Math. Prog. Comp. (2017) 9:499–526
DOI 10.1007/s12532-017-0118-1

FULL LENGTH PAPER

# Lift-and-project cuts for convex mixed integer nonlinear programs

## Linear programming based separation and extended formulations

Mustafa R. Kılınç[1] · Jeff Linderoth[2] · James Luedtke[2]

Received: 19 April 2016 / Accepted: 3 January 2017 / Published online: 25 January 2017

**Abstract** We describe a computationally effective method for generating lift-and-project cuts for convex mixed-integer nonlinear programs (MINLPs). The method relies on solving a sequence of cut-generating linear programs and in the limit generates an inequality as strong as the lift-and-project cut that can be obtained from solving a cut-generating nonlinear program. Using this procedure, we are able to approximately optimize over the rank one lift-and-project closure for a variety of convex MINLP instances. The results indicate that lift-and-project cuts have the potential to close a significant portion of the integrality gap for convex MINLPs. In addition, we find that using this procedure within a branch-and-cut solver for convex MINLPs significantly reduces the total solution time for many instances. We also demonstrate that combining lift-and-project cuts with an extended formulation that exploits separability of convex functions yields significant improvements in both relaxation bounds and the time to calculate the relaxation. Overall, these results suggest that with an effective separation routine, like the one proposed here, lift-and-project cuts may be as effective for solving convex MINLPs as they have been for solving mixed-integer linear programs.

**Electronic supplementary material** The online version of this article (doi:10.1007/s12532-017-0118-1) contains supplementary material, which is available to authorized users.

✉ Mustafa R. Kılınç
mkilinc@cmu.edu

Jeff Linderoth
linderoth@wisc.edu

James Luedtke
jim.luedtke@wisc.edu

1 Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA, USA

2 Department of Industrial and Systems Engineering, Wisconsin Institutes of Discovery, University of Wisconsin-Madison, Madison, WI, USA

**Keywords** Lift-and-project cuts · Extended formulations · Mixed-integer nonlinear programming

**Mathematics Subject Classification** 90C11 · 90C30 · 90C57

## 1 Introduction

The focus of this work is on the effective generation of lift-and-project cuts for convex mixed-integer nonlinear programs (MINLPs). A MINLP is the optimization problem

$$\begin{aligned} z_{\text{MINLP}} = \text{minimize} \quad & c^T x \\ \text{subject to} \quad & g_j(x) \le 0 \quad \forall j \in J, \\ & x \in P, \quad x_I \in \mathbb{Z}^{|I|}, \end{aligned} \tag{MINLP}$$

where $J$ is the index set of nonlinear constraints, and $I \subseteq N \overset{\text{def}}{=} \{1, \ldots, n\}$ is the index set of discrete variables. The set $P = \{x \in \mathbb{R}^n \mid Ax \le b\}$ is a bounded polyhedral subset of $\mathbb{R}^n$. We define $g(x) : \mathbb{R}^n \to \mathbb{R}^{|J|}$ as the vector-valued function $g(x) = (g_1(x), g_2(x), \ldots, g_{|J|}(x))^T$ and $\nabla g(x)^T \in \mathbb{R}^{|J| \times n}$ as the Jacobian of $g$.

We assume the functions $g_j$ are differentiable and convex, so that by relaxing the constraints $x_I \in \mathbb{Z}^{|I|}$, a smooth convex program, the nonlinear programming (NLP) relaxation, is formed, and there exists an $M < \infty$ such that $\|\nabla g_j(x)\|_2 \le M$ and $|g_j(x)| \le M$ for all $x \in P$. In (MINLP) we also assume the objective is linear. This is without loss of generality, as a (convex) nonlinear objective function $f(x)$ can be included with the addition of an auxiliary variable $\eta$, changing the objective to minimize $\eta$ and adding the constraint $f(x) - \eta \le 0$. This step is important. Our aim is to add valid linear inequalities (cuts) that exclude the solution of a relaxation to (MINLP) from the feasible region. Without a linear objective function, the minimizer of the relaxation may lie in the strict interior of the feasible region, and thus may not be cut off using linear inequalities.

If $J = \emptyset$, then (MINLP) is a mixed integer linear program (MILP). Over the past decades, algorithms for solving MILPs have improved by orders of magnitude. A crucial ingredient in the improvement of MILP software has been cuts. The reader is referred to [22,23] for some recent surveys on cutting planes for MILP. Research on the effective generation and use of cuts for MINLP is far less advanced, although some authors have shown that many common MILP cuts can be extended for use in nonlinear settings. For example, Cezik and Iyengar [21] show how the classic Gomory cut [34] can be extended to the case of mixed integer second-order cone programs (MISOCP), which are (MINLP) where $g_j(x) = \|Hx - f\|_2 - r^T x - c \ \forall j \in J$. Atamtürk and Narayan [2] extended mixed integer rounding cuts [47] to the case of MISOCP. Modaresi et al. [45] study the generalization of intersection (and related) cuts to the realm of MINLP.

We seek a computationally efficient procedure for generating the lift-and-project cuts proposed in [20,53] for convex MINLP, which are based on extending the disjunctive cuts of [3] to the realm of convex MINLP. As in [53], we generate disjunctive cuts only for single variable disjunctions of the form $(x_i \le k \vee x_i \ge k+1)$ for some $i \in I$, and following common practice, we consequently refer to the inequalities as *lift-and-project cuts*. We present our approach for the more general case of split disjunction of the form $(\pi x \leq \pi_0 \vee \pi \geq \pi_0 + 1)$, but we only experiment with lift-and-project cuts. A limitation of the lift-and-project procedure in [53] for MINLP is that identifying a cut requires solving an auxiliary cut generating nonlinear problem (NLP) that is twice the size of the original relaxation, which is computationally expensive. Stubbs and Mehrotra [53] report computational results only on four instances, the largest of which has $n = 30$ variables. Further, they report numerical difficulties in generating the lift-and-project cuts using the separation procedure that relies on solving a cut generating (non-differentiable) NLP.

