove the trust region and support stabilization in case no improving solution could be found for 20 iterations.

*Test set* We used the publicly available instances of the MINLPLib [42], which at time of the experiments contained 1683 instances. We selected the instances that were

available in OSiL format and consisted of nonlinear expressions that could be handled by SCIP, in total 1671 instances.

*Gap closed* We use the *gap closed* in order to compare dual bounds. Let  $d_1 \in \mathbb{R}$  and  $d_2 \in \mathbb{R}$  be two dual bounds for (1) and  $p \in \mathbb{R}$  a reference primal bound. The *gap closed* improvement is measured by the function  $GC : \mathbb{R}^3 \rightarrow [-1, 1]$  defined as

$$GC(p, d_1, d_2) := \begin{cases} 0, & \text{if } d_1 = d_2 \\ +1 - \frac{p-d_1}{p-d_2}, & \text{if } d_1 > d_2 \\ -1 + \frac{p-d_2}{p-d_1}, & \text{if } d_1 < d_2 \end{cases}$$

*Performance evaluation* To evaluate algorithmic performance we use the shifted geometric means, which provide a measure for relative differences. The *shifted geometric mean* of values  $v_1, \dots, v_N \geq 0$  with shift  $s \geq 0$  is defined as

$$\left( \prod_{i=1}^N (v_i + s) \right)^{1/N} - s.$$

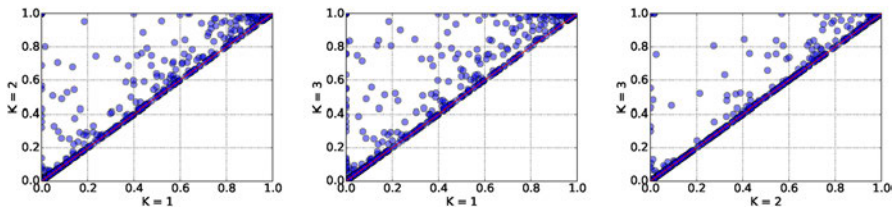
This measure avoids results being dominated by outliers with large absolute values, and also avoids an over-representation of differences among very small values. See also the discussion in [1,2,29]. As shift values we use 10 seconds for averaging over running time and 5% for averaging over gap closed values.

*Hardware and software* The experiments were performed on a cluster of 64bit Intel Xeon X5672 CPUs at 3.2 GHz with 12 MB cache and 48 GB main memory. In order to safeguard against a potential mutual slowdown of parallel processes, we ran only one job per node at a time. We used a development version of SCIP with CPLEX 12.8.0.0 as LP solver [31], the graph automorphism package bliss 0.73 [33] for detecting MILP symmetry, the algorithmic differentiation code CppAD 20180000.0 [12] and Ipopt 3.12.11 with Mumps 4.10.0 [44] as NLP solver [13,66].

## 6.2 Computational results

*ROOTGAP Experiment* From all instances of MINLPLib, we filter those for which SCIP's MILP relaxation proves optimality in the root node, no primal solution is known, or SCIP aborted due to numerical issues in the LP solver. This leaves 633 instances for the ROOTGAP experiment.

Figure 6 visualizes the achieved gap closed values via scatter plots. The plots show that for the majority of the instances we can close significantly more gap than the MILP relaxation. There are 173 instances for which  $K = 2$  closes at least 1% more gap than  $K = 1$ , and even more gap can be closed using  $K = 3$ . There are 21 instances for which  $K = 1$  could not close any gap, but  $K = 2$  could close some. On 11 additional instances  $K = 3$  could close gap, which was not possible with  $K = 2$ . Finally, comparing  $K = 2$  and  $K = 3$  shows that on 105 instances  $K = 3$  could close at least 1% more gap than  $K = 2$ . Interestingly, for most of these instances  $K = 2$  could already close at least 50% of the root gap.



