Mathematical Programming Computation (2023) 15:703–731
https://doi.org/10.1007/s12532-023-00246-4

FULL LENGTH PAPER

A computational study of perspective cuts

Ksenia Bestuzheva[1] · **Ambros Gleixner**[2] · **Stefan Vigerske**[3]

Received: 4 March 2021 / Accepted: 4 May 2023 / Published online: 21 August 2023

Abstract
The benefits of cutting planes based on the perspective function are well known for many specific classes of mixed-integer nonlinear programs with on/off structures. However, we are not aware of any empirical studies that evaluate their applicability and computational impact over large, heterogeneous test sets in general-purpose solvers. This paper provides a detailed computational study of perspective cuts within a linear programming based branch-and-cut solver for general mixed-integer nonlinear programs. Within this study, we extend the applicability of perspective cuts from convex to nonconvex nonlinearities. This generalization is achieved by applying a perspective strengthening to valid linear inequalities which separate solutions of linear relaxations. The resulting method can be applied to any constraint where all variables appearing in nonlinear terms are semi-continuous and depend on at least one common indicator variable. Our computational experiments show that adding perspective cuts for convex constraints yields a consistent improvement of performance, and adding perspective cuts for nonconvex constraints reduces branch-and-bound tree sizes and strengthens the root node relaxation, but has no significant impact on the overall mean time.

Keywords Perspective cuts · Mixed-integer nonlinear programming · Nonconvex optimization · Computational study

Mathematics Subject Classification 90-04 · 90-05 · 90-08 · 90C11 · 90C26 · 90C30 · 90C57

✉ Ksenia Bestuzheva
bestuzheva@zib.de

Ambros Gleixner
gleixner@htw-berlin.de

Stefan Vigerske
svigerske@gams.com

1 Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany

2 Zuse Institute Berlin and HTW Berlin, Berlin, Germany

3 GAMS Software GmbH, c/o Zuse Institute Berlin, Berlin, Germany

1 Introduction

Consider a mixed-integer nonlinear program (MINLP) with semi-continuous variables:

$$\min \ \langle \boldsymbol{c}, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle \tag{1a}$$

$$\text{s.t. } g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \leq 0, \tag{1b}$$

$$(\underline{y}_j - y_j^0) z_k \leq y_j - y_j^0 \leq (\overline{y}_j - y_j^0) z_k, \ \forall j \in \mathcal{S}_k, \ \forall k \in \mathcal{I}, \tag{1c}$$

$$\boldsymbol{x} \in \mathbb{R}^n, \ \boldsymbol{y} \in \mathbb{R}^p, \ \boldsymbol{z} \in \{0, 1\}^q. \tag{1d}$$

Here, $\langle \boldsymbol{c}, (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \rangle$ is the linear objective function given by a scalar product of a constant vector $\boldsymbol{c} \in \mathbb{R}^{n+p+q}$ and the vectors of continuous variables $\boldsymbol{x}$, semi-continuous variables $\boldsymbol{y}$ and binary variables $\boldsymbol{z}$. Constraints (1b) are given by inequalities $g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \leq 0$, where $g : \mathbb{R}^n \times \mathbb{R}^p \times [0, 1]^q \to \mathbb{R}^m$ is a vector function and some of its elements g_i are nonlinear.

The set $\mathcal{S}_k \subseteq \{1, \ldots, p\}$ shall contain the indices of semi-continuous variables controlled by the indicator variable z_k, and $\mathcal{I} \subseteq \{1, \ldots, q\}$ is the set of indices of all indicator variables. Constraints (1c) ensure that for each $j \in \mathcal{S}_k$, the value of y_j belongs to the domain $[\underline{y}_j, \overline{y}_j]$ when the indicator variable z_k is equal to 1 and has a fixed value y_j^0 when z_k is equal to 0. Semi-continuous variables are typically used to model "on" and "off" states of a process and can be found in such problems as optimal line switching in electrical networks [12], blending [39] and production planning [5], to name but a few.

In order to simplify the notation, in the rest of the paper the subscript k will be omitted and we will be referring to a vector of semi-continuous variables $\boldsymbol{y} \in \mathbb{R}^p$ controlled by the indicator variable $z \in \{0, 1\}$. The semi-continuity relation is then defined by the inequality $(\underline{\boldsymbol{y}} - \boldsymbol{y}^0)z \leq \boldsymbol{y} - \boldsymbol{y}^0 \leq (\overline{\boldsymbol{y}} - \boldsymbol{y}^0)z$.

In the theoretical part of the paper, we consider constraints of the form

$$g(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{y}) - x_\ell \leq 0 \tag{2}$$

for some $\ell \in \{1, \ldots, n\}$ and $f : \mathbb{R}^p \to \mathbb{R}$. When $z = 0$, function f is reduced to a fixed value $f(\boldsymbol{y}^0)$. A common example of such constraints are on/off constraints which become redundant when the corresponding indicator variable is set to 0.

The continuous variable x_ℓ represents the linear non-semi-continuous part of the expression on the left-hand side. This is without loss of generality, since the same arguments can be directly adapted for a more general linear part. For the experiments, arbitrary linear parts are allowed, with the condition that any non-semi-continuous variables must only appear linearly in the entire expression.

Many state-of-the-art algorithms for the solution of MINLP (1) make use of nonlinear and linear programming relaxations where the condition $z \in \{0, 1\}$ is replaced with $z \in [0, 1]$. However, for a constraint of the form (2), simply dropping the integrality condition generally does not produce the tightest possible continuous relaxation. The reason for this is that the dependence of $\boldsymbol{y}$'s bounds on the indicator variable z is not

exploited by a straightforward continuous relaxation. Consequently, the same applies for the linearization of Constraint (2) via gradient cuts [29], that is, inequalities

$$f(\hat{\boldsymbol{y}}) + \left\langle \nabla f(\hat{\boldsymbol{y}}), \boldsymbol{y} - \hat{\boldsymbol{y}} \right\rangle \leq x_\ell, \tag{3}$$

where $\hat{\boldsymbol{y}}$ is the point at which f is linearized.

The strongest continuous relaxation of the set described by Constraint (2), given that $\boldsymbol{y}$ is semi-continuous, can be achieved by applying the perspective reformulation [15]. Linearizing this reformulation provides valid linear inequalities known as perspective cuts.

In this paper we present a computational study of perspective cuts within SCIP [7], a general-purpose solver that implements an LP-based branch-and-cut algorithm to solve mixed-integer nonlinear programs to global optimality. Section 2 provides the theoretical background for this study and a review of applications that can be found in existing literature. In Sect. 3, we describe our approach to creating perspective cuts and show that for convex instances, it is equivalent to linearizing the perspective formulation via gradient cuts. Section 4 gives an outline of our implementation of perspective cuts in SCIP, which includes detection of suitable structures and separation and strengthening of perspective cuts. Finally, in Sect. 5 the results of computational experiments on instances from MINLPLib[1] [9] are presented.

2 Perspective formulations for convex nonlinearities

This section gives a review of the existing literature on theoretical and computational results related to perspective formulations for convex nonlinearities.

2.1 Theoretical background

A key concept that is widely used in constructing convex hulls of unions of convex sets is that of the perspective function:

Definition 1 [32] For a given convex function $f : \mathbb{R}^p \to \mathbb{R}$, its perspective function $\tilde{f} : \mathbb{R}^{p+1} \to (\mathbb{R} \cup \{+\infty\})$ is defined as:

$$\tilde{f}(\boldsymbol{y}, z) = \begin{cases} z f(\boldsymbol{y}/z), & \text{if } z > 0, \\ +\infty, & \text{otherwise,} \end{cases}$$

where $\boldsymbol{y} \in \mathbb{R}^p$, $z \in \mathbb{R}$.

2.1.1 Convex hulls of unions of convex sets

The line of research that is concerned with formulating convex hulls of unions of convex sets began with the works of Balas [3, 4] on unions of polyhedral sets. These

[1] https://www.minlplib.org

works introduced the disjunctive programming approach to represent the feasible set of a mixed-integer program as a union of sets.

Considering the case of nonlinear sets, Ceria and Soares [10] studied convex hull formulations for unions of convex sets and their applications to disjunctive programming. Stubbs and Mehrotra [35] described the convex hull of the feasible set of a convex 0–1 program in an extended space and developed a procedure for generating cutting planes. Grossmann and Lee [24] extended the convex hull results to generalized disjunctive programs (GDPs). Similarly to disjunctive programming, feasible sets of GDPs are given as unions of convex sets, but more general logical relations are also allowed.

Khajavirad and Sahinidis [30] employed a convex hull formulation in line with the aforementioned works in order to construct a convex envelope of a nonconvex lower semi-continuous function, thus extending the applicability of disjunctive programming results to a wider class of problems.

2.1.2 Formulations in the original space

For the case where all sets in the disjunction lie in orthogonal subspaces, Tawarmalani et al. [36] proposed a formulation of the convex hull of a disjunction of a finite number of orthogonal sets. This formulation does not require the introduction of additional variables. Further, this technique is applied for constructing relaxations of nonconvex constraints that are not naturally disjunctive.

Hijazi et al. [27, 28] formulated the convex hull in the original space for the case of two sets corresponding to the two possible values of a binary indicator variable z. When $z = 0$, the values of $\boldsymbol{y}$ are restricted to an interval $[\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0]$. The function f is required to be isotone, that is, monotone in any given coordinate.

Let F denote the disjunctive set obtained as the union of two sets F^0 and F^1 corresponding to values 0 and 1 of z, respectively:

$$F^0 = \{(\boldsymbol{y}, z) \mid \boldsymbol{y} \in [\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0],\ z = 0\}, \tag{4}$$

$$F^1 = \{(\boldsymbol{y}, z) \mid f(\boldsymbol{y}) \leq 0,\ \boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}],\ z = 1\}. \tag{5}$$

The work by Hijazi et al. [27, 28] provides the formulation of the convex hull of F in the space of original problem variables using an exponential number of constraints. To avoid an exponentially large formulation, the authors employed a convex relaxation given by a subset of constraints defining the convex hull. They showed that this relaxed formulation provides strong dual bounds which yield considerable improvements in performance for the delay-constrained routing problem in telecommunications. Bestuzheva et al. [8] extended this result to the case of univariate non-isotone functions.