An implementation of lift-and-project cuts for the special case of MISOCP appears in the Ph.D. thesis of Drewes [27]. For general convex MINLPs, Zhu and Kuno [58] suggest to first solve the NLP relaxation of (MINLP), then build a polyhedral relaxation of the NLP relaxation by using *linearization cuts* derived using gradients of the nonlinear functions taken at the relaxation solution. Lift-and-project cuts are then derived based on this polyhedral relaxation. Using ideas from [13], Bonami [12] also proposes solving an NLP as the first step in a procedure for generating lift-and-project cuts. The optimal value of this NLP indicates whether or not a violated lift-and-project cut exists, and if so, a polyhedral relaxation is derived using the solution from this NLP, and a lift-and-project cut is generated based on this polyhedral relaxation. This method is guaranteed to identify a lift-and-project cut when one exists. In addition, since the construction of the polyhedral relaxation is done separately from the separation of the lift-and-project cut, it is straightforward to use any existing enhancements for separating lift-and-project cuts based on the polyhedral relaxation, see, e.g., [7,13,31]. All of these approaches require solving an NLP to generate a cut. In addition, the method in [58] is not guaranteed to identify a lift-and-project cut when one exists, and the approach in [13] is not guaranteed to identify a *most violated* (with respect to the normalization used) lift-and-project cut.

In this work, we introduce two new purely linear programming (LP) based procedures for generating lift-and-project cuts for convex MINLPs. The first method is a simple approach that can be interpreted as a computationally efficient adaptation of the method in [58], in which the polyhedral relaxation used within the lift-and-project separation procedure is constructed by solving a sequence of LPs, rather than by solving the NLP relaxation of (MINLP). We demonstrate that, as with the approach in [58], this simple approach may fail to find a violated lift-and-project cut when one exists. We therefore propose an iterative method that solves a sequence of cut-generating LPs, in which the lift-and-project cut obtained is improved at each iteration by strategically adding more linearization cuts to improve the polyhedral relaxation of the nonlinear constraints of (MINLP). In contrast to the methods in [12,58], we demonstrate that when separating a given solution, the cut generated from our procedure is as strong as the lift-and-project cut of [53] in the limit. A key advantage of our approach is that it produces a valid inequality at every iteration, so that we can terminate early in the event of "tailing off" in the cut generation procedure. We thus have a procedure that can effectively exploit the middle ground between our LP-based adaptation of the method of [58], which is computationally cheap but may generate weak inequalities, and the approach of [53], which can generate a most violated lift-and-project cut, but requires solving a large cut generating NLP.

An interesting feature of many of the convex MINLP test instances is that the nonlinear functions appearing in them are separable. It has been observed that when solving a MINLP using a linearization-based algorithm, it is beneficial to reformulate the problem into an *extended formulation* that exploits this separability before applying the algorithm [38,54]. The computational experiments in the Ph.D. thesis of the first author [43] show that using the extended formulation indeed yields significant improvements in the performance of all convex MINLP solvers that implement a linearization-based method, such as Outer Approximation [28] and LP/NLP-Based Branch-and-Bound algorithm [49]. Another advantage of using extended formulations is that the cuts generated in the extended space can be significantly stronger than the ones generated in the original space [10,11,46]. As our procedure for generating lift-and-project cuts is a linearization-based method, we also investigated the impact of using an extended formulation for such separable instances while using lift-and-project cuts. We find that using lift-and-project cuts in the extended formulation yields significantly improved relaxation bounds as compared to using them in the original formulation, and that these bounds can be computed significantly faster.

Another contribution of our work is to use the proposed iterative procedure for generating lift-and-project cuts to investigate the strength of the *rank one lift-and-project closure* for convex MINLP problems. This study adds to the growing literature investigating the strength of closures for various classes of cuts in mixed-integer *linear* programming [8,13,14,24,30]. In particular, the papers [13,14] demonstrated that the rank one lift-and-project closure can provide very tight relaxations for many MILP instances. Rank one lift-and-project cuts are cuts that can be obtained by a single application of the lift-and-project procedure, using only constraints (linear and nonlinear) in the original problem description. In contrast, higher rank cuts are derived by using previously-generated lift-and-project cuts. A key insight from the closure studies for MILP problems is that rank one cuts can lead to strong relaxations, and, importantly, help to avoid numerical inaccuracies that can often occur when employing higher rank inequalities. Using our proposed procedure for separating lift-and-project cuts, we find that the rank one lift-and-project closure reduces the relaxation gap by 76.5% on a set of 167 instances which do not exhibit separability. On our set of 55 instances that exhibit separability, we find that the lift-and-project closure reduces the relaxation gap by 23.3% when using the original formulation, and by 81.0% when we use the extended formulation that exploits the separability.

Encouraged by these closure results, we incorporate our lift-and-project cut separation procedure into FilMINT, a linearization-based solver for convex MINLPs [1]. FilMINT is based on solving LP relaxations constructed from using linearization cuts that outer approximate the convex continuous relaxation. In this preliminary implementation, relatively simple strategies were used to limit the computational effort expended in generating the cuts. We find that the lift-and-project cut separation procedure enabled the solution of several instances that previously could not be solved in a 2 h time limit by FilMINT and significantly reduced the solution time on many other instances. As further evidence of the impact of our proposed procedure we mention that, based on an early version of this manuscript [42] and the first author's PhD thesis

[43], a hybrid between our procedure and that of Bonami [12] is now implemented in the commercial software CPLEX for solving convex MINLPs and is reported to enable solving instances in their test set five times faster than without lift-and-project cuts [15].

The remainder of the paper is organized as follows. In Sects. 2.1 and 2.2, we review basic results on lift-and-project cuts for MILP and convex MINLP, respectively. In Sect. 3 we present the first, simple, technique for generating lift-and-project cuts and demonstrate that the approach may fail to find a violated lift-and-project cut when one exists. We describe our iterative method for generating lift-and-project cuts and prove its equivalence to the approach of [53] in Sect. 4. We review techniques for obtaining extended formulations that exploit separability in Sect. 5. In Sect. 6, we present results using our approach to obtain the lift-and-project closure and solve instances to optimality on a broad suite of convex MINLPs instances. Conclusions are offered in Sect. 7.