**Fig. 6** Scatter plot comparing the root gap closed values of the ROOTGAP experiment comparing  $(K = 1, K = 2)$ ,  $(K = 1, K = 3)$ , and  $(K = 2, K = 3)$ . For example, each point  $(x, y) \in [0, 1]^2$  in the top left plot corresponds to an instance for which  $100x\%$  of the gap is closed by  $K = 1$  and  $100y\%$  closed by  $K = 2$

**Table 1** Aggregated results for the ROOTGAP experiment

Group	# instances	$K = 1$ (%)	$K = 2$ (%)	$K = 3$ (%)
ALL	633	18.4	21.4	23.4
$m \geq 10$	528	14.6	16.9	18.4
$m \geq 20$	391	10.7	12.3	13.5
$m \geq 50$	229	7.1	7.9	8.5
AFFECTED				
ALL	469	35.0	42.2	46.9
$m \geq 10$	370	30.1	36.0	40.1
$m \geq 20$	244	26.2	31.5	35.4
$m \geq 50$	115	23.9	28.0	30.8

A row  $m \geq x$  considers all instances that have at least  $x$  many nonlinear constraints. The second part of the table only considers instances where at least one setting closes at least 1% of the root gap

Aggregated results are reported in Table 1 and we refer to the supplementary material for detailed instance-wise results. First, we observe an average gap reduction of 18.4% for  $K = 1$ , 21.4% for  $K = 2$ , and 23.4% for  $K = 3$ , respectively. The same tendency is true when considering groups of instances that are defined by a bound on the minimum number of nonlinear constraints. For example, for the 391 instances with at least 20 nonlinear constraints after preprocessing,  $K = 2$  and  $K = 3$  close 1.6% and 2.8% more gap than  $K = 1$ , respectively. Table 1 also reports results when filtering out the 164 instances for which less than 1% gap was closed by Algorithm 2. We consider these instances *unaffected*. On the 469 affected instances we close on average up to 46.9% of the gap, and we see that  $K = 3$  closes 4.7% more gap than  $K = 2$  and 11.9% more than  $K = 1$ .

These results show that using surrogate relaxations has a significant impact on reducing the root gap. Additionally, using the generalized surrogate dual for  $K = 2$  and  $K = 3$  reduces significantly more gap than the classic surrogate dual.

*BENDERS Experiment* Table 2 reports aggregated results for the BENDERS experiment. We refer to the supplementary material for detailed instance-wise results.

First, we observe that the DEFAULT performs significantly better than PLAIN. Table 2 shows that on 100 of the 457 affected instances DEFAULT closes at least 1% more gap than PLAIN. Only on 36 instances PLAIN closes more gap, but over all



**Table 2** Aggregated results for the BENDERS experiment

Group	P	PLAIN			NOSTAB			NOSUPP			NOEARLY		
		M	L	rgc	M	L	rgc	M	L	rgc	M	L	rgc
ALL	457	100	36	86.9	40	31	98.3	41	27	98.4	94	40	88.2
$m \geq 10$	346	90	29	84.1	34	27	98.7	38	24	98.5	85	33	85.4
$m \geq 20$	222	72	10	77.2	25	17	98.5	32	20	98.2	65	12	79.7
$m \geq 50$	107	35	1	74.2	13	7	98.9	14	12	98.4	32	2	76.3

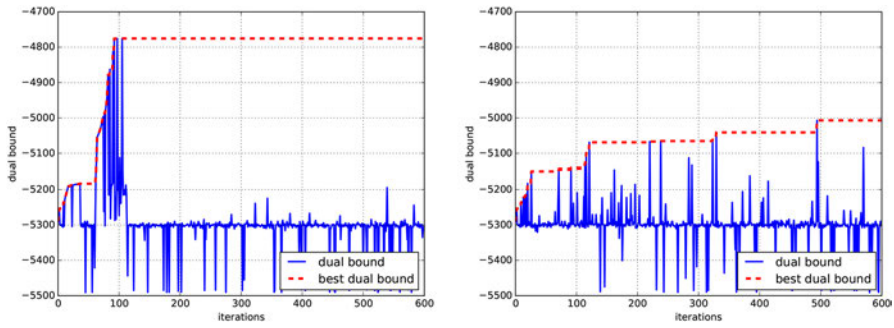
The column “M”/“L” reports the number of instances for which DEFAULT could close at least 1% more/less root gap than the settings of the corresponding column. Column “rgc” reports the average root gap closed relative to our default settings (in %). Instances for which no setting could close at least 1% of the root gap are filtered out

instances it closes on average 13.1% less gap than DEFAULT. On instances with a larger number of nonlinear constraints, DEFAULT performs even better: on the 107 instances with at least 50 nonlinear constraints, DEFAULT computes 35 times a better and only 1 time a worse dual bound than PLAIN. For these 107 instances, PLAIN closes 25.8% less gap than DEFAULT. Interestingly, there are instances for which PLAIN could not close any gap but DEFAULT could. There is no instance for which the opposite is true.