2.1.3 Formulations for a union of a convex set and a singleton set

Consider a special case where the set corresponding to the 0 value of the indicator variable is a singleton set, and let F^1 be now defined through an epigraph, that is,

$F = F^0 \cup F^1$, where

$$F^0 = \{(x_\ell, \boldsymbol{y}, z) \mid f(\boldsymbol{y}^0) \leq x_\ell,\ \boldsymbol{y} = \boldsymbol{y}^0,\ z = 0\}, \tag{6}$$
$$F^1 = \{(x_\ell, \boldsymbol{y}, z) \mid f(\boldsymbol{y}) \leq x_\ell,\ \boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}],\ z = 1\}. \tag{7}$$

Note that F is the set given by Constraint (2) together with semi-continuity conditions on $\boldsymbol{y}$. For this case, original space convex hull formulations are known which do not require an exponential number of constraints, unlike the more general formulations [27, 28].

In particular, Frangioni and Gentile [15] proposed a compact reformulation in the original space. Considering a semi-continuous vector $\boldsymbol{y}$, an indicator variable z, and a convex function f depending only on $\boldsymbol{y}$ such that $\boldsymbol{y}^0 = (0, \ldots, 0)$ and $f(\boldsymbol{0}) = 0$, they capture the disjunctive structure by defining a new nonconvex function f^d:

$$f^d(\boldsymbol{y}, z) = \begin{cases} f(\boldsymbol{y}) & \text{if } z = 1,\ y \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}], \\ 0 & \text{if } \boldsymbol{y} = z = 0, \\ +\infty & \text{otherwise.} \end{cases}$$

The function f^d is directly related to the set F: the latter can be described as the set of all points $(x_\ell, \boldsymbol{y}, z)$ such that $f^d(\boldsymbol{y}, z)$ is finite and $f^d(\boldsymbol{y}, z) \leq x_\ell$. Frangioni and Gentile described the convex envelope of f^d:

$$\overline{co} f^d(\boldsymbol{y}, z) = \begin{cases} \tilde{f}(\boldsymbol{y}, z) & \text{if } z \in (0, 1], \\ 0 & \text{if } z = 0, \\ +\infty & \text{otherwise.} \end{cases}$$

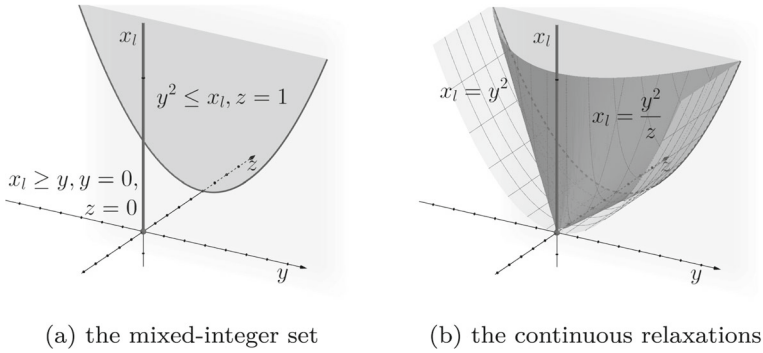
In a related work, Günlük and Linderoth [25] showed that, under similar assumptions, the convex hull of F is given by

$$\text{conv}(F) = \{(x_\ell, \boldsymbol{y}, z) \mid \overline{co} f^d(\boldsymbol{y}, z) \leq x_\ell,\ \underline{\boldsymbol{y}} z \leq \boldsymbol{y} \leq \overline{\boldsymbol{y}} z,\ z \in [0, 1]\}. \tag{8}$$

Therefore, replacing f with $\overline{co} f^d$ in Constraint (2) results in a reformulation with the tightest possible continuous relaxation: the *perspective reformulation*. This is a valid reformulation since $f = \overline{co} f^d$ for $z \in \{0, 1\}$.

Figure 1 shows an example of a disjunctive set and compares its continuous relaxations. The disjunctive set (shown in Fig. 1a) consists of the ray $\{(x_\ell, y, z) \mid x_\ell \geq 0,\ y = 0,\ z = 0\}$ and the convex set $\{(x_\ell, y, z) \mid x_\ell \geq f(y),\ z = 1\}$, where $f(y) = y^2$. The convex hull, shown in Fig. 1b, is the closure of the set of all points above the dark gray surface defined by $x_\ell = y^2/z, z \in (0, 1]$. This equation is obtained by applying the perspective operator to f: $\tilde{f}(y, z) = zf(y/z) = y^2/z$. For comparison, the boundary of the straightforward continuous relaxation given by $x_\ell = y^2$, $z \in [0, 1]$, is shown in Fig. 1b in light gray color.

Due to the division by z in the perspective function, the perspective reformulation (8) is non-differentiable at $z = 0$. This may lead to numerical difficulties in regions where



(a) the mixed-integer set (b) the continuous relaxations

Fig. 1 Example of a disjunctive set and its convex hull (created with Geogebra https://www.geogebra.org/)

the indicator variable is close to zero. In some special cases, formulation (8) can be written as a second-order cone (SOC) constraint [1, 17, 25, 37]. In particular, this is possible when Constraint (2) itself is SOC-representable. For the general case where the perspective reformulation is not SOC-representable, Furman et al. [21] proposed an ϵ-approximation of the convex hull of a disjunctive set which alleviates these numerical issues.

Frangioni and Gentile [13, 18] introduced projection approaches for additively separable closed convex functions. In the *projected perspective reformulation* (P^2R) [18], the perspective function is projected into the space of continuous variables and rewritten as a piecewise-convex function. This technique avoids the numerical issues associated with the perspective functions while yielding strong bounds, but at the cost of using piecewise-continuous functions which cannot be directly passed to off-the-shelf solvers. The *approximated projected perspective reformulation* (AP^2R) method [13] lifts the P^2R formulation back into the original space by reintroducing the indicator variables. AP^2R can be solved by general-purpose solvers and, if $\boldsymbol{y}^0 \leq \underline{\boldsymbol{y}}$, has the same number of variables and constraints as the original problem. The bound provided by AP^2R is generally weaker than the one from P^2R. An algorithm was proposed that enhances bound quality of AP^2R by using dual information obtained from P^2R [14], thus combining some of the bounding strength of P^2R with the computational efficiency of AP^2R.

Alternatively, the perspective reformulation (8) can be represented by an infinite number of linear outer approximations which are then dynamically separated. Suppose that we have a point $(\hat{x}_\ell, \hat{\boldsymbol{y}}, \hat{z})$ such that $\hat{z} \in (0, 1)$ and $\hat{x}_\ell < \tilde{f}(\hat{\boldsymbol{y}}, \hat{z})$. By performing first-order analysis of the convex envelope $\overline{co}\, f^d$, Frangioni and Gentile [15] derived cuts that separate $(\hat{x}_\ell, \hat{\boldsymbol{y}}, \hat{z})$ from the convex hull of F, referred to as *perspective cuts*:

$$\left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y} \right\rangle + \left(f(\boldsymbol{y}^*) - \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y}^* \right\rangle \right) z \leq x_\ell, \tag{9}$$

where $\boldsymbol{y}^* = \hat{\boldsymbol{y}}/\hat{z}$.

It is easy to adjust the perspective reformulation and the inequalities (9) for the case of nonzero $\boldsymbol{y}^0$ and $f(\boldsymbol{y}^0)$:

$$\left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y} - \boldsymbol{y}^0 \right\rangle + (f(\boldsymbol{y}^*) - f(\boldsymbol{y}^0) - \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y}^* - \boldsymbol{y}^0 \right\rangle) z + f(\boldsymbol{y}^0) \leq x_\ell, \tag{10}$$

where $\boldsymbol{y}^* = (\hat{\boldsymbol{y}} - \boldsymbol{y}^0)/\hat{z} + \boldsymbol{y}^0$. We refer to (10) as the *perspective cut* at $\boldsymbol{y}^*$.

2.2 Existing applications and computational results

Perspective cuts and reformulations were tested on several applications which contain convex functions of semi-continuous variables.

Frangioni and Gentile [15] applied perspective cuts (10) to the *thermal unit commitment* problem. In order to avoid the technical difficulties of incorporating perspective cuts into a general-purpose solver, the authors implemented their own NLP-based branch-and-cut algorithm. Perspective cuts are applied to the objective function via a specialized separation procedure, which replaces a univariate term with its perspective linearization if the perspective linearization is tighter at the current relaxation solution. The linearization is represented by an auxiliary variable, and as more perspective cuts are added for the term, the variable is set to be equal to the maximum of all linearizations. Perspective cuts were shown to have a considerable impact on the performance. The geometric mean of the running time of the best performing setting was smaller than that for the algorithm with the straightforward continuous relaxation by a factor of 60.

Perspective reformulations were studied by Günlük and Linderoth [25, 26]. Their key observation is that for some problems, the perspective reformulation can be written with the use of second-order cone constraints. The applications studied in this paper are:

- *Separable quadratic uncapacitated facility location* on a testset consisting of 16 instances. With the perspective reformulation, 50% more instances are solved within the time limit of 8 hours and on the instances that are solved with both formulations, the perspective formulation is faster by a factor of 8 when comparing the geometric mean.
- *Network design with congestion constraints* on a testset consisting of 35 instances. The perspective formulation is solved for 29 instances within the time limit of 4 hours, as opposed to only 2 instances with the standard formulation.
- *Mean-variance optimization (portfolio optimization)* on a testset consisting of 20 instances. Although none of the instances are solved within the time limit of 10,000 CPU seconds, perspective reformulation significantly improves the gap. For example, the final gap between the best found lower and upper bounds is reduced from 185.1% with the standard formulation to 4.2% with the perspective reformulation on instances of smaller size, and from 490.0 to 5.9% on instances of larger size.

Atamtürk and Gómez [2] applied the perspective-based conic reformulation to the *image segmentation* problem, testing it on 4 instances of different sizes. On the one instance that was solved within a time limit of 1 hour, the running time was reduced by a factor of 18 when compared to the standard formulation. On the three remaining instances, using the perspective formulation resulted in a 45–55% decrease of the remaining gap at time limit.