*Notation* For a set $X$, $\operatorname{conv}(X)$ is the convex hull of $X$, and for a polyhedral set $P \subseteq \mathbb{R}^{n+p}$, $\operatorname{proj}_x(P) = \{x \in \mathbb{R}^n \mid \exists (x, y) \in P\}$ is a projection of $P$. For a differentiable convex function $g : \mathbb{R}^n \to \mathbb{R}$ and $x \in \mathbb{R}^n$, $\nabla g(x)$ is the gradient of $g$ at $x$. We denote by $e$ an appropriate length vector of all ones, and by $e_j$ a unit vector having a one in component $j$ and zeros elsewhere.

## 2 Background

### 2.1 Lift-and-project cuts for MILP

Consider the mixed-integer set

$$X = \{x \in P \mid x_I \in \mathbb{Z}^{|I|}\},$$

where $P = \{x \in \mathbb{R}^n \mid Ax \le b\}$. If $\pi \in \mathbb{Z}^n$ satisfies $\pi_i = 0$ for all $i \notin I$, and $\pi_0 \in \mathbb{Z}$, then

$$X \subseteq P^{(\pi,\pi_0)} \overset{\text{def}}{=} \operatorname{conv}\big(\{x \in P \mid \pi x \le \pi_0\} \cup \{x \in P \mid \pi x \ge \pi_0 + 1\}\big),$$

and hence any inequality valid for $P^{(\pi,\pi_0)}$ is valid for $X$. Inequalities valid for $P^{(\pi,\pi_0)}$ are referred to as split cuts in general, and as lift-and-project cuts when they are derived from simple disjunctions of the form $\pi = e_j$ for some $j \in I$. In our computational study we focus only on lift-and-project cuts to avoid the complexity of choosing $\pi$, but we present the theory for general $\pi$. Using disjunctive programming theory of [3], given a point $\bar{x} \in P$, the separation problem for $\bar{x}$ over the set $P^{(\pi,\pi_0)}$ can be solved with the following primal cut generating problem:

$$\begin{aligned}
\operatorname{PC}(P, \pi, \bar{x}) : \quad & \text{minimize} \quad \|x - \bar{x}\| \\
& \text{subject to} \quad Ay - \lambda b \le 0, \qquad Az - \mu b \le 0, \\
& \qquad\qquad\quad \pi y - \lambda \pi_0 \le 0, \quad -\pi z + \mu(\pi_0 + 1) \le 0,
\end{aligned}$$

$$\begin{aligned} y+z&=x, & \lambda+\mu&=1,\\ \lambda&\geq 0, & \mu&\geq 0, \end{aligned}$$

where $\|\cdot\|$ is a norm, often taken to be either $\|\cdot\|_1$ or $\|\cdot\|_\infty$, in which case the problem can be formulated as a linear program. We suppress the dependence on $\pi_0$ in the notation $\mathrm{PC}(P,\pi,\bar{x})$ because for a given $\pi$ and $\bar{x}$ the value of $\pi_0$ is $\lfloor \pi^T\bar{x}\rfloor$. The dual of $\mathrm{PC}(P,\pi,\bar{x})$ is the problem:

$$\begin{aligned} \mathrm{DC}(P,\pi,\bar{x}):\quad \text{maximize}\quad & \alpha^T\bar{x}-\beta \\ \text{subject to}\quad & u^TA+u_0\pi-\alpha=0, & v^TA-v_0\pi-\alpha=0,\\ & u^Tb+u_0\pi_0-\beta\leq 0, & v^Tb-v_0(\pi_0+1)-\beta\leq 0,\\ & u\geq 0,\ u_0\geq 0, & v\geq 0,\ v_0\geq 0,\\ & \|\alpha\|_*\leq 1, \end{aligned}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$. Problem $\mathrm{PC}(P,\pi,\bar{x})$ finds a point $x\in P^{(\pi,\pi_0)}$ that minimizes the distance to $\bar{x}$, where distance is measured using the norm $\|\cdot\|$, whereas $\mathrm{DC}(P,\pi,\bar{x})$ finds an inequality valid for $P^{(\pi,\pi_0)}$ of the form $\alpha x\leq\beta$ that maximizes violation of $\bar{x}$, subject to the normalization condition that $\|\alpha\|_*\leq 1$.

In the MILP literature it has also been observed that the following alternative primal separation problem can be solved to determine if $\bar{x}\in P^{(\pi,\pi_0)}$:

$$\begin{aligned} \mathrm{PC}'(P,\pi,\bar{x}):\quad \text{minimize}\quad & \eta \\ \text{subject to}\quad & Ay-\lambda b-\eta e\leq 0, & Az-\mu b-\eta e\leq 0,\\ & \pi y-\lambda\pi_0-\eta\leq 0, & -\pi z+\mu(\pi_0+1)-\eta\leq 0,\\ & y+z=\bar{x}, & \lambda+\mu=1,\\ & \lambda\geq 0, & \mu\geq 0, \end{aligned}$$

The dual of this problem, denoted $\mathrm{DC}'(P,\pi,\bar{x})$, is identical to $\mathrm{DC}(P,\pi,\bar{x})$ with the exception that the normalization $\|\alpha\|_*\leq 1$ is replaced by the *standard normalization condition* (SNC) [4]:

$$u^Te+v^Te+u_0+v_0=1.$$

Computational experiments have demonstrated that using the SNC yields better cuts than using other normalizations. See [31] for an interesting study providing possible explanations for this phenomenon.

After obtaining a split cut using one of the above problems, it can be strengthened using the integrality of variables, a technique introduced by [6] and known as *monoidal strengthening*. This can be done for any row of the constraints $Ax\leq b$ such that the slack $(b-Ax)_i$ must be integral for any $x\in X$. We describe the details for a row in $Ax\leq b$ corresponding to non-negativity of an integer decision variable, i.e., $-x_k\leq 0$ for some $k\in I$. Let $(\alpha,\beta,u,v,u_o,v_0)$ be a solution to $\mathrm{DC}(P,\pi,\bar{x})$ and let $u_{x_k}$ and

$v_{x_k}$ be the value of dual variables corresponding to the constraint $-x_k \leq 0$. Then, the coefficient of $x_k$ in the inequality $\alpha x \leq \beta$ can be replaced by

$$\tilde{\alpha}_k = \alpha_k + \max\{u_{x_k} + \lfloor m \rfloor u_0, v_{x_k} - \lceil m \rceil v_0\}$$

where $m = (v_{x_k} - u_{x_k})/(u_0 + v_0)$. Following [7,13], when monoidal strengthening is applied to a lift-and-project cut, the resulting cut is referred to as a strengthened lift-and-project cut.