Next, we analyze which components of Algorithm 2 are responsible for the significantly better performance of DEFAULT compared to PLAIN. Table 2 shows that DEFAULT dominates NOSTAB, NOSUPP, and NOEARLY with respect to the average gap closed and the difference between the number of wins and the number of losses on each subset of the instances. The most important component is the early termination of the master problem. By disabling this feature, Algorithm 2 closes 11.8% less gap on all instances and even 23.7% on those which have at least 50 nonlinear constraints.

Regarding both stabilization techniques, Table 2 seems to suggest that they are not crucial. However, both techniques are important to exploit the  $\lambda$  space in a more structured way, which help us to find better aggregation vectors faster. To visualize this, we use the instance `genpooling_lee1` from Example 4. Figure 7 shows the achieved dual bounds in each iteration of Algorithm 2 for DEFAULT and NOSTAB. Both settings run with an iteration limit of 600. First, we observe that the achieved dual bound of  $-4775.26$  with DEFAULT is significantly better than the dual bound of  $-5006.95$  when using NOSTAB. The best dual bound is found after 97 iterations by DEFAULT and after 494 iterations with NOSTAB. The steepest improvement in the dual bounds for DEFAULT is seen between iterations 62 and 112, where the support was fixed and the trust region was enforced. After iteration 112 the algorithm removed both stabilizers and no further dual bound improvement could be found. In the iterations where no stabilization is used by DEFAULT, we do not observe any pattern indicating which of the two settings finds a better dual bound—the behavior seems rather random. This randomness, and the large time limit used, might explain the similar results for DEFAULT and NOSTAB of Table 2.

*DUALBOUND Experiment* For this experiment, we include all instances which could not be solved by SCIP with default settings within three hours, have a final gap of at least ten percent, terminate without an error, and contain at least four nonlinear



**Fig. 7** Comparison of the achieved dual bounds and the obtained aggregation vectors during Algorithm 2 for DEFAULT (left) and NOSTAB (right) on `genpooling_lee1` using  $K = 3$ . The red dashed curve shows the best found dual bound so far, whereas the blue curve shows the computed dual bound in every iteration (color figure online)

**Table 3** Dual bounds computed by SCIP, Algorithm 2, and the best known dual bounds reported in the MINLPLib for all `polygon*` and four `facloc*` instances

Instance	best primal	DB (MINLPLib)	DB (SCIP)	DUALBOUND
<code>polygon25</code>	-0.78	-5.80	-4.24	-3.94
<code>polygon50</code>	-0.78	-15.27	-10.78	-8.72
<code>polygon75</code>	-0.78	-24.87	-16.82	-13.55
<code>polygon100</code>	-0.78	-34.00	-24.37	-19.03
<code>facloc1_3_95</code>	12.30	4.46	5.50	5.70
<code>facloc1_4_80</code>	7.88	0.16	0.09	0.41
<code>facloc1_4_90</code>	10.46	0.48	0.49	1.18
<code>facloc1_4_95</code>	11.18	0.79	1.40	2.40

constraints. This leaves in total 209 instances for the DUALBOUND experiment. The supplementary material reports detailed results on the subset of instances for which the Algorithm 2 was able to improve on the bound obtained by SCIP with default settings, which was the case for 53 of the 209 instances. On these, the average gap of 284.3% for SCIP with default settings could be reduced to an average gap of 142.8%.

Two notable subsets of instances are the challenging `polygon*` instances and four of the `facloc*` instances. In these, Algorithm 2 finds better bounds than the reported best known dual bounds from the MINLPLib, as shown in Table 3.

### 7 Surrogate duality during the tree search

While we obtain strong dual bounds with Algorithm 2, complex instances still require branching in order to solve them to provable optimality. In this section, we show a promising technique to incorporate Algorithm 2 into spatial branch and bound. We discuss this technique in two instances of MINLPLib, while a full implementation is subject of future work.