Aktürk et al. [1] presented a perspective-based conic reformulation of the *machine-job assignment problem with controllable processing times*. The tests were conducted on 180 randomly generated instances of varying sizes with quadratic and cubic objectives. For problems with a quadratic objective, 91% of the 90 instances were solved when using the strengthened conic formulation, whereas at most 36% of instances were solved when using non-perspective formulations. For problems with a cubic objective, 88% of the 90 instances were solved with the perspective formulation and at most 27% were solved with non-perspective formulations.

A comparison between SOC-based perspective formulations and perspective cutting planes was performed by Frangioni and Gentile [17]. Using the CPLEX-11 solver, they tested the two approaches on two sets of mixed-integer quadratic problems, namely, the *Markowitz mean-variance model* and the *unit commitment problem*. The difference between the two formulations is particularly significant with the setup used in the paper [17] since by default, CPLEX obtains dual bounds by solving nonlinear relaxations. The results favor the cutting planes approach, with the difference being larger for the Markowitz mean-variance problem. The authors observed that the advantage of perspective cuts stems mostly from efficient reoptimization of linear programs. They add that the perspective conic reformulation is more competitive for problems that are larger, more nonlinear (i.e., have more nonlinear constraints or non-quadratic nonlinear constraints) or have richer structure.

Frangioni and Gentile [13, 18] tested the projected perspective reformulation (P^2R) and the approximated perspective projected reformulation (AP^2R) on *sensor placement*, *nonlinear network design*, *mean-variance portfolio* and *unit commitment* problems. P^2R was implemented as part of a specialized branch-and-bound algorithm and AP^2R was solved directly with CPLEX 12. Both approaches were compared to perspective cuts implemented as a callback in CPLEX. Computational results show that P^2R is the best performing method for problems that have a well-suited structure and require little or no branching. For problems with more complex structures, AP^2R is competitive with the perspective cut approach. When there are constraints linking indicator variables and few linear approximations provide a good estimate of the original nonlinear function, perspective cuts tend to be the best performing method.

Salgado et al. [33] studied the *alternating current optimal power flow problem with activation/deactivation of generators* (ACOPFG) using 8 test instances. They tested two perspective-based reformulations of the objective function. The first uses four perspective cuts of the form (10); the other is obtained by applying AP^2R [13]. Although the results of enhancing the standard ACOPFG model with perspective reformulations are inconclusive, an outer approximation [34] of the problem significantly benefits from both perspective cuts and AP^2R. The perspective cuts approach performs best, solving one more instance than the standard formulation within the time limit of

1 hour and taking less than 4 s on all the remaining instances, whereas the standard formulation requires over 1000 s on most instances.

The applications on perspective cuts are not limited to the structure given by (2). Gómez [23] considered mixed-integer sets given by a conic quadratic constraint with semi-continuous variables and proposed a convex hull formulation for a special case when some variables are unbounded, and valid inequalities for the bounded case. Anstreicher and Burer proposed a convex hull representation for the set $\{(\boldsymbol{y}, \boldsymbol{y}\boldsymbol{y}^T, \boldsymbol{z}\boldsymbol{z}^T) \mid 0 \leq \boldsymbol{y} \leq \boldsymbol{z} \in \{0,1\}^n\}$ for the case $n = 2$. For mixed-integer quadratic problems, perspective-based approaches were developed which decompose the problem Hessian in ways that enable the use of perspective reformulations [16, 19, 40]. These reformulations either have to be developed specifically for a narrower class of constraints, or require non-trivial transformations of the problem to be performed before the start of the solution process. Such tailored approaches may yield better performance than the general-purpose approach that this paper focuses on, at the cost of the necessity to develop such approaches for special, more restrictive classes of problems.

In many of the aforementioned applications, perspective cuts were applied only to the objective function. In this work, we focus on constraints, which is without loss of generality since the objective function can always be moved into a constraint, and the theory developed for objective functions can be directly adapted to the case of constraints.

3 Generalized perspective cuts

If f is nonconvex, neither the gradient cuts (3) nor the perspective cuts (10) are guaranteed to be valid. However, the perspective reformulation can be applied to a convex underestimator of f, from which the perspective cuts (10) can be derived. Alternatively, any linear inequality $\phi(\boldsymbol{y}) \leq x_\ell$ that is valid for the 'on' set F^1 can be adjusted for the 'off' set F^0.

In the following, we propose a cut extension procedure that ensures that the generated inequality is equivalent to $\phi(\boldsymbol{y}) \leq x_\ell$ when the indicator is equal to 1 and holds with equality at the point $(x_\ell, \boldsymbol{y}, z) = (f(\boldsymbol{y}^0), \boldsymbol{y}^0, 0)$.

Theorem 1 *(Generalized perspective cuts) Consider a vector of semi-continuous variables $\boldsymbol{y} \in \mathbb{R}^p$ with an indicator $z \in \{0,1\}$, such that $\boldsymbol{y} = \boldsymbol{y}^0$ if $z = 0$, and a linear inequality $\phi(\boldsymbol{y}) \leq x_\ell$ that is valid for the set*

$$F^1 = \{(x_\ell, \boldsymbol{y}, z) \mid x_\ell \geq f(\boldsymbol{y}),\ \ \boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}],\ \ z = 1\}.$$

Let

$$\tilde{\phi}(\boldsymbol{y}, z) = \phi(\boldsymbol{y}) + \Big(f(\boldsymbol{y}^0) - \phi(\boldsymbol{y}^0)\Big)(1 - z).$$

Then the linear inequality $\tilde{\phi}(\boldsymbol{y}, z) \leq x_\ell$ *is valid for the set* $F^0 \cup F^1$*, where*

$$F^0 = \{(x_\ell, \boldsymbol{y}, z) \mid \ x_\ell \geq f(\boldsymbol{y}^0), \ \boldsymbol{y} = \boldsymbol{y}^0, \ z = 0\}.$$

Proof It is sufficient to check the validity for each possible value of $z \in \{0, 1\}$. By substituting $z = 1$ and $z = 0$ in $\tilde{\phi}(\boldsymbol{y}, z)$, we immediately obtain

1. $\tilde{\phi}(\boldsymbol{y}, 1) = \phi(\boldsymbol{y}) \ \forall \boldsymbol{y} \in \mathbb{R}^p$ and
2. $\tilde{\phi}(\boldsymbol{y}^0, 0) = f(\boldsymbol{y}^0)$,

respectively. Therefore, $\tilde{\phi}(\boldsymbol{y}, z) \leq x_\ell$ is a valid inequality. □

If the cut $\phi(\boldsymbol{y}) \leq x_\ell$ is already valid for F^0, then the described above adjustment always produces a cut that is at least as strong as the original cut. Since $\phi(\boldsymbol{y}) \leq x_\ell$ is in this case implied by $f(\boldsymbol{y}) \leq x_\ell$ for $(x_\ell, \boldsymbol{y}) \in F^0$, we have $\phi(\boldsymbol{y}^0) \leq f(\boldsymbol{y}^0)$. Hence the coefficient of $(1 - z)$ in $\tilde{\phi}(\boldsymbol{y}, z)$ is nonnegative and

$$\tilde{\phi}(\boldsymbol{y}, z) \geq \phi(\boldsymbol{y}), \ \forall z \in [0, 1], \ \forall \boldsymbol{y} \in \mathbb{R}^p.$$

If additionally $\phi(\boldsymbol{y}^0) < f(\boldsymbol{y}^0)$, i.e., the original cut is not tight at $\boldsymbol{y}^0$, then the new cut is also stronger. Otherwise, if $\phi(\boldsymbol{y}) \leq x_\ell$ does not hold for F^0 (that is, if $\phi(\boldsymbol{y}^0) > f(\boldsymbol{y}^0)$), then the adjustment is necessary to obtain a cut that is valid for $F^0 \cup F^1$.

This cut extension procedure has two main advantages:

1. It does not depend on the convexity of f and requires no assumptions on the cut except for its validity for F^1.
2. In the case where $\boldsymbol{y}^0 \notin [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}]$, variable bounds for F^1 are tighter than those for $F^0 \cup F^1$. This is useful for nonconvex constraints since the tightness of their relaxations depends on variable bounds, and therefore cuts constructed for $\boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}]$ will generally be stronger than those for $\boldsymbol{y} \in [\min\{\boldsymbol{y}^0, \underline{\boldsymbol{y}}\}, \max\{\boldsymbol{y}^0, \overline{\boldsymbol{y}}\}]$.

When the cut strengthening is applied to the convex setting, the result is equivalent to the well-known perspective cuts:

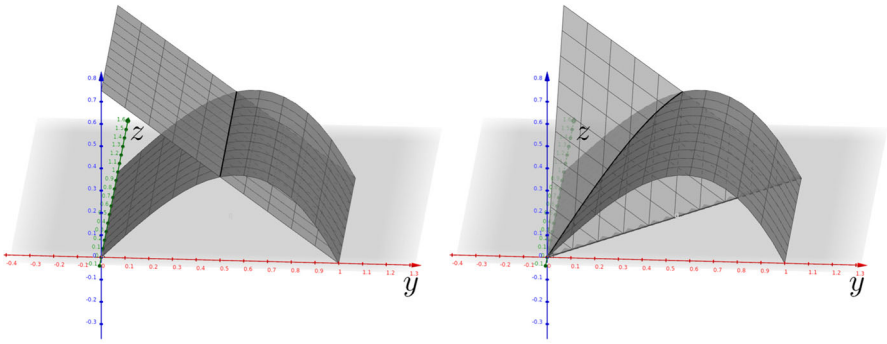
Theorem 2 (Alternative derivation of perspective cuts) *Suppose that* $f : \mathbb{R}^p \to \mathbb{R}$ *is convex and* $(\hat{x}_\ell, \hat{\boldsymbol{y}}, \hat{z}) \notin conv(F)$ *as defined in Sect. 2.1. Consider the gradient cut (3) at point* $\boldsymbol{y}^* = (\hat{\boldsymbol{y}} - \boldsymbol{y}^0)/\hat{z} + \boldsymbol{y}^0$ *for Constraint (2):*

$$\phi(\boldsymbol{y}) = f(\boldsymbol{y}^*) + \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y} - \boldsymbol{y}^* \right\rangle \leq x_\ell.$$

Let $\tilde{\phi}(\boldsymbol{y}, z)$ *be the linear function obtained from* $\phi(\boldsymbol{y})$ *by following the strengthening procedure in Theorem 1. Then* $\tilde{\phi}(\boldsymbol{y}, z)$ *is written as follows:*

$$\tilde{\phi}(\boldsymbol{y}, z) = \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y} - \boldsymbol{y}^0 \right\rangle + (f(\boldsymbol{y}^*) - f(\boldsymbol{y}^0) - \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y}^* - \boldsymbol{y}^0 \right\rangle) z + f(\boldsymbol{y}^0)$$

and the cut $\tilde{\phi}(\boldsymbol{y}, z) \leq x_\ell$ *is equivalent to the perspective cut (10) at point* $(\hat{\boldsymbol{y}}, \hat{z})$.