### 2.2 Lift-and-project cuts for MINLPs

We now review the basic theory on lift-and-project cuts for convex MINLPs, based on [53]. In this section, we further assume that variables are bounded below and above with finite bounds of the form $l \leq x \leq u$ and these constraints are part of $Ax \leq b$ defining $P$. We denote the feasible region of the continuous relaxation of (MINLP) as

$$R = \{x \in P \mid g(x) \leq 0\}.$$

Given $\pi \in \mathbb{Z}^n$ and $\pi_0 \in \mathbb{Z}$ in which $\pi_i = 0$ for all $i \notin I$, the set

$$R^{(\pi,\pi_0)} \overset{\text{def}}{=} \operatorname{conv}\big(\{x \in R \mid \pi x \leq \pi_0\} \cup \{x \in R \mid \pi x \geq \pi_0 + 1\}\big)$$

is a relaxation of the feasible region to (MINLP), and hence valid inequalities for $R^{(\pi,\pi_0)}$ are also valid for (MINLP).

A key result in [53] is that there is an extended formulation of $R^{(\pi,\pi_0)}$ in terms of convex, nonlinear, inequalities using a variable transformation and a new function related to a given convex function $h(x) : C \subset \mathbb{R}^n \to \mathbb{R}$ as follows:

$$\tilde{h}(x,\lambda) = \begin{cases} \lambda h(x/\lambda) & \text{if } x/\lambda \in C, \lambda > 0, \\ 0 & \text{if } x = 0, \lambda = 0. \end{cases}.$$

$\tilde{h}(x,\lambda)$ corresponds to the perspective function of $h$ in the case $\lambda > 0$ and $x/\lambda \in C$ [39]. Note that $\tilde{h}(x,\lambda)$ may be a non-differentiable function, even if $h$ itself is differentiable.

It has been shown in [53] that the convex set

$$\mathcal{M}^{(\pi,\pi_0)} = \left\{ (x,y,z,\lambda,\mu) \;\middle|\; \begin{array}{ll} y + z = x, & \lambda + \mu = 1, \\ \tilde{g}(y,\lambda) \leq 0, & \tilde{g}(z,\mu) \leq 0 \\ Ay \leq \lambda b, & Az \leq \mu b, \\ \pi y \leq \lambda \pi_0, & -\pi z \leq -\mu(\pi_0 + 1), \\ \lambda \geq 0, & \mu \geq 0 \end{array} \right\} \tag{1}$$

provides an extended formulation for $R^{(\pi,\pi_0)}$. In other words, $\operatorname{proj}_x(\mathcal{M}^{(\pi,\pi_0)}) = R^{(\pi,\pi_0)}$. Therefore, given a point $\bar{x} \notin R^{(\pi,\pi_0)}$, a linear inequality separating $\bar{x}$ from

$R^{(\pi,\pi_0)}$ can be found by minimizing the distance $d(x) = \|x - \bar{x}\|$ between the point and the set, in any norm $\|\cdot\|$. Specifically, consider the minimization problem

$$d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x}) = \min_{(x,\tilde{y},\tilde{z},\lambda,\mu)\in\mathcal{M}^{(\pi,\pi_0)}} d(x). \tag{2}$$

The following theorem states that a lift-and-project cut may be obtained using a subgradient of $d(\cdot)$ at an optimal solution to (2).

**Theorem 1** ([53]) *Let $\bar{x} \notin R^{(\pi,\pi_0)}$, $x^*$ be an optimal solution of (2). Then there exists a subgradient $\xi$ of $d$ at $x^*$ such that $\xi^T(\bar{x} - x^*) < 0$ and $\xi^T(x - x^*) \geq 0\ \forall x \in R^{(\pi,\pi_0)}$.*

There are two disadvantages of using (2) directly to generate cuts. First, one must solve a nonlinear program that is twice the size of the original problem in order to generate a valid inequality. Second, the description of the set $\mathcal{M}^{(\pi,\pi_0)}$ contains non-differentiable functions, leading to possible numerical difficulties when using nonlinear programming software designed for differentiable functions.

## 3 A simple LP-based lift-and-project cut generation strategy for MINLP

A simple strategy for generating lift-and-project cuts for a MINLP problem is to solve a CGLP, exactly as would be done for a MILP, based on a given polyhedral outer approximation of the relaxed feasible region. Specifically, given a finite set of points $\mathcal{K} \stackrel{\text{def}}{=} \{\bar{x}^1, \ldots, \bar{x}^K\} \subseteq P$, the *linearization cuts*:

$$g(\bar{x}) + \nabla g(\bar{x})^T(x - \bar{x}) \leq 0, \quad \bar{x} \in \mathcal{K} \tag{3}$$

define a polyhedral relaxation of $R$. For the finite set of points $\mathcal{K}$, define the polyhedral set

$$Q(\mathcal{K}) \stackrel{\text{def}}{=} \{x \in \mathbb{R}^n \mid Ax \leq b,\ \nabla g(\bar{x}^k)^T x \leq \nabla g(\bar{x}^k)^T \bar{x}^k - g(\bar{x}^k),\ k = 1, \ldots, K\}.$$

Clearly, $Q(\mathcal{K}) \supseteq R$, and hence the cut generating problems $\mathrm{PC}(Q(\mathcal{K}), \pi, \bar{x})$ or $\mathrm{PC}'(Q(\mathcal{K}), \pi, \bar{x})$ can be used to generate valid lift-and-project cuts for $R^{(\pi,\pi_0)}$.