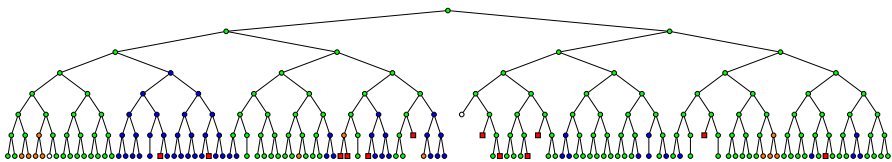
**Algorithm 3:** Surrogate approximation

---

**Input:** node  $v$ , parent node  $p$ , parent aggregation candidates  $C_p \subseteq \Lambda$   
**Output:**  $D \in \mathbb{R}$  valid dual bound for  $v$ , aggregation candidates  $C_v \subseteq C_p$

- 1: initialize  $D \leftarrow S_v^K(0)$
- 2: initialize  $C_v \leftarrow C_p$
- 3: **for**  $\lambda \in C_v$  **do**
- 4:   **if**  $D < S_v^K(\lambda)$  **then**
- 5:     **return**  $(S_v^K(\lambda), C_v)$
- 6:   **else**
- 7:      $C_v \leftarrow C_v \setminus \{\lambda\}$
- 8:   **end if**
- 9: **end for**
- 10: **return**  $(D, \emptyset)$

---



**Fig. 8** Visualization of Algorithm 3 for the instance `himmel116` of the MINLPLib. The size of the aggregation pool has been limited to three and we used bread-first-search as the node selection strategy. The colors determine  $|C_v|$  at each node  $v$ : green for three, blue for two, orange for one, white for zero aggregations. Red square nodes could be pruned by Algorithm 3, i.e., the proven dual bound exceeds the value of an incumbent solution (color figure online)

Since Algorithm 2 is too costly to be used in every node of a branch-and-bound tree, our technique focuses on extracting information of a *single* execution of Algorithm 2 in the root node, and reuses this information throughout the tree.

Let  $\Lambda := \{\lambda_1, \dots, \lambda_L\} \subseteq \mathbb{R}^{Km}$  be the set of aggregation vectors that have been computed during Algorithm 2 in the root node, and that imply a tighter dual bound than the MILP relaxation. Instead of using Algorithm 2 in a node  $v$ , we select an aggregation vector  $\lambda$  from  $\Lambda$  and solve  $S_v^K(\lambda)$ , which is equal to  $S^K(\lambda)$  except that the global linear relaxation is replaced with a locally valid linear relaxation for  $v$ . If  $S_v^K(\lambda)$  results in a better dual bound than the local MILP relaxation, i.e.,  $S_v^K(0)$ , then we skip the remaining aggregation vectors in  $\Lambda$  and continue with the tree search. If the dual bound does not improve, then we discard  $\lambda$  in the sub-tree with root  $v$ .

The intuition behind discarding aggregations as we search down the tree is twofold. First, since the aggregations are computed in the root node, their ability to provide good dual bounds is expected to deteriorate with the increasing depth of an explored node. Second, we would like to alleviate the computational load of checking for too many aggregations as the branch-and-bound tree-size increases. The idea is stated in Algorithm 3; the algorithm assumes  $C_r = \Lambda$  for the root node  $r$ . In Fig. 8 we illustrate Algorithm 3 for the instance `himmel116`.

In principle, Algorithm 3 can lead to stronger dual bounds in local nodes of the branch-and-bound tree, which could result in a smaller tree. However, for the challenging instances of the DUALBOUND experiment we observed that solving  $S_v^K(\lambda)$  is too costly and almost always runs into the time limit. In these cases, Algorithm 3

was not able to improve the dual bound. An exception to this behavior is instance `multiplants_mtglc2`, which contains 28 nonlinear constraints, 193 continuous variables, and 104 integer variables. SCIP with default settings proves a dual bound of  $-4096.04$ . This bound is improved by Algorithm 2 to  $-3161.13$ , and Algorithm 3 can further improve it to  $-2935.58$ . This last bound improvement happens in the early stages of the tree search; there were only seven nodes for which a better local dual bound was found, and the aggregation candidates were quickly filtered out. After this, SCIP was not able to improve the dual bound any further. In total, SCIP processed 491860 nodes.

## 8 Summary and future work

In this article, we studied theoretical and computational aspects of surrogate relaxations for MINLPs. We developed the first algorithm to solve a generalization of the surrogate dual problem that allows multiple aggregations of nonlinear constraints. We also proposed computational enhancements and discussed a first idea on how to exploit surrogate duality in a spatial branch-and-bound solver.