(a) original cut valid for $z = 1$, $y \in [0.5, 1]$ (b) generalized perspective cut

Fig. 2 Example of cut extension (created with Geogebra https://www.geogebra.org/)

Proof The coefficient of $(1 - z)$ in $\tilde{\phi}(\boldsymbol{y}, z)$ is

$$\alpha = f(\boldsymbol{y}^0) - \phi(\boldsymbol{y}^0) = f(\boldsymbol{y}^0) - f(\boldsymbol{y}^*) - \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y}^0 - \boldsymbol{y}^* \right\rangle.$$

Adding $\alpha(1 - z)$ to the left hand side of the gradient cut produces the perspective cut (10):

$$\tilde{\phi}(\boldsymbol{y}, z) = \phi(\boldsymbol{y}) + \alpha(1 - z) =$$
$$\left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y} - \boldsymbol{y}^0 \right\rangle + (f(\boldsymbol{y}^*) - f(\boldsymbol{y}^0) - \left\langle \nabla f(\boldsymbol{y}^*), \boldsymbol{y}^* - \boldsymbol{y}^0 \right\rangle)z + f(\boldsymbol{y}^0).$$

□

To paraphrase, for a convex function f the perspective cut at a solution $(\hat{x}_\ell, \hat{\boldsymbol{y}}, \hat{z})$ of the LP relaxation can equivalently be obtained by first generating a gradient cut for f at the modified point $\boldsymbol{y}^*$ and then applying the strengthening procedure from Theorem 1.

Let us consider an example to illustrate the cut extension method.

Example 1 Consider a constraint $f(y) = -y^3 + y \leq x_\ell$. The boundary of the feasible region is shown in Fig. 2 by the dark gray nonlinear surface, and the feasible points are located above it. Let $0.5z \leq y \leq z$, where y is a scalar, semi-continuous variable modeled using the binary variable $z \in \{0, 1\}$.

First we find an underestimator of $f(y)$ valid for $z = 1$. In this case, y is constrained to belong to the interval $[0.5, 1]$. Since $f(y)$ is concave on $[0.5, 1]$, the underestimator is the secant through points $(0.5, f(0.5))$ and $(1, f(1))$:

$$f^{sec}(y) = -0.75y + 0.75.$$

The cut $f^{sec} \leq x_\ell$ (shown in Fig. 2a) is not valid for the whole feasible set. In particular, a feasible point $(y, z, x_\ell) = (0, 0, 0)$ violates the cut: $f^{sec}(0, 0) = 0.75 > 0 = x_\ell$.

Now we extend the cut so that to ensure validity at $z = 0$. By Theorem 1, the new cut is written as:

$$\tilde{f}^{sec} = f^{sec} + (f(y^0) - f^{sec}(y^0))(1 - z) = -0.75y + 0.75z.$$

This cut, shown in Fig. 2b, is valid for the whole feasible set given by the cubic constraint and the semi-continuity condition.

4 Implementation of perspective cuts

An effective implementation of perspective cuts within a general-purpose solver requires providing methods for detecting suitable structures in a general problem and generating the cuts during the solution process. In the following, we describe our implementation within SCIP, but many considerations discussed here will be applicable to MINLP solvers in general.

4.1 Organization of nonlinear constraints in SCIP

SCIP builds a relaxation for the MINLP (1) by means of an extended formulation, where auxiliary variables $\boldsymbol{w}$ are introduced for the subexpressions that constitute the constraint functions $g(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})$. Without loss of generality, we can assume that (1b) has been replaced by a new system

$$h_i(\boldsymbol{x}, \boldsymbol{y}, w_1, \ldots, w_{i-1}, \boldsymbol{z}) \lesseqgtr w_i, \qquad i = 1, \ldots, m', \tag{11a}$$

$$\boldsymbol{w}^\ell \leq \boldsymbol{w} \leq \boldsymbol{w}^u, \tag{11b}$$

where $\boldsymbol{w}^l$ and $\boldsymbol{w}^u$ denote global lower and upper bounds on $\boldsymbol{w}$.

The handling of nonlinear constraints in the version of SCIP used for this work is performed by modules called "nonlinearity handlers". Each nonlinearity handler works on a specific structure (e.g. quadratic, convex, etc.) and provides callback methods. For the purposes of this paper, three types of callbacks are relevant:

- *Detection callbacks* receive an expression and determine whether it is suitable for the nonlinearity handler.
- *Estimation callbacks* provide linear under- and overestimators given an expression and a point at which to linearize it.
- *Enforcement callbacks* enforce a given violated constraint by adding cutting planes, tightening bounds, detecting infeasibility, etc.

Our addition of generalized perspective cuts is implemented via a specialized perspective nonlinearity handler.

4.2 Structure detection

The detection algorithm identifies constraints of the form (11), where h_i is nonlinear and at least one other nonlinearity handler provides an estimation callback for it. All variables that h_i depends on must be semi-continuous with at least one common indicator variable. If several binary variables satisfying this condition are found, all such variables are stored for use in cut generation.

A special case is that of h_i being a sum. Here, only the variables appearing in nonlinear terms of the sum are required to be semi-continuous.

To determine whether a variable y_j is semi-continuous, the detection callback of the perspective nonlinearity handler searches for pairs of implied bounds on y_j with the same indicator z_k:

$$\begin{aligned} y_j &\leq \alpha^{(u)} z_k + \beta^{(u)}, \\ y_j &\geq \alpha^{(\ell)} z_k + \beta^{(\ell)}. \end{aligned}$$

If $\beta^{(u)} = \beta^{(\ell)}$, then y_j is a semi-continuous variable and $y_j^0 = \beta^{(u)}$, $\underline{y}_j = \alpha^{(\ell)} + \beta^{(\ell)}$ and $\overline{y}_j = \alpha^{(u)} + \beta^{(u)}$.

This information can be obtained either directly from linear constraints in y_j and z_k, or by finding implicit relations between y_j and z_k. Such relations can be detected by probing, which fixes z_k to its possible values and propagates all constraints in the problem, thus detecting implications of $z_k = 0$ and $z_k = 1$. SCIP stores the implied bounds in a globally available data structure.

In addition, the perspective nonlinearity handler detects semi-continuous auxiliary variables, that is, variables w_i that were introduced to express the extended formulation (11). Given $h_i(\boldsymbol{y}, w_1, \ldots, w_{i-1}) \lesseqqgtr w_i$, where variables $\boldsymbol{y}$ and $w_1, \ldots, w_{i-1}$ are semi-continuous and depend on the same indicator z_k, the auxiliary variable w_i is semi-continuous with $w_i^0 = h_i(\boldsymbol{y}^0, w_1^0, \ldots, w_{i-1}^0)$ and $[\underline{w}_i, \overline{w}_i] = h_i([\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0], [\underline{w}_1, \overline{w}_1], \ldots, [\underline{w}_{i-1}, \overline{w}_{i-1}])$ computed by interval arithmetic.

According to Theorem 1, the constraint must have the form

$$w_i \geq h_i(\boldsymbol{y}, w_1, \ldots, w_{i-1}),$$

where all variables $\boldsymbol{y}, w_1, \ldots, w_{i-1}$ are semi-continuous with respect to the same indicator z_k. In our implementation we allow a more general form:

$$\begin{aligned} h_i(\boldsymbol{x}, \boldsymbol{y}, w_1, \ldots, w_{i-1}, \boldsymbol{z}) = {} & h_{i,k}^{sc}(\boldsymbol{y}, w_1, \ldots, w_r) \\ & + h_{i,k}^{nsc}(\boldsymbol{x}, w_{r+1}, \ldots, w_{i-1}, \boldsymbol{z}) \lesseqqgtr w_i, \end{aligned} \tag{12}$$

where h^{sc} is a nonlinear function, h^{nsc} is a linear function, variables $\boldsymbol{y}$ and $w_1, \ldots, w_r$ are semi-continuous and variables $\boldsymbol{x}, w_{r+1}, \ldots, w_{i-1}$ can be non-semi-continuous. Auxiliary variables are assumed to be sorted so that semi-continuous variables $w_1, \ldots, w_r$ come before the non-semi-continuous variables $w_{r+1}, \ldots, w_{i-1}$ and the

variable w_i representing h_i. The semi-continuity of auxiliary variables is determined based on the semi-continuity of the expressions they represent.

Thus, for each suitable indicator z_k the function h_i is split up into the semi-continuous part $h^{sc}_{i,k}$, which can depend only on variables that are semi-continuous with respect to indicator z_k, and a non-semi-continuous part $h^{nsc}_{i,k}$, which depends on non-semi-continuous variables. The non-semi-continuous part must be linear. If the sum has a constant term, the constant is considered to be part of $h^{sc}_{i,k}$.

4.3 Separation and strengthening of generalized perspective cuts

During the cut generation loop, generalized perspective cuts as in Theorem 1 are constructed for constraints of the form (12).

In the following, let $\boldsymbol{v}$ denote the vector of all problem variables: $\boldsymbol{v} = (\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{w}, \boldsymbol{z})$, and let $F^1_{i,k}$ and $F^0_{i,k}$ be the sets of points satisfying a constraint of the form (12) for a given $i \in \{1, \ldots, m'\}$ together with implied variable bounds for $z_k = 1$ and $z_k = 0$, respectively. For simplicity, we fix the inequality sign and consider "less than or equal to" constraints:

$$\begin{aligned} F^1_{i,k} =& \{\boldsymbol{v} \mid h_i(\boldsymbol{x}, \boldsymbol{y}, w_1, \ldots, w_{i-1}, \boldsymbol{z}) \leq w_i, \boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}], \boldsymbol{w} \in [\underline{\boldsymbol{w}}, \overline{\boldsymbol{w}}], z_k = 1\}, \\ F^0_{i,k} =& \{\boldsymbol{v} \mid h_i(\boldsymbol{x}, \boldsymbol{y}, w_1, \ldots, w_{i-1}, \boldsymbol{z}) \leq w_i, \boldsymbol{y} = \boldsymbol{y}^0, \boldsymbol{w} = \boldsymbol{w}^0, z_k = 0\}. \end{aligned}$$

Suppose that the point $\hat{\boldsymbol{v}}$ violates the nonlinear constraint:

$$h_i(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{w}_1, \ldots, \hat{w}_{i-1}, \hat{\boldsymbol{z}}) > \hat{w}_i.$$

If h_i is nonconvex and its estimators depend on variable bounds, the enforcement callback first performs probing for $z_k = 1$ in order to tighten the implied bounds $[\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}]$ and $[\underline{w}_j, \overline{w}_j]$ for $j \leq r$.