The key question to be answered in using this strategy is which points $\mathcal{K}$ to use to define the polyhedral relaxation. Zhu and Kuno [58] proposed to use the linearization cuts from a single point: the optimal solution of the NLP relaxation, where the NLP relaxation includes any previous cuts that have been generated. This approach is computationally expensive, however, as it requires solving an NLP before generating each cut. A simple alternative to approximate this strategy is to alternate between adding violated linearization cuts of the form (3) and solving a CGLP to generate lift-and-project cuts, where the outer approximation used in the CGLP is defined by all linearization cuts added up to that point. This strategy is depicted in Fig. 1, where

$$Q_C(\mathcal{K}) \stackrel{\text{def}}{=} \{x \in Q(\mathcal{K}) \mid \alpha^c x \leq \beta^c,\ c = 1, \ldots, |C|\} \tag{4}$$

$$\begin{tikzcd} Q_C(\mathcal{K}) \arrow[loop above, "\text{Linearizations}"] \arrow[r, bend left, "Q(\mathcal{K})"] & \text{CGLP} \arrow[l, bend left, "\text{Cuts}"] \end{tikzcd}$$

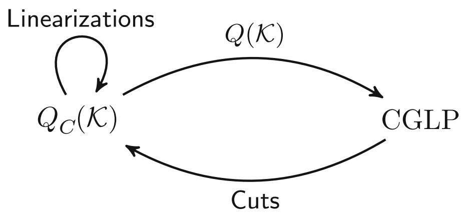
**Fig. 1** LP-based cut generation strategy for MINLP

is the polyhedral relaxation defined by including both the linearization cuts and the set of accumulated lift-and-project cuts, $C$.

We next show an example that demonstrates that this simple approach may fail to separate a rank one lift-and-project cut, even if one exists. In other words, the cuts generated may not be as strong as the lift-and-project cuts that can be obtained by solving (2). Note that if one is willing to use higher rank lift-and-project cuts, i.e., by including the previously generated cuts within the lift-and-project CGLP, then it is always possible to generate a violated inequality that cuts off the current relaxation solution. This was observed in [58] for the case of convex MINLP. However, it has been observed within the MILP literature that it is preferable to restrict attention to low-rank lift-and-project cuts as these avoid potential numerical instabilities, and have been shown to provide strong relaxation bounds [8,30,31].

*Example 1* Consider the MINLP instance:

$$\begin{aligned} \text{maximize } & x_1 + x_2 && \text{(5a)}\\ \text{subject to: } & 7x_1 + 8x_2 \le 9 && \text{(5b)}\\ & 8x_1 + 7x_2 \le 9 && \text{(5c)}\\ & x_1^2 + x_2^2 \le (9/10)^2 && \text{(5d)}\\ & x_1, x_2 \in \{0, 1\}. && \text{(5e)} \end{aligned}$$

The only feasible solution to this instance is $(0, 0)$; the feasible region of the continuous relaxation is depicted in Fig. 2. The optimal NLP relaxation solution is $\bar{x}_R = (3/5, 3/5)$. As this solution is strictly feasible to the nonlinear constraint (5d), there are no violated linearization cuts that can be added, and hence the problems $\text{PC}(P, e_j, \bar{x}_R)$, $j = 1, 2$ for generating lift-and-project cuts would be based on the outer approximation defined by (5b), (5c) and the bounds on the variables. Using $\|\cdot\|_\infty$ as the norm in the objective of $\text{PC}(P, e_j, \bar{x}_R)$, and generating a single lift-and-project cut for each $j = 1, 2$ yields the following cuts:

$$6x_1 + 7x_2 \le 7, \quad 7x_1 + 6x_2 \le 7.$$

The right picture in Fig. 2 shows the updated continuous relaxation with these cuts added (the dashed lines). The optimal solution to this relaxation is $\bar{x}_D = (7/13, 7/13)$. This solution again satisfies (5d) strictly, so that no linearization cuts from (5d) can be added to cut it off. Let $P = \{(x_1, x_2) \in [0, 1] \times [0, 1] \mid (5b), (5c)\}$ be the polyhedral outer-approximation, and let $P_i^0 = P \cap \{x_i = 0\}$, and $P_i^1 = P \cap \{x_i = 1\}$, for $i = 1, 2$.

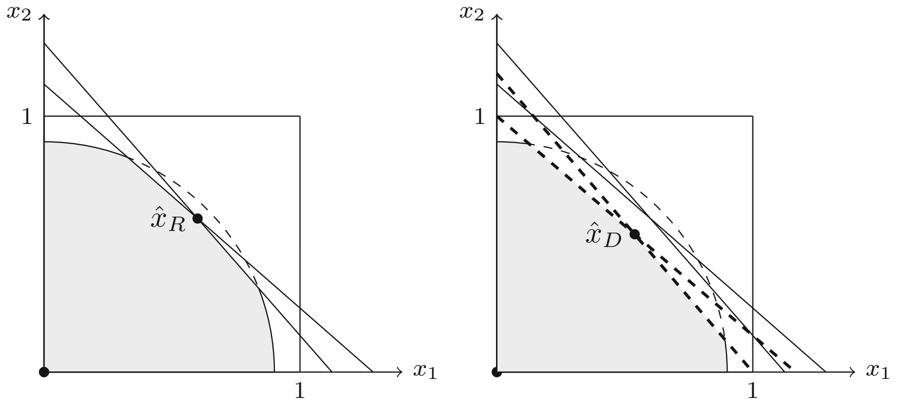

**Fig. 2** Continuous relaxation of the instance in example 1 (*left*) and with lift-and-project cuts (*right*)

This solution satisfies: $\bar{x}_D \in \text{conv}(P_i^0 \cup P_i^1)$ for $i = 1, 2$, and hence cannot be cut off by a lift-and-project cut using this procedure.

On the other hand, the only solution to the continuous relaxation of (5) in which $x_1 = 1$ has $x_2 = 0$, and hence $x_2 \leq 0$ is a valid rank one lift-and-project cut. Similarly, $x_1 \leq 0$ is a valid rank one lift-and-project cut. Thus, in this example the rank one lift-and-project cuts are sufficient to define the convex hull of the feasible region. Note that if the linearization cuts $x_1 \leq 9/10$ and $x_2 \leq 9/10$ derived from (5d) were included in the outer approximation, then the cuts $x_1 \leq 0$ and $x_2 \leq 0$ could have been derived from the CGLP.