Our extensive computational study on the heterogeneous set of publicly available instances of the MINLPLib shows that surrogate duality can lead to significantly better dual bounds than using SCIP with default settings. Additionally, our experiments show that the presented computational enhancements are important to obtain good dual bounds for problems with a large number of nonlinear constraints. Finally, our tree experiments show that using the result of Algorithm 2 during the tree search can lead to significantly better dual bounds than solving MINLPs with standard spatial branch and bound, although in this latter direction the research is still preliminary.

There are two open questions related to generalized surrogate duality we would like to highlight. First, consider the case that each constraint of (1) is quadratic, i.e.,  $g_i(x) = x^T Q_i x + q_i^T x + b_i$  for each  $i \in \mathcal{M}$ . Adding the constraints  $\sum_{i \in \mathcal{M}} \lambda_i^k Q_i \succeq 0$  for all  $k \in \{1, \dots, K\}$  to the master problem (12) enforces that each sub-problem is a convex mixed-integer quadratically-constrained program. This increases the complexity of the master problem but, at the same time, reduces the complexity of the sub-problems. It would be interesting to understand this trade-off computationally. Second, it remains an open question how a pure surrogate-based spatial branch-and-bound approach could perform in practice. Warm-start strategies based on the generated points  $\mathcal{X}$  of a parent node and branching rules based on surrogates relaxations are some examples of tools that could be developed in order to successfully incorporate surrogate duality into branch and bound.

**Acknowledgements** We gratefully acknowledge support from the Research Campus MODAL (BMBF Grants 05M14ZAM, 05M20ZBM) and the Institute for Data Valorization (IVADO) through an IVADO Postdoctoral Fellowship. We would also like to thank the two anonymous reviewers for their valuable feedback and suggestions which greatly helped improving this article.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

<sup>2</sup> This is a maximization problem. To be consistent with our presentation throughout the paper we treat it as a minimization problem by presenting the additive inverses of the objective value bounds.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Achterberg, T.: Constraint integer programming. Ph.D. thesis, Technische Universität Berlin (2007). <https://doi.org/10.14279/depositonce-1634>, <http://nbn-resolving.de/urn:nbn:de:kobv:83-opus-16117>
2. Achterberg, T., Wunderling, R.: Mixed integer programming: analyzing 12 years of progress. In: Jünger, M., Reinelt, G. (eds.) Facets of combinatorial optimization, pp. 449–481. Springer, Berlin (2013). [https://doi.org/10.1007/978-3-642-38189-8\\_18](https://doi.org/10.1007/978-3-642-38189-8_18)
3. Alidaee, B.: Zero duality gap in surrogate constraint optimization: a concise review of models. *Eur. J. Oper. Res.* **232**(2), 241–248 (2014). <https://doi.org/10.1016/j.ejor.2013.04.023>