Estimation callbacks of non-perspective nonlinearity handlers are called in order to find valid cuts that separate $\hat{\boldsymbol{v}}$ from $F^1_{i,k}$, which are then modified according to Theorem 1. For a constraint of the generalized form $h_i = h^{sc}_{i,k} + h^{nsc}_{i,k} \leq w_i$, which is described in Sect. 4.2, an estimation callback will provide an underestimator of h_i:

$$\underline{h}_i = \underline{h}^{sc}_{i,k} + h^{nsc}_{i,k}.$$

This underestimator consists of an underestimator of the semi-continuous part $\underline{h}^{sc}_{i,k}$ and the non-semi-continuous part $h^{nsc}_{i,k}$, which remains unchanged since it is already linear and shares none of the variables with the semi-continuous part. The extension procedure from Theorem 1 is applied only to $\underline{h}^{sc}_{i,k}$ to obtain $\tilde{\underline{h}}^{sc}_{i,k}$. Since $h^{sc}_{i,k}$ depends only on semi-continuous variables, similar arguments to Sect. 3 hold for feasibility and tightness of the strengthened underestimator $\tilde{\underline{h}}^{sc}_{i,k}$. The strengthened underestimator of h_i is then written as

$$\tilde{\underline{h}}_{i,k} = \tilde{\underline{h}}^{sc}_{i,k} + h^{nsc}_{i,k}.$$

If $\hat{\boldsymbol{v}}$ violates the cut $\underline{\tilde{h}}_{i,k} \leq w_i$, the cut is passed to the SCIP core where it will be considered for addition to the LP relaxation.

Let us consider an example of generalized perspective cut separation by extending Example 1.

Example 2 The extended formulation of the constraint from Example 1 is written as:

$$h(x_l, y) = -y^3 + y - x_l \leq w \leq 0.$$

The semi-continuous part of h is $h^{sc}(y) = -y^3 + y$, and the non-semi-continuous part is $h^{nsc}(x_l) = -x_l$. The perspective underestimator of h is $\underline{h}(x_l, y, z) = -0.75y + 0.75z - x_l$ and the perspective cut is written as follows:

$$-0.75y + 0.75z - x_l \leq w.$$

Suppose that the point to be separated is $(\hat{x}_l, \hat{y}, \hat{z}, \hat{w}) = (0, 0.4, 0.7, 0)$. Substituting the variables with their values at this point in the perspective cut, we get a violated inequality $0.225 \leq 0$. Therefore the cut is violated and will be considered for addition to the LP relaxation.

5 Computational results

This section presents the results of computational experiments. A development version of SCIP [6] was used, together with the linear solver SoPlex 5.0.1.3 [22] and the nonlinear solver Ipopt 3.12.13 [38]. All the experiments were run on a cluster of 3.60GHz Intel Xeon E5-2680 processors with 64 GB memory per node. The time limit was set to one hour and the optimality gap limit to 0.01%.

The basis of our experiments were the instances of MINLPLib. These include also instances from QPLib[2] [20], and we analyze results on subsets of convex/nonconvex and quadratic/non-quadratic instances in Sect. 5.3.

Throughout the section, we analyze the following settings, each defined by the types of constraints for which perspective cuts are added:

- *Off*: perspective cuts are disabled;
- *Convex*: perspective cuts are enabled only for convex constraints;
- *Full*: perspective cuts are enabled for both convex and nonconvex constraints.

5.1 Detection of suitable structures

Out of the 1703 instances of MINLPLib, suitable constraints of the extended form (12) were detected for 186 instances. Table 1 shows the numbers of instances where at least one such constraint was detected, when counting: all instances, instances where detection succeeded for convex constraints only, instances where detection succeeded both for convex and nonconvex constraints and instances where detection succeeded for nonconvex constraints only.

[2] https://qplib.zib.de.

Table 1 Detection results

All	Convex	Both	Nonconvex
186	89	53	44

Only those instances were counted for Table 1 where suitable constraints were detected in the main problem. Additionally, sometimes a constraint can only be detected by the perspective nonlinearity handler in a subproblem. A typical example of this is a heuristic creating a subproblem to represent a restricted version of the main problem. This often involves fixing some variables or modifying bounds, which can result in new semi-continuous variables and thus new suitable constraints. In our test set there are 3 instances where suitable constraints were found only in subproblems. However, subproblem detections are not guaranteed to have an impact on performance. Because of this, subproblem detections are not counted in Table 1.

As pointed out in Sect. 2, in many problems where perspective cuts can be applied, all expressions defined in terms of semi-continuous variables are contained in the objective function. However, since SCIP supports linear objective functions only, any nonlinear term $f(x)$ in the objective is replaced by an auxiliary variable x_{obj}, and a constraint $f(x) \leq x_{obj}$ is added. Therefore, a relevant question to ask is how many constraints in the model are suitable for perspective cuts.

To this end, in Fig. 3 we report for each instance the numbers of nonlinear constraints (11a) in the extended formulation (*Nconss*) and the numbers of such constraints where perspective cuts can be applied (*Ndetects*). Comparing these numbers shows what fraction of the nonlinear part of an instance is amendable to the perspective approach, and is motivated by the fact that it is the nonlinear part of an instance that presents the most difficulty for the solver and determines how difficult to solve the instance is. The plot shows that this fraction tends to be large, with many points on the diagonal, representing the instances where all nonlinear constraints fit the structure suitable for perspective cuts, and only few points far below the diagonal, representing the instances where the number of nonlinear constraints fitting this structure is considerably less than the number of all nonlinear constraints.

5.2 Overall performance impact

This section evaluates the overall impact of perspective cuts on the performance of SCIP. Its purpose is to give an overview of how the three major settings compare against each other before moving onto more detailed comparisons in the next section.

In order to robustify our results against the effects of performance variability [31], four different permutations of the order of variables and constraints were applied to each of the 186 instances, for which suitable structures were detected. Each permutation is treated as a separate instance, and together with the instances without any permutation they comprise a test set of 930 instances. In our analysis, we exclude instances where one of the solver settings encountered numerical troubles or where the numerical results of the different solver settings are inconsistent.

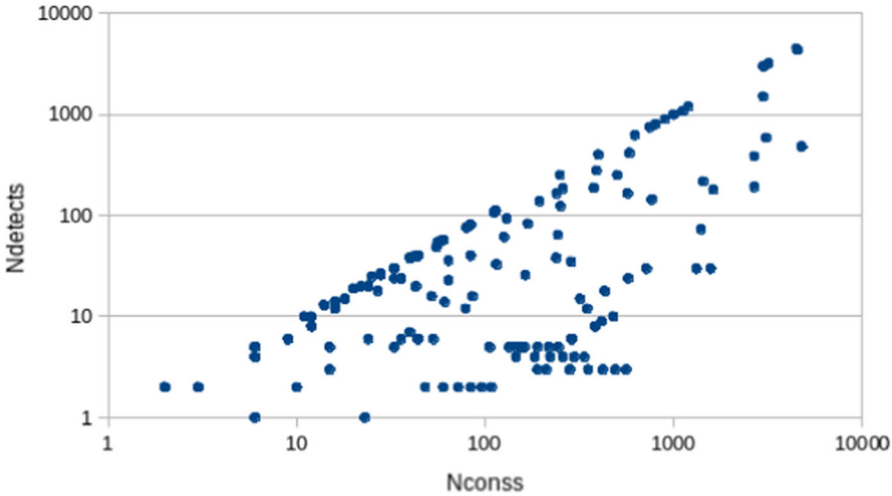


Fig. 3 Numbers of nonlinear constraints in the extended formulation (*Nconss*) and constraints amenable to perspective cuts (*Ndetects*) (created with LibreOffice https://www.libreoffice.org/)

Table 2 Overview of solved instances

	Off	Convex	Full
Solved	741	764	759
Limit	175	154	154
Fails	14	12	17

Table 3 Overall results on the subset of 672 affected instances

	Off	Convex	Full
Time	13.79	11.23	11.27
Relative time	1.00	0.81	0.82
Nodes	620	479	472
Relative nodes	1.00	0.77	0.76

Table 2 provides an overview of the number of instances solved to global optimality by each setting. The row *Limit* contains the count of instances where the time limit was reached. The row *Fails* reports the number of instances where numerical troubles were encountered. The largest number of instances solved with a given setting was 764, yielded by setting *Convex*, which had both the smallest number of numerical fails and time outs. It is followed by *Full*, which solved 759 instances. The setting *Off* solved the least number of instances overall.

Table 3 shows the shifted geometric mean of the running time in seconds (with a shift of 1 s) and the shifted geometric mean of the number of branch-and-bound nodes (with a shift of 100 nodes). All numbers in the table are computed for the subset of 672 *affected* instances: all instances where at least two of the three settings yielded a different solving path (judged by a different number of linear programming iterations),

Table 4 Overall results on the subset of 404 affected convex instances

	Off	Convex	Full
Time	9.58	6.95	6.95
Relative time	1.00	0.73	0.73
Nodes	502	329	329
Relative nodes	1.00	0.66	0.66

Table 5 Overall results on the subset of 268 affected nonconvex instances

	Off	Convex	Full
Time	23.50	22.40	22.61
Relative time	1.00	0.95	0.96
Nodes	844	809	781
Relative nodes	1.00	0.96	0.93

Table 6 Overall results on the subset of 215 affected quadratic instances

	Off	Convex	Full
Time	36.64	24.55	24.78
Relative time	1.00	0.67	0.68
Nodes	448	258	248
Relative nodes	1.00	0.58	0.55

where the solver failed with none of the settings, and solved the instance to optimality with at least one setting.