This example illustrates the key limitation of this simple approach: the ability to generate lift-and-project cuts based on a polyhedral outer approximation is limited by the linearization cuts that are used. In particular, using the linearization cuts available from the solution of the NLP relaxation solution is not sufficient to guarantee that a rank one lift-and-project cut will be found if one exists. The procedure of [12] overcomes this limitation by first solving a nonlinear program to find the right points to obtain the linearization cuts from. In the next section, we present a purely LP-based procedure that overcomes this limitation by dynamically updating the set of linearization cuts.

## 4 An LP-based strategy that converges to the best possible lift-and-project cut

We propose to approximately solve the nonlinear program (2) by solving a sequence of cut-generating linear programs in which the set of linearization cuts used to approximate $R$ is strategically updated. This algorithm is essentially a variant of the classical cutting plane algorithm of Kelley [41], with a simple but important modification to deal with the non-differentiable functions used in the definition of $\mathcal{M}^{(\pi,\pi_0)}$.

The algorithm uses different sets of linearization cuts to approximate the two sets in the disjunction, $R^1 = \{x \in R \mid \pi^T x \leq \pi_0\}$ and $R^2 = \{x \in R \mid \pi^T x \geq \pi_0 + 1\}$. Thus, the algorithm solves a cut-generating linear program of the form:

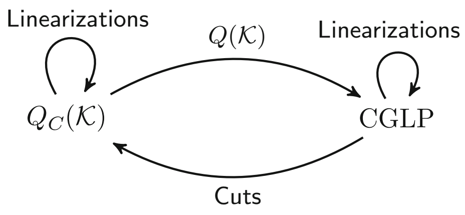

**Fig. 3** Iterative lift-and-project cut generation strategy

$$
\begin{aligned}
&\mathrm{PC}(P^1, P^2, \pi, \bar{x}):\\
&\quad \text{minimize} && \|x-\bar{x}\| \\
&\quad \text{subject to} && A^1 y - \lambda b^1 \le 0, && A^2 z - \mu b^2 \le 0,\\
& && \pi y - \lambda \pi_0 \le 0, && -\pi z + \mu(\pi_0+1) \le 0,\\
& && y + z = x, && \lambda + \mu = 1,\\
& && \lambda \ge 0, && \mu \ge 0
\end{aligned}
$$

where $P^k = \{x \in \mathbb{R}^n \mid A^k x \le b^k\}$, $k = 1, 2$ are polyhedra with the given explicit inequality descriptions. The other CGLP variant discussed in Sect. 2.1, $\mathrm{PC}'(P, \pi, \bar{x})$, is extended similarly.

Our LP-based algorithm for generating a lift-and-project cut is given in Algorithm 1 and illustrated in Fig. 3. In the algorithm, $\mathcal{K}^1$ and $\mathcal{K}^2$ represent the set of points at which linearization cuts are taken to approximate the sets $R^1$ and $R^2$, respectively. In this simple version of the algorithm, these sets are initialized to be empty, but in a more sophisticated implementation, they would be initialized with points used previously in the branch-and-cut algorithm. The key feature of the algorithm is how these sets are updated after obtaining a solution $(x^t, y^t, z^t, \lambda^t, \mu^t)$ to $\mathrm{PC}(P^1, P^2, \pi, \bar{x})$. An intuitive strategy might be to add linearization cuts based on the point $x^t$, which is the closest point to $\bar{x}$ using the current outer-approximation. However, to ensure convergence of the algorithm, it is necessary to use the "unscaled" points of $y^t$ and $z^t$ solutions. Specifically, we add linearization cuts based on the points $y^t/\lambda^t$ and $z^t/\mu^t$ (when $\lambda^t, \mu^t \neq 0$, respectively). These points have been used previously for a different purpose (selecting alternative disjunctions) in [25], where they are referred to as the "friends" of $x^t$.

The quality of the cut generated by Algorithm 1 may be measured by the objective value of the cut-generating linear program, which is the distance between $\bar{x}$ and the current outer approximation of the set. Using this measure, the following theorem states that, in the limit, the cut generated by Algorithm 1 is as strong as the lift-and-project cut that would be obtained using the method in [53].

**Theorem 2** *Assume $R_1 \cup R_2 \neq \emptyset$, $P$ is bounded, and there exists an $M < \infty$ such that $\|\nabla g_j(x)\|_2 \le M$ and $|g_j(x)| \le M$ for all $x \in P$. Then, if Algorithm 1 stops at iteration $t$, then*

$$\|x^t - \bar{x}\| = d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x}).$$

```
$\mathcal{K}^1 \leftarrow \emptyset, \mathcal{K}^2 \leftarrow \emptyset, t \leftarrow 1$
for $t = 1, 2, \ldots$ do
  Solve PC($Q(\mathcal{K}^1), Q(\mathcal{K}^2), \pi, \bar{x}$) and let $(x^t, y^t, z^t, \lambda^t, \mu^t)$ be an optimal solution.
  if ($\lambda^t = 0$ or $g(y^t/\lambda^t) \le 0$) and ($\mu^t = 0$ or $g(z^t/\mu^t) \le 0$)) then
    STOP ($x^t \in R^{(\pi,\pi_0)}$).
  end if
  if $\lambda^t \neq 0$ then
    $\mathcal{K}^1 \leftarrow \{y^t/\lambda^t\} \cup \mathcal{K}^1$
  end if
  if $\mu^t \neq 0$ then
    $\mathcal{K}^2 \leftarrow \{z^t/\mu^t\} \cup \mathcal{K}^2$
  end if
end for
```

**Algorithm 1:** Iterative solution of (2).

*Otherwise,*

$$\lim_{t\to\infty} \|x^t - \bar{x}\| = d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x}).$$

*Proof* For each $t \ge 1$, let $d_t(\bar{x}) = \|x^t - \bar{x}\|$, which is the optimal objective value of PC($Q(\mathcal{K}^1), Q(\mathcal{K}^2), \pi, \bar{x}$) in iteration $t$. Because $R \subseteq Q(\mathcal{K}^k)$, $k = 1, 2$ in each iteration, it follows that $\mathcal{M}^{(\pi,\pi_0)}$ is a subset of the feasible region of PC($Q(\mathcal{K}^1), Q(\mathcal{K}^2), \pi, \bar{x}$), and hence $d_t(\bar{x}) \le d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x})$ for all $t$.