4. Amor, H.M.B., Desrosiers, J., Frangioni, A.: On the choice of explicit stabilizing terms in column generation. *Discrete Appl. Math.* **157**(6), 1167–1184 (2009). <https://doi.org/10.1016/j.dam.2008.06.021>
5. Balas, E.: Discrete programming by the filter method. *Oper. Res.* **15**(5), 915–957 (1967). <https://doi.org/10.1287/opre.15.5.915>
6. Balas, E.: Disjunctive programming: properties of the convex hull of feasible points. *Discrete Appl. Math.* **89**(1–3), 3–44 (1998). [https://doi.org/10.1016/s0166-218X\(98\)00136-X](https://doi.org/10.1016/s0166-218X(98)00136-X)
7. Banerjee, K.: Generalized Lagrange multipliers in dynamic programming. Ph.D. thesis, University of California, Berkeley (1971)
8. Belotti, P., Caferri, S., Lee, J., Liberti, L.: On feasibility based bounds tightening. Technical Report 3325, Optimization Online (2012). [http://www.optimization-online.org/DB\\_HTML/2012/01/3325.html](http://www.optimization-online.org/DB_HTML/2012/01/3325.html)
9. Benichou, M., Gauthier, J.M., Girodet, P., Hentges, G., Ribiere, G., Vincent, O.: Experiments in mixed-integer linear programming. *Math. Program.* **1**(1), 76–94 (1971). <https://doi.org/10.1007/BF01584074>
10. Bonami, P., Lodi, A., Tramontani, A., Wiese, S.: On mathematical programming with indicator constraints. *Math. Program.* **151**(1), 191–223 (2015). <https://doi.org/10.1007/s10107-015-0891-4>
11. Burlacu, R., Geißler, B., Schewe, L.: Solving mixed-integer nonlinear programmes using adaptively refined mixed-integer linear programmes. *Optim. Methods Softw.* (2019). <https://doi.org/10.1080/10556788.2018.1556661>
12. COIN-OR: CppAD, a package for differentiation of C++ algorithms. <http://www.coin-or.org/CppAD>
13. COIN-OR: Ipopt, interior point optimizer. <http://www.coin-or.org/Ipopt>
14. Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust Region Methods. Society for Industrial and Applied Mathematics, University City (2000). <https://doi.org/10.1137/1.9780898719857>
15. Djerdjour, M., Mathur, K., Salkin, H.M.: A surrogate relaxation based algorithm for a general quadratic multi-dimensional knapsack problem. *Oper. Res. Lett.* **7**(5), 253–258 (1988). [https://doi.org/10.1016/0167-6377\(88\)90041-7](https://doi.org/10.1016/0167-6377(88)90041-7)
16. du Merle, O., Villeneuve, D., Desrosiers, J., Hansen, P.: Stabilized column generation. *Discrete Math.* **194**(1–3), 229–237 (1999). [https://doi.org/10.1016/s0012-365X\(98\)00213-1](https://doi.org/10.1016/s0012-365X(98)00213-1)
17. Dyer, M.E.: Calculating surrogate constraints. *Math. Program.* **19**(1), 255–278 (1980). <https://doi.org/10.1007/BF01581647>
18. Fisher, M., Lageweg, B., Lenstra, J., Kan, A.: Surrogate duality relaxation for job shop scheduling. *Discrete Appl. Math.* **5**(1), 65–75 (1983). [https://doi.org/10.1016/0166-218\(83\)990016-1](https://doi.org/10.1016/0166-218(83)990016-1)
19. Gavish, B., Pirkul, H.: Efficient algorithms for solving multiconstraint zero-one knapsack problems to optimality. *Math. Program.* **31**(1), 78–105 (1985). <https://doi.org/10.1007/BF02591863>