From Table 3 one can see that a significant improvement is achieved when enabling perspective cuts for convex constraints. The results with the settings *Convex* and *Full*, however, are almost identical.

5.3 Overall performance impact by instance type

In this section, we evaluate the impact of perspective cuts on solver performance when considering subsets of instances grouped by type: convex and nonconvex instances, and quadratic and general nonlinear instances.

Tables 4 and 5 show results obtained for the convex and nonconvex subsets of affected instances, respectively. The impact of enabling perspective cuts for convex constraints is more pronounced on the convex instance subset, but some impact can be observed on the nonconvex instance subset as well, since nonconvex instances can contain convex constraints. Enabling perspective cuts for nonconvex constraints, as expected, does not have any effect on the convex instance subset, and is consistent with what is shown in Table 3 but more pronounced on the nonconvex instance subset.

From Tables 6 and 7 we can see that the impact of perspective cuts on quadratic instances is greater than the impact on general nonlinear instances. The effect of enabling perspective cuts for nonconvex constraints in addition to perspective cuts for convex constraints remains small for both subsets.

Table 7 Overall results on the subset of 457 affected general nonlinear instances

	Off	Convex	Full
Time	8.53	7.65	7.65
Relative time	1.00	0.90	0.90
Nodes	719	626	622
Relative nodes	1.00	0.87	0.87

Table 8 Relevant and affected instances

	Off vs *Convex*	*Convex* vs *Full*
Relevant	710	485
Affected	544	205

Table 9 Root node dual bound differences

	Off	Convex	Convex	Full
better by > 50%	16	46	0	31
better by 5–50%	25	39	14	11
same within 5%	584		429	

5.4 Detailed comparisons

In this section we provide a more detailed analysis of the performance results by comparing pairs of settings, in order to evaluate the impact of each major feature more thoroughly.

First, we present the numbers of relevant and affected instances in Table 8. It has the following rows:

- *Relevant:* the number of instances where more expressions are detected with the second setting than with the first setting;
- *Affected:* the number of instances which were solved with at least one of the two settings, where the number of linear programming iterations differs between the two settings and the solver failed with none of the two settings.

From Table 8 we can see that while nonconvex structures can be found on more than half of the test set, applying generalized perspective cuts to nonconvex functions affects the solving path less often than applying perspective cuts to convex functions.

Table 9 summarizes the effect of perspective cuts on the dual bound at the end of the root node. It reports the numbers of instances of the subset *Relevant* where the dual bound was better with the corresponding setting by a percentage that is specified in the first column, as well as the numbers of instances where the dual bound change was less than 5%.

A significant difference in root node dual bound can be observed only for a relatively small number of instances. The comparison between *Off* and *Convex* is consistent with the results in the above tables, with *Convex* improving more dual bounds than *Off*. The comparison between *Convex* and *Full* deserves a closer look. When inspecting medium dual bound changes (5–50%), *Convex* yields a better bound than *Full* slightly more

Table 10 Time on subsets of affected instances

	Off	Convex	Convex	Full
Instances in $[0, 3600]$:	544		205	
Time	12.53	9.70	24.30	24.82
Relative time	1.00	0.77	1.00	1.02
Faster	95	193	43	51
Instances in $[10, 3600]$:	276		149	
Time	70.96	45.27	57.47	59.12
Relative time	1.00	0.64	1.00	1.03
Faster	50	122	29	35
Instances in $[100, 3600]$:	100		49	
Time	506.17	183.90	263.57	285.85
Relative time	1.00	0.36	1.00	1.08
Faster	18	64	13	15
Instances in $[1000, 3600]$:	45		14	
Time	1444.28	425.60	814.18	1034.83
Relative time	1.00	0.29	1.00	1.27
Faster	10	32	5	5

often than the other way round. However, when considering only large improvements ($> 50\%$), we observe that those were always due to setting *Full*. From this we conclude that, overall, enabling perspective cuts for nonconvex constraints improves the quality of dual bounds in the root node.

On several instances, a decrease in the quality of root node dual bounds can be observed when enabling perspective cuts. This is due to the fact that adding a different cut results in a different LP, which leads to different, possibly better, cuts being added, as well as changes in many other parts of the solution process which can likewise contribute to an improvement in the dual bound.

Table 10 compares the running time when considering pairs of settings and the corresponding subsets of *affected* instances. It has the following rows:

- Time: shifted geometric mean of the running time in seconds (with a shift of 1 s);
- Relative time: shifted geometric mean of the running time relative to the first of the two settings;
- Faster: the number of instances where SCIP was faster with the given setting than with the other setting by at least 25%.

In order to analyze the impact on subsets of increasingly hard instances, these rows repeat for three subsets of instances given by time brackets $[t, 3600]$, which contain the instances that were solved to optimality with both settings and took at least t s by at least one setting.

The results shown in Table 10 strongly confirm that enabling perspective cuts for convex constraints decreases the mean running time. This effect becomes more pronounced as the difficulty of the instances increases. On instances that took at least 100 s to solve, setting *Convex* was faster almost by a factor of 3. The additional activation of perspective cuts for nonconvex constraints, however, rather had a detrimental effect

Table 11 Number of nodes on subsets of affected instances

	Off	Convex	Convex	Full
Nodes on $[0, 3600]$	775	567	619	590
Relative	1.00	0.73	1.00	0.95
Nodes on $[10, 3600]$	4289	2436	1188	1170
Relative	1.00	0.57	1.00	0.98
Nodes on $[100, 3600]$	24924	7819	14891	15503
Relative	1.00	0.31	1.00	1.04
Nodes on $[1000, 3600]$	46517	15638	166889	199558
Relative	1.00	0.34	1.00	1.20

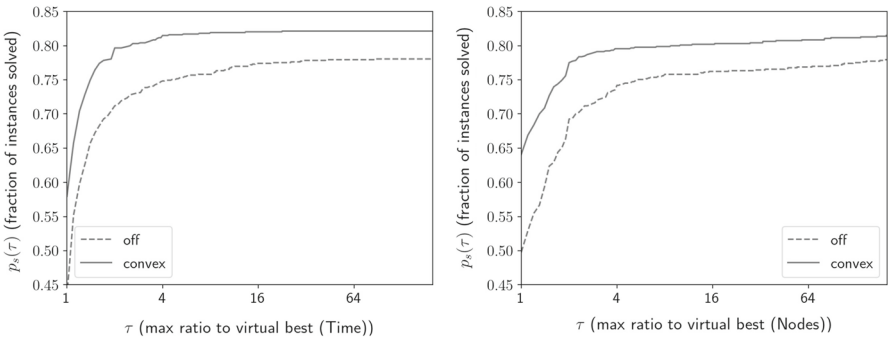


Fig. 4 Performance profiles comparing *Off* and *Convex* (created with Matplotlib https://matplotlib.org/)

on performance, especially as instances become more difficult. This is despite the fact that there are more speed-ups than slow-downs when switching from setting *Convex* to setting *Full*, as seen from the rows *Faster*. Hence, the increase in the mean time is due to significant slow-downs on a few challenging instances. However, these observed slow-downs should not be overestimated since the size of the subsets [100, 3600] and [1000, 3600] are comparatively small.

A comparison of branch-and-bound tree sizes is given in Table 11. Again, a consistent improvement is observed when enabling perspective cuts for convex expressions. This improvement becomes more pronounced as the instances become more challenging. Here we also observe an overall improvement when enabling perspective cuts for nonconvex constraints. However, on the harder subsets [100, 3600] and [1000, 3600], *Convex* still remains the best setting.

Figures 4 and 5 show performance profiles [11] for running time and number of nodes with settings *Off* and *Convex* and settings *Convex* and *Full*, respectively. Let $t_{i,s}$ denote the running time for instance i with setting s. The virtual best setting used for the running time performance profiles, denoted by index vb, is defined as a setting whose running time for each instance is equal to the minimum of the running times with the two settings that are being compared: $t_{i,vb} = \min_s\{t_{i,s}\}$. The horizontal axis represents the maximum allowed ratios to the time with the virtual best setting, denoted

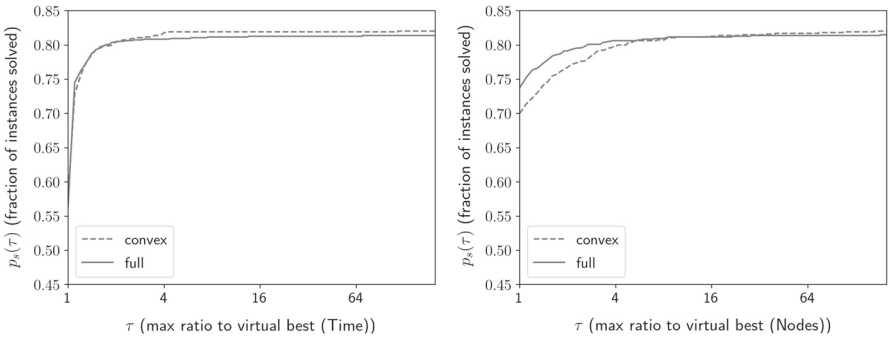


Fig. 5 Performance profiles comparing *Convex* and *Full* (created with Matplotlib https://matplotlib.org/)

by τ. The vertical axis represents the fraction of instances solved within the maximum allowed fraction of time of the virtual best, denoted as $p_s(\tau)$:

$$p_s(\tau) = \frac{\text{number of instances } i \text{ s.t.: } t_{i,s} \leq \tau \cdot t_{i,vb}}{\text{total number of instances}}.$$

Performance profiles for the number of nodes in the branch-and-bound tree are constructed similarly, the only difference being that $t_{.,.}$ is replaced everywhere with $n_{.,.}$, which represents the number of nodes per instance and setting.

From Fig. 4 we see that setting *Convex* dominates setting *Off*. With *Convex*, over 55% of instances are solved faster or as fast as with setting *Off*. It is able to solve around 80% of instances within a factor of 4 of the best time, and the curve approaches approximately 82% in the limit, i.e., *Convex* is able to solve around 82% of the instances. The respective numbers for *Off* lie at around 10% lower than those for *Convex*. A very similar picture is observed for the number of nodes.