Now, if the algorithm terminates at iteration $t$, then if both $\lambda^t > 0$ and $\mu^t > 0$, then $x^t = \lambda^t(y^t/\lambda^t) + \mu^t(z^t/\mu^t)$, which shows $x^t \in R^{(\pi,\pi_0)}$ because $g(y^t/\lambda^t) \le 0$ and $g(z^t/\mu^t) \le 0$ by the termination condition. Similarly, $x^t \in R^{(\pi,\pi_0)}$ if either $\lambda^t = 0$ or $\mu^t = 0$. Since $x^t \in R^{(\pi,\pi_0)}$ it follows that $d_t(\bar{x}) \ge d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x})$, implying the result.

Now assume the algorithm never terminates. By construction, $\{d_t(\bar{x})\}$ forms a monotonically increasing sequence, $d_1(\bar{x}) \le d_2(\bar{x}) \le \ldots$, bounded above by $d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x})$. Therefore, the sequence converges. The proof continues by showing there exists a subsequence of $\{x^t, t \ge 1\}$ that converges to a point in $R^{(\pi,\pi_0)}$, from which it follows that $\{d_t(\bar{x})\}$ converges to $d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x})$.

Let $\mathcal{T}_\lambda = \{t \ge 1 : \lambda^t > 0\}$ and $\mathcal{T}_\mu = \{t \ge 1 : \mu^t > 0\}$. For $t \in \mathcal{T}_\lambda$, define $\bar{y}^t = y^t/\lambda^t$, and for $t \in \mathcal{T}_\mu$, define $\bar{z}^t = z^t/\mu^t$. If $\mathcal{T}_\lambda$ is infinite, then arguments of Kelley [41] demonstrate that there is a subsequence of $\{\bar{y}^t : t \in \mathcal{T}_\lambda\}$ which converges to a point $\bar{y} \in R^1$. By an identical argument it also follows that if $\mathcal{T}_\mu$ is infinite, then there is a subsequence of $\{\bar{z}^t : t \in \mathcal{T}_\mu\}$ which converges to a point $\bar{z} \in R^2$.

Now, let $\mathcal{T}_+ = \mathcal{T}_\lambda \cap \mathcal{T}_\mu$. We consider two cases, depending on whether or not $\mathcal{T}_+$ is infinite.

*Case 1 $\mathcal{T}_+$ is infinite* Using the arguments above, there exists a subsequence of $\mathcal{T}' \subseteq \mathcal{T}_+$ such that $\{(x^t, \bar{y}^t, \bar{z}^t, \lambda^t, \mu^t) : t \in \mathcal{T}'\}$ converges to a point $(\bar{x}, \bar{y}, \bar{z}, \bar{\lambda}, \bar{\mu})$ in which $\bar{y} \in R^1$ and $\bar{z} \in R^2$. Because $x^t = \lambda^t \bar{y}^t + \mu^t \bar{z}^t$ and $\lambda^t + \mu^t = 1$ for all $t \in \mathcal{T}'$, it also follows that $\bar{x} = \bar{\lambda}\bar{y} + \bar{\mu}\bar{z}$ and $\bar{\lambda} + \bar{\mu} = 1$. Thus, $\bar{x} \in \mathcal{M}^{(\pi,\pi_0)}$, and so $\{d_t(\bar{x}) : t \in \mathcal{T}'\}$ converges to $d_{\mathcal{M}^{(\pi,\pi_0)}}(\bar{x})$. It follows that the entire convergent sequence $\{d_t(\bar{x}) : t \ge 1\}$ converges to the same.

*Case 2 $\mathcal{T}_+$ is finite* This case consists of two symmetric subcases: either $\mathcal{T}_\lambda$ contains an infinite subsequence $\mathcal{T}_\lambda^1$ in which $\lambda_t = 1$ for all $t \in \mathcal{T}_\lambda^1$, or $\mathcal{T}_\mu$ contains an infinite subsequence $\mathcal{T}^1_\mu$ in which $\mu_t = 1$ for all $t \in \mathcal{T}^1_\mu$. We analyze only the first of these symmetric cases. So suppose the infinite subsequence $\mathcal{T}^1_\lambda$ exists. Again using the arguments above, there exists a subsequence $\mathcal{T}'$ of $\mathcal{T}^1_\lambda$ such that $\{\bar{y}^t : t \in \mathcal{T}'\}$ converges to a point $\bar{y} \in R^1$. But now, because $\lambda^t = 1$ for all $t \in \mathcal{T}'$ it immediately follows that $\{x^t : t \in \mathcal{T}'\}$ also converges to $\bar{y} \in R^1 \subseteq \mathcal{M}^{(\pi,\pi_0)}$. The remaining argument is identical to case 1. □

A consequence of Theorem 2 is that if $\bar{x} \notin R^{(\pi,\pi_0)}$, then the algorithm will find a valid inequality for $R^{(\pi,\pi_0)}$ that cuts off $\bar{x}$ in finitely many iterations. Furthermore, in the limit, the valid inequality will be supported by a point in $R^{(\pi,\pi_0)}$.

We now revisit Example 1 from Sect. 3, and demonstrate that Algorithm 1 successfully obtains the best rank one lift-and-project cuts.