20. Geoffrion, A.M.: Implicit enumeration using an imbedded linear program. Technical report (1967). <https://doi.org/10.21236/ad0655444>
21. Glover, F.: A multiphase-dual algorithm for the zero-one integer programming problem. *Oper. Res.* **13**(6), 879–919 (1965). <https://doi.org/10.1287/opre.13.6.879>
22. Glover, F.: Surrogate constraints. *Oper. Res.* **16**(4), 741–749 (1968). <https://doi.org/10.1287/opre.16.4.741>
23. Glover, F.: Surrogate constraint duality in mathematical programming. *Oper. Res.* **23**(3), 434–451 (1975). <https://doi.org/10.1287/opre.23.3.434>
24. Glover, F.: Heuristics for integer programming using surrogate constraints. *Decis. Sci.* **8**(1), 156–166 (1977). <https://doi.org/10.1111/j.1540-5915.1977.tb01074.x>
25. Glover, F.: Tutorial on surrogate constraint approaches for optimization in graphs. *J. Heuristics* **9**(3), 175–227 (2003). <https://doi.org/10.1023/A:1023721723676>
26. Gomory, R.E.: An algorithm for the mixed integer problem. Technical Report P-1885, The RAND Corporation (1960)
27. Greenberg, H.J., Pierskalla, W.P.: Surrogate mathematical programming. *Oper. Res.* **18**(5), 924–939 (1970). <https://doi.org/10.1287/opre.18.5.924>
28. Grossmann, I.E., Sahinidis, N.V.: Special issue on mixed integer programming and its application to engineering, part I. *Optim. Eng.* **3**(4) (2002)
29. Hendel, G.: Empirical analysis of solving phases in mixed integer programming. Master's thesis, Technische Universität Berlin (2014). <http://nbn-resolving.de/urn:nbn:de:0297-zib-54270>
30. Horst, R., Tuy, H.: *Global Optimization*. Springer, Berlin (1996). <https://doi.org/10.1007/978-3-662-03199-5>
31. ILOG, I.: ILOG CPLEX: High-performance software for mathematical programming and optimization. <http://www.ilog.com/products/cplex/>
32. Kelley, J.E., Jr.: The cutting-plane method for solving convex programs. *J. Soc. Ind. Appl. Math.* **8**(4), 703–712 (1960). <https://doi.org/10.1137/0108053>
33. Junttila, T., Kaski, P.: bliss: A tool for computing automorphism groups and canonical labelings of graphs. <http://www.tcs.hut.fi/Software/bliss/> (2012)
34. Karwan, M.H.: Surrogate constraint duality and extensions in integer programming. Ph.D. thesis, Georgia Institute of Technology (1976)
35. Karwan, M.H., Rardin, R.L.: Some relationships between Lagrangian and surrogate duality in integer programming. *Math. Program.* **17**(1), 320–334 (1979). <https://doi.org/10.1007/BF01588253>
36. Karwan, M.H., Rardin, R.L.: Searchability of the composite and multiple surrogate dual functions. *Oper. Res.* **28**(5), 1251–1257 (1980). <https://doi.org/10.1287/opre.28.5.1251>
37. Karwan, M.H., Rardin, R.L.: Surrogate duality in a branch-and-bound procedure. *Naval Res. Logist. Q.* **28**(1), 93–101 (1981). <https://doi.org/10.1002/nav.3800280107>
38. Karwan, M.H., Rardin, R.L.: Surrogate dual multiplier search procedures in integer programming. *Oper. Res.* **32**(1), 52–69 (1984). <https://doi.org/10.1287/opre.32.1.52>
39. Kim, S.L., Kim, S.: Exact algorithm for the surrogate dual of an integer programming problem: sub-gradient method approach. *J. Optim. Theory Appl.* **96**(2), 363–375 (1998). <https://doi.org/10.1023/A:1022622231801>
40. Margot, F.: Symmetry in integer linear programming. In: Jünger, M., et al. (eds.) *50 Years of Integer Programming 1958–2008*, pp. 647–686. Springer, Berlin (2009). [https://doi.org/10.1007/978-3-540-68279-0\\_17](https://doi.org/10.1007/978-3-540-68279-0_17)
41. McCormick, G.P.: Computability of global solutions to factorable nonconvex programs: part i—convex underestimating problems. *Math. Program.* **10**(1), 147–175 (1976). <https://doi.org/10.1007/BF01580665>
42. MINLP library. <http://www.minplib.org/>
43. Misener, R., Floudas, C.A.: ANTIGONE: Algorithms for coNTinuous/integer global optimization of nonlinear equations. *J. Global Optim.* **59**(2–3), 503–526 (2014). <https://doi.org/10.1007/s10898-014-0166-2>
44. MUMPS: Multifrontal massively parallel sparse direct solver. <http://mumps.enseeiht.fr>
45. Nakagawa, Y.: An improved surrogate constraints method for separable nonlinear integer programming. *J. Oper. Res. Soc. Jpn.* **46**(2), 145–163 (2003). <https://doi.org/10.15807/jorsj.46.145>
46. Narciso, M.G., Lorena, L.A.N.: Lagrangean/surrogate relaxation for generalized assignment problems. *Eur. J. Oper. Res.* **114**(1), 165–177 (1999). [https://doi.org/10.1016/s0377-2217\(98\)00038-1](https://doi.org/10.1016/s0377-2217(98)00038-1)