According to Fig. 5, the setting *Full* is roughly on par with *Convex* in terms of running time if the ratio we are interested in is below 3. As the ratio increases, *Convex* becomes the better setting, which reflects the fact that it solves slightly more instances than *Full*. When looking at the number of nodes, *Full* yields the best result for around 74% of instances, as opposed to around 70% yielded by *Convex*. For ratios above 16, *Convex* is better than *Full*, which, again, is due to it solving more instances.

5.5 Feature evaluation: bound tightening

As explained in Sect. 4.3, if the constraint is nonconvex and its relaxation depends on variable bounds, then the indicator variable is first set to 1 and bound tightening is performed. A cut is then computed for this possibly tighter set F_i^1 and strengthened according to Theorem 1. In this section we evaluate the usefulness of this feature.

To this end, we introduce the setting *Full-noBT*. It is equivalent to *Full* except that the bound tightening feature is disabled. Table 12 compares settings *Full-noBT* and *Full* for the 68 *affected* instances.

Table 12 Comparison between *Full-noBT* and *Full*

	Fails	Limit	Solved	RootImpr > 50%	Time	Nodes
Full-noBT	16	153	761	4	34.45	2910
Full	17	154	759	25	33.68	2618

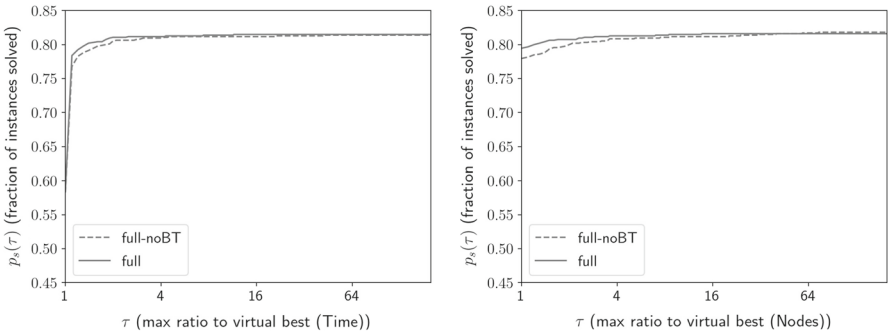


Fig. 6 Performance profiles comparing *Full-noBT* and *Full* (created with Matplotlib https://matplotlib.org/)

When bound tightening is disabled, two more instances are solved due to one less fail and one less time out. Enabling it, on the other hand, leads to large (> 50%) root node dual bounds improvements on 25 out of 68 affected instances and a comparable weakening of root node dual bounds only on 4 instances. Enabling bound tightening also yields a small decrease in the mean time (2.2%) and a moderate decrease in the number of nodes (10%). According to these results, the two settings are very close in performance, *Full-noBT* being the slightly more reliable setting and *Full* yielding smaller branch-and-bound trees and slightly better solving times.

Performance profiles comparing *Full-noBT* and *Full* are shown in Fig. 6. The curves are very close since there are few affected instances. The setting *Full* performed slightly better than *Full-noBT* both in terms of running times and tree sizes, and the two settings are nearly identical in the limit.

5.6 Comparison to perspective reformulations

Another approach that can be applied to convex constraints is to modify the problem formulation itself by applying a perspective reformulation to the constraints, as described in Sect. 2.1.3. Table 13 shows a comparison between the performance of perspective cuts and of perspective reformulations on the instances from MINLPLib for which a reformulated version is available. In particular, this includes *rsyn* (retrofit-synthesis), *squfl* (separable quadratic uncapacitated facility location) and *syn* (synthesis) instances; reformulations for *clay* instances are not included, since the semi-continuity relations present in these instances are modeled in a form not detected by our implementation.

Table 13 Comparison of perspective cuts and perspective reformulations on 310 convex instances

	Full	Reformulated	Reformulated-convex
Solved	308	253	305
Time	7.08	18.02	2.80
Relative time	1.00	2.55	0.40
Nodes	261	797	4.5
Relative nodes	1.00	3.05	0.02

Perspective reformulations of *(r)syn* instances are not SOC-representable, and the non-differentiability at 0 is alleviated by replacing the indicators z in the reformulated expressions with $\epsilon + (1 - \epsilon)z$, as proposed by Furman et al. [21]. This is equivalent to z when $z = 1$, and at $z = 0$ other constraints in the model ensure that the feasible set remains the same as in the original formulation. Furman et al. determined the best value of ϵ experimentally, and the instances from the test set we ran our experiments on use the value $1e-06$. Further details on reformulations employed in these instances and references to articles discussing these reformulations can be found in the instance descriptions provided by MINLPLib.

Reformulated versions of *squfl* instances were not included into the initial test set comprised of instances selected as described in Sect. 5.1. The reason for this is that SCIP identified reformulated constraints as second order cone constraints involving non-semi-continuous variables, thus not satisfying the conditions stated in Sect. 4.2. Reformulated versions of *(r)syn* instances were included in the initial test set, but were never among the *affected* instances, since perspective cuts were redundant.

The first column of Table 13, *Full*, shows the performance on the non-reformulated versions of the instances with perspective cuts turned on, similarly to the setting *Full* in the previous comparisons. The second column, *Reformulated*, shows the performance on the reformulated instances. The drastic worsening of performance here is to a large extent due to SCIP not recognizing the convexity of the reformulated constraints. Thus, we enabled an option which instructs SCIP to assume that all constraints in an instance are convex and reran it on the reformulated instances. The results of this run are presented in column *Reformulated-convex*. We observe a considerable decrease in the geometric mean time and branch-and-bound tree size compared to the non-reformulated instances with perspective cuts enabled.

These results suggest that there are other reductions that SCIP can perform based on the reformulated constraints, which are important for the overall performance. Solving techniques that benefit from a stronger nonlinear formulation include, for example, bound propagation and the generation of gradient cuts at a solution point of the problem's NLP relaxation. Thus we conclude that where reformulations are available, they can yield a greater performance improvement compared to only cut strengthening. However, one must be careful that the solver does not lose the view of convexity, and that explicit perspective reformulations do not lead to numerical instability. One source of numerical difficulties is the non-differentiability of such reformulations, which, however, can to a large extent be alleviated by utilizing ϵ-approximations [21].

Table 14 Comparison between *Full* and *Full+Box*

	Fails	Limit	Solved	Time	Nodes
Full	3	60	110	16.79	816
Full+Box	3	58	112	17.85	880

The second source of numerical difficulties is that explicit nonlinear reformulations may lead to incorrect solutions due to errors produced by finite-precision floating-point arithmetic. For a more detailed discussion see, for example, Section 3.1 of the paper by Bestuzheva et al. [7].

5.7 Cut strengthening for the union of a nonlinear set and a box

In a preliminary experiment, we extended our method to the more general case where the variable bounds on $\boldsymbol{y}$ are reduced to a non-singleton box when the indicator variable $z = 0$, namely:

$$F^0 = \{(x_\ell, \boldsymbol{y}, z) \mid x_\ell \in [\underline{x}_\ell^0, \overline{x}_\ell^0],\ \boldsymbol{y} \in [\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0],\ z = 0\}, \tag{13a}$$

$$F^1 = \{(x_\ell, \boldsymbol{y}, z) \mid x_\ell \geq f(\boldsymbol{y}),\ \boldsymbol{y} \in [\underline{\boldsymbol{y}}, \overline{\boldsymbol{y}}],\ z = 1\}. \tag{13b}$$

In other words, the inequality $x_\ell \geq f(\boldsymbol{y})$ is an on/off constraint which becomes inactive when $z = 0$.

In detection, we consider constraints of the original (non-extended) problem formulation with at least one semi-continuous variable with respect to some indicator z, but we no longer require that all variables appearing in nonlinear terms are semi-continuous. Instead, to test redundancy of the inequality $x_\ell \geq f(\boldsymbol{y})$, we use interval arithmetic to compute an upper bound on $f([\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0])$ and check that it does not exceed $\underline{x}_\ell^0$. If the constraint is redundant at $z = 0$, it is marked for cut strengthening.

The cut strengthening procedure is an extension of the procedure presented in Sect. 3. Given a linear cut $\phi(\boldsymbol{y}) \leq x_\ell$ that is valid for the set given by $f(\boldsymbol{y}) \leq x_\ell$, we build a strengthened cut $\phi(\boldsymbol{y}) + \alpha(1 - z) \leq x_\ell$ by choosing the largest constant α that is guaranteed to maintain the validity of the cut. To this end, we identify a vertex $(\hat{x}_\ell, \hat{\boldsymbol{y}})$ of the box $[\underline{x}_\ell^0, \overline{x}_\ell^0] \times [\underline{\boldsymbol{y}}^0, \overline{\boldsymbol{y}}^0]$ where the value of $\phi(\boldsymbol{y}) - x_\ell$ is largest and set $\alpha = \hat{x}_\ell - \phi(\hat{\boldsymbol{y}})$.

When run on 1703 MINLPLib instances, the detection algorithm identified 173 instances where suitable structures are present. On these instances, we used the setting *Full* as a baseline and compared it to the setting that enables generalized cut strengthening, which we call *Full+Box*.

We report the results in Table 14. Enabling perspective cuts for sets of the form (13) resulted in an increase of 6% in running time and an increase of 8% in the number of nodes. However, with setting *Full+Box* two more instances could be solved.

An efficient implementation of this method would require careful handling of variable bounds and their dependence on the indicator variable in order to obtain the strongest possible cuts. Based on these preliminary results, we believe that perspective cuts for sets (13) are worth further investigation.

6 Conclusion

In this paper we introduced a general method to construct perspective cuts not only for convex constraints as previously proposed in the literature, but also for nonconvex constraints for which linear underestimators are readily available. We conducted a computational study of perspective cuts for convex and nonconvex constraints. Relevant structures were detected in about 10% of MINLPLib instances. The computational results indicate that adding perspective cuts for convex constraints reduces the mean running times and tree sizes by over 20%. Adding perspective cuts for nonconvex constraints can be detrimental to performance on challenging instances and can lead to an increased amount of numerical issues, which is reflected in a small decrease in the number of solved instances. Despite this, perspective cuts for nonconvex constraints reduce the geometric mean of the number of nodes of the branch-and-bound tree by 5% and improve dual bounds at the root node.