47. Neame, P.J.: Nonsmooth dual methods in integer programming. Ph.D. thesis, University of Melbourne, Department of Mathematics and Statistics (2000)
48. Nemhauser, G.L., Wolsey, L.A.: A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Math. Program.* **46**(1–3), 379–390 (1990). <https://doi.org/10.1007/BF01585752>
49. Penot, J.P., Volle, M.: Surrogate programming and multipliers in quasi-convex programming. *SIAM J. Control Optim.* **42**(6), 1994–2003 (2004). <https://doi.org/10.1137/s0363012902327819>
50. Quesada, I., Grossmann, I.E.: Global optimization algorithm for heat exchanger networks. *Ind. Eng. Chem. Res.* **32**(3), 487–499 (1993). <https://doi.org/10.1021/ie00015a012>
51. Quesada, I., Grossmann, I.E.: A global optimization algorithm for linear fractional and bilinear programs. *J. Global Optim.* **6**(1), 39–76 (1995). <https://doi.org/10.1007/BF01106605>
52. Ryoo, H., Sahinidis, N.: Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Comput. Chem. Eng.* **19**(5), 551–566 (1995). [https://doi.org/10.1016/0098-1354\(94\)00097-2](https://doi.org/10.1016/0098-1354(94)00097-2)
53. Sarin, S., Karwan, M.H., Rardin, R.L.: A new surrogate dual multiplier search procedure. *Naval Res. Logist.* **34**(3), 431–450 (1987). [https://doi.org/10.1002/1520-6750\(198706\)34:3<365::AID-NAV3220340305>3.0.CO;2-P](https://doi.org/10.1002/1520-6750(198706)34:3<365::AID-NAV3220340305>3.0.CO;2-P)
54. Sarin, S., Karwan, M.H., Rardin, R.L.: Surrogate duality in a branch-and-bound procedure for integer programming. *Eur. J. Oper. Res.* **33**(3), 326–333 (1988). [https://doi.org/10.1016/0377-2217\(88\)90176-2](https://doi.org/10.1016/0377-2217(88)90176-2)
55. SCIP—solving constraint integer programs. <http://scip.zib.de>
56. Serali, H.D., Adams, W.P.: *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems*. Springer, New York (1999). <https://doi.org/10.1007/978-1-4757-4388-3>
57. Serali, H.D., Fraticelli, B.M.P.: Enhancing RLT relaxations via a new class of semidefinite cuts. *J. Global Optim.* **22**(1/4), 233–261 (2002). <https://doi.org/10.1023/A:1013819515732>
58. Suzuki, S., Kuroiwa, D.: Necessary and sufficient constraint qualification for surrogate duality. *J. Optim. Theory Appl.* **152**(2), 366–377 (2011). <https://doi.org/10.1007/s10957-011-9893-4>
59. Templeman, A.B., Xingsi, L.: A maximum entropy approach to constrained non-linear programming. *Eng. Optim.* **12**(3), 191–205 (1987). <https://doi.org/10.1080/03052158708941094>
60. van Ackooij, W., Frangioni, A., de Oliveira, W.: Inexact stabilized benders’ decomposition approaches with application to chance-constrained problems with finite support. *Comput. Optim. Appl.* **65**(3), 637–669 (2016). <https://doi.org/10.1007/10589-016-9851-z>
61. Vielma, J.P.: Embedding formulations and complexity for unions of polyhedra. *Manag. Sci.* **64**(10), 4721–4734 (2018). <https://doi.org/10.1287/mnsc.2017.2856>
62. Vielma, J.P.: Small and strong formulations for unions of convex sets from the Cayley embedding. *Math. Program.* **177**(1–2), 21–53 (2018). <https://doi.org/10.1007/s10107-018-1258-4>
63. Vigerske, S.: Decomposition in multistage stochastic programming and a constraint integer programming approach to mixed-integer nonlinear programming. Ph.D. thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät II (2013). <http://nbn-resolving.de/urn:nbn:de:kobv:11-100208240>
64. Vigerske, S., Gleixner, A.: SCIP: global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. *Optim. Methods Softw.* **33**(3), 563–593 (2017). <https://doi.org/10.1080/10556788.2017.1335312>
65. Wentges, P.: Weighted Dantzig–Wolfe decomposition for linear mixed-integer programming. *Int. Trans. Oper. Res.* **4**(2), 151–162 (1997). [https://doi.org/10.1016/s0969-6016\(97\)00001-4](https://doi.org/10.1016/s0969-6016(97)00001-4)
66. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.* **106**(1), 25–57 (2005). <https://doi.org/10.1007/s10107-004-0559-y>
67. Xingsi, L.: An aggregate constraint method for non-linear programming. *J. Oper. Res. Soc.* **42**(11), 1003–1010 (1991). <https://doi.org/10.1057/jors.1991.190>
68. Zhou, K., Kılınç, M.R., Chen, X., Sahinidis, N.V.: An efficient strategy for the activation of MIP relaxations in a multicore global MINLP solver. *J. Global Optim.* **70**(3), 497–516 (2017). <https://doi.org/10.1007/s10898-017-0559-0>