These results indicate that perspective cuts improve performance of general-purpose solvers. However, in order to efficiently utilize perspective cuts for nonconvex structures, careful implementation and tuning is necessary.

Further, we compared perspective cuts to nonlinear perspective reformulations or their ϵ-approximations on those instances from our test set where such reformulations are available. In our experiments, applying reformulations reduced the mean solving time by 60% compared to applying perspective cuts only. Thus, when available, perspective reformulations are preferable to only perspective cuts. Furthermore, the success of nonlinear perspective reformulations indicates that even non-SOC-representable reformulations are worth further investigation despite the associated numerical difficulties.

There are several directions for future work. One such direction is extending detection algorithms. Some problems contain constraints that do not satisfy the requirements in our current implementation, but with more careful analysis of the problem structure can be revealed to be suitable for applying perspective cuts. An example of this are constraints containing expressions that are defined by non-semi-continuous variables, but are semi-continuous due to other constraints in the problem. A straightforward approach would be to test expressions for semi-continuity with respect to all binary variables present in the problem, but the computational cost of doing so may become very high. Thus, more sophisticated algorithms are required to identify such cases.

Another direction for future research is generalizing the cut strengthening method to on/off variables whose "off" domain is a non-singleton interval, as well as to more general types of on/off sets. The convex hull characterization and strong convex relaxations based on perspective functions are known for isotone or non-isotone univariate convex constraints such that the "off" domain reduces to a box defined by variable bounds [8, 27, 28], as discussed in Sect. 2.1.2. However, the exponential number of constraints describing the convex hull complicates the choice of a linear outer approximation, and, to the best of our knowledge, no compact formulation of the convex hull is known for the general case of multivariate nonconvex or even convex constraints.

In a preliminary experiment, we implemented a cut strengthening procedure that works on convex and nonconvex constraints which become redundant when an indicator variable z is set to 0. The implementation that we tested did not yield an

improvement in either time or tree sizes, but enabled us to solve two more instances. A more thorough and efficient implementation of this method may be one subject of future work. Alternatively, one could attempt to implement perspective cuts as an outer approximation of the relaxed formulation by Hijazi et al. [27, 28], since this relaxation was shown to yield very strong dual bounds.

Acknowledgements The work for this article has been conducted within the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF grant numbers 05M14ZAM, 05M20ZBM).

Funding Open Access funding enabled and organized by Projekt DEAL. The work for this article has been conducted within the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF grant numbers 05M14ZAM, 05M20ZBM).

Data availibility All data analyzed during this study are publicly available. URLs are included in this published article.

Declarations

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

Code availability The code is publicly available at https://github.com/KBestuzheva/SCIP-perspective-cuts.

References

1. Aktürk, M.S., Atamtürk, A., Gürel, S.: A strong conic quadratic reformulation for machine-job assignment with controllable processing times. Oper. Res. Lett. **37**(3), 187–191 (2009). https://doi.org/10.1016/j.orl.2008.12.009
2. Atamtürk, A., Gómez, A.: Strong formulations for quadratic optimization with M-matrices and indicator variables. Math. Program. **170**(1), 141–176 (2018). https://doi.org/10.1007/s10107-018-1301-5
3. Balas, E.: Disjunctive programming. Ann. Discrete Math. **5**, 3–51 (1979)
4. Balas, E.: Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. SIAM J. Algebr. Discrete Methods **6**(3), 466–486 (1985)
5. Barbaro, R., Ramani, R.: Generalized multiperiod MIP model for production scheduling and processing facilities selection and location. Min. Eng. **38**(2), 107–114 (1986)
6. Bestuzheva, K.: KBestuzheva/SCIP-perspective-cuts: implementation of perspective cuts in SCIP (2023). https://doi.org/10.5281/zenodo.8134526
7. Bestuzheva, K., Besançon, M., Chen, W.K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S., Matter, F., Mühmer, E., Müller, B., Pfetsch, M., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., Witzig, J.: Enabling research through the SCIP optimization Suite 8.0. ACM Trans. Math. Softw. (2023). https://doi.org/10.1145/3585516

8. Bestuzheva, K., Hijazi, H., Coffrin, C.: Convex relaxations for quadratic on/off constraints and applications to optimal transmission switching. INFORMS J. Comput. **32**(3), 682–696 (2020)
9. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib: a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**(1), 114–119 (2003). https://doi.org/10.1287/ijoc.15.1.114.15159
10. Ceria, S., Soares, J.: Convex programming for disjunctive convex optimization. Math. Program. **86**(3), 595–614 (1999). https://doi.org/10.1007/s101070050106
11. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002). https://doi.org/10.1007/s101070100263
12. Fisher, E.B., O'Neill, R.P., Ferris, M.C.: Optimal transmission switching. IEEE Trans. Power Syst. **23**(3), 1346–1355 (2008). https://doi.org/10.1109/TPWRS.2008.922256
13. Frangioni, A., Furini, F., Gentile, C.: Approximated perspective relaxations: a project and lift approach. Comput. Optim. Appl. **63**(3), 705–735 (2016). https://doi.org/10.1007/s10589-015-9787-8
14. Frangioni, A., Furini, F., Gentile, C.: Improving the approximated projected perspective reformulation by dual information. Oper. Res. Lett. **45**(5), 519–524 (2017)
15. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Math. Program. **106**(2), 225–236 (2006). https://doi.org/10.1007/s10107-005-0594-3
16. Frangioni, A., Gentile, C.: SDP diagonalizations and perspective cuts for a class of nonseparable MIQP. Oper. Res. Lett. **35**(2), 181–185 (2007)
17. Frangioni, A., Gentile, C.: A computational comparison of reformulations of the perspective relaxation: SOCP vs. cutting planes. Oper. Res. Lett. **37**(3), 206–210 (2009). https://doi.org/10.1016/j.orl.2009.02.003
18. Frangioni, A., Gentile, C., Grande, E., Pacifici, A.: Projected perspective reformulations with applications in design problems. Oper. Res. **59**(5), 1225–1232 (2011). https://doi.org/10.1287/opre.1110.0930
19. Frangioni, A., Gentile, C., Hungerford, J.: Decompositions of semidefinite matrices and the perspective reformulation of nonseparable quadratic programs. Math. Oper. Res. **45**(1), 15–33 (2020)
20. Furini, F., Traversi, E., Belotti, P., Frangioni, A., Gleixner, A., Gould, N., Liberti, L., Lodi, A., Misener, R., Mittelmann, H., et al.: QPLIB: a library of quadratic programming instances. Math. Program. Comput. **11**(2), 237–265 (2019)
21. Furman, K.C., Sawaya, N.W., Grossmann, I.E.: A computationally useful algebraic representation of nonlinear disjunctive convex sets using the perspective function. Comput. Optim. Appl. **76**(2), 589–614 (2020)
22. Gamrath, G., Anderson, D., Bestuzheva, K., Chen, W.K., Eifler, L., Gasse, M., Gemander, P., Gleixner, A., Gottwald, L., Halbig, K., et al.: The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin (2020)
23. Gómez, A.: Strong formulations for conic quadratic optimization with indicator variables. Math. Program. **188**(1), 193–226 (2021)
24. Grossmann, I.E., Lee, S.: Generalized convex disjunctive programming: Nonlinear convex hull relaxation. Comput. Optim. Appl. **26**(1), 83–100 (2003). https://doi.org/10.1023/A:1025154322278
25. Günlük, O., Linderoth, J.: Perspective reformulations of mixed integer nonlinear programs with indicator variables. Math. Program. **124**(1–2), 183–205 (2010). https://doi.org/10.1007/s10107-010-0360-z
26. Günlük, O., Linderoth, J.: Perspective reformulation and applications. In: J. Lee, S. Leyffer (eds.) Mixed Integer Nonlinear Programming, pp. 61–89. Springer, New York, NY (2012). https://doi.org/10.1007/978-1-4614-1927-3_3
27. Hijazi, H., Bonami, P., Cornuéjols, G., Ouorou, A.: Mixed integer nonlinear programs featuring "on/off" constraints: convex analysis and applications. Electron. Notes Discrete Math. **36**, 1153–1160 (2010)
28. Hijazi, H., Bonami, P., Cornuéjols, G., Ouorou, A.: Mixed-integer nonlinear programs featuring "on/off" constraints. Comput. Optim. Appl. **52**(2), 537–558 (2012)
29. 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
30. Khajavirad, A., Sahinidis, N.V.: Convex envelopes generated from finitely many compact convex sets. Math. Program. **137**(1–2), 371–408 (2013)
31. Lodi, A., Tramontani, A.: Performance variability in mixed-integer programming. In: Theory Driven by Influential Applications, pp. 1–12. INFORMS (2013). https://doi.org/10.1287/educ.2013.0112
32. Rockafellar, R.T.: Convex analysis. Princeton University Press (2015)

33. Salgado, E., Gentile, C., Liberti, L.: Perspective cuts for the ACOPF with generators. In: New Trends in Emerging Complex Real Life Problems, pp. 451–461. Springer (2018)
34. Salgado, E., Scozzari, A., Tardella, F., Liberti, L.: Alternating current optimal power flow with generator selection. In: Lee, J., Rinaldi, G., Mahjoub, A.R. (eds.) Combinatorial Optimization, pp. 364–375. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-96151-4_31
35. Stubbs, R.A., Mehrotra, S.: A branch-and-cut method for 0–1 mixed convex programming. Math. Program. **86**(3), 515–532 (1999). https://doi.org/10.1007/s101070050103
36. Tawarmalani, M., Richard, J.P.P., Chung, K.: Strong valid inequalities for orthogonal disjunctions and bilinear covering sets. Math. Program. **124**(1–2), 481–512 (2010)
37. Tawarmalani, M., Sahinidis, N.V.: Convex extensions and envelopes of lower semi-continuous functions. Math. Program. **93**(2), 247–263 (2002). https://doi.org/10.1007/s10107-002-0308-z
38. 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 (2006). https://doi.org/10.1007/s10107-004-0559-y
39. Williams, H.P.: The reformulation of two mixed integer programming problems. Math. Program. **14**(1), 325–331 (1978). https://doi.org/10.1007/BF01588974
40. Zheng, X., Sun, X., Li, D.: Improving the performance of MIQP solvers for quadratic programs with cardinality and minimum threshold constraints: A semidefinite program approach. INFORMS J. Comput. **26**(4), 690–703 (2014)