*Example 2 (Example 1, continued.)* Once again, we begin with the initial relaxation solution $\bar{x}_R = (3/5, 3/5)$. Using the disjunction $x_2 \leq 0 \vee x_2 \geq 1$, we solve problem $\text{PC}(P^1, P^2, \pi, \bar{x})$ using $\|\cdot\|_\infty$ as the norm in the objective. The optimal optimal solution is $y = (6/13, 0)$, $\lambda = 6/13$, $z = (1/13, 7/13)$, $\mu = 7/13$, and the valid inequality (obtained from the dual) is $7x_1 + 6x_2 \leq 7$. Following Algorithm 1, we compute $y/\lambda = (1, 0)$ and $z/\mu = (1/7, 1)$ and add these points to the sets $\mathcal{K}_1$ and $\mathcal{K}_2$, respectively. This corresponds to adding the inequality $2x_1 \leq 1.81$ to the description of $Q(\mathcal{K}_1)$ and the inequality $(2/7)x_1 + 2x_2 \leq (1/7)^2 + 1 + (9/10)^2 \approx 1.830408$ to the description of $Q(\mathcal{K}_2)$. The updated problem $\text{PC}(P^1, P^2, \pi, \bar{x})$ is then solved again, and in this case the solution is $y = (0, 0), \lambda = 1, z = (0, 0), \mu = 0$, and the resulting inequality is $x_2 \leq 0$. It is easy to check that this inequality is valid for the disjunction with these linearization cuts added, since for the case $x_2 \geq 1$, it holds that $(2/7)x_1 + 2x_2 \geq 2$, and so that term of the disjunction is empty after adding the inequality $(2/7)x_1 + 2x_2 \leq 1.830408$. Therefore, the inequality $x_2 \leq 0$ is valid for the disjunctive term $x_2 \geq 1$, and of course, is also valid for the term with $x_2 \leq 0$. Thus, in this example, updating the polyhedral approximation with a single linearization inequality is sufficient to yield the best possible lift-and-project cut. A symmetric result occurs when a cut is generated using the disjunction $x_1 \leq 0 \vee x_1 \geq 1$.

Algorithm 1 can be modified to use the $\text{PC}'(P, \pi, \bar{x})$ variant of of the CGLP (i.e., to use the standard normalization condition) in place of the problem $\text{PC}(Q(\mathcal{K}^1), Q(\mathcal{K}^2), \pi, \bar{x})$. However, in the case where the functions $g_j(x)$, $j \in J$ are known only to be convex over $P$, since $\text{PC}'(P, \pi, \bar{x})$ may relax constraints of $P$, care must be taken to avoid generating linearization cuts at points outside the domain where the nonlinear functions are known to be convex. In particular, a subset of the constraints defining $P$, say $Dx \leq d$ must be chosen such that the functions $g_j(x)$, $j \in J$ are convex over the region $\{x \in \mathbb{R}^n \mid Dx \leq d\}$. For example, if $P \subseteq \mathbb{R}^n_+$, and the functions $g_j : j \in J$ are convex over $\mathbb{R}_+$, then we may take $D = -I$ and $d = 0$. We then define the following CGLP:

$$
\begin{aligned}
\text{PC}'(P^1, P^2, \pi, \bar{x}) : \quad & \text{minimize} && \eta \\
& \text{subject to } A^1 y - \lambda b^1 - \eta e \leq 0, && A^2 z - \mu b^2 - \eta e \leq 0, \\
& \qquad\qquad Dy - \lambda d \leq 0, && Dz - \mu d \leq 0,
\end{aligned}
$$

$$\begin{aligned} \pi y - \lambda\pi_0 - \eta \leq 0, \quad & -\pi z + \mu(\pi_0 + 1) - \eta \leq 0, \\ y + z = \bar{x}, \quad & \lambda + \mu = 1, \\ \lambda \geq 0, \quad & \mu \geq 0, \end{aligned}$$

where again $P^k = \{x \in \mathbb{R}^n \mid A^k x \leq b^k\}$, $k = 1, 2$. In $\mathrm{PC}'(P^1, P^2, \pi, \bar{x})$, the constraints $A^1 y \leq \lambda b^1$ and $A^2 z \leq \mu b^2$ are relaxed using the objective variable $\eta$, whereas the constraints $Dy \leq \lambda d$ and $Dz \leq \mu d$ are not relaxed. This ensures that in Algorithm 1, linearization cuts are only derived at points in the region for which the functions $g_j$, $j \in J$ are convex. To ensure $\mathrm{PC}'(P^1, P^2, \pi, \bar{x})$ is feasible, $D, d$ should be chosen such that $\bar{x}$ can be written as a convex combination of points that satisfy $Dx \leq d$ and either $\pi x \leq \pi_0$ or $\pi x \geq \pi_0 + 1$.

## 5 Extended formulations via separability

It has been observed that using an extended formulation derived from exploiting separability of an instance can significantly improve the performance of linearization-based methods for convex MINLP ([38,43,54,57]). Since we are using a linearization-based approach for generating lift-and-project cuts, we also explore the use of such extended formulations when using our approach for generating lift-and-project cuts. We therefore describe here approaches for deriving such extended formulations for convex MINLP problems where the constraint functions are *convex-separable*.

A convex function $h : \mathbb{R}^n \to \mathbb{R}$ is called *convex-separable* if it can be written as

$$h(x) = \sum_{i=1}^{p} h_i(x).$$

where each of the $p$ functions $h_i : \mathbb{R}^n \to \mathbb{R}$ are convex. While the functions $h_i$ may in general be functions of $n$ variables, we are usually interested in cases where the $h_i$ functions are univariate or depend on a small number of variables.

Consider a convex MINLP problem of the form:

$$\begin{aligned} z_{\mathrm{MINLP}} = \min \ & c^T x \\ \text{s.t.} \ & \sum_{p=1}^{p_j} g_{jp}(x) \leq 0 \quad \forall j \in J, \\ & x \in X, \quad x_I \in \mathbb{Z}^{|I|}, \end{aligned} \tag{MINLP}$$

where $g_j(x) := \sum_{p=1}^{p_j} g_{jp}(x)$ is convex-separable $\forall j \in J$.

The convex-separability property of the problem (MINLP) can be exploited by introducing variables $y_{jp}$ to represent $g_{jp}(x)$, for $j \in J$, $p = 1, \ldots, p_j$, leading to the extended formulation:

$$\begin{aligned} z_{\mathrm{MINLP}} = \min \ & c^T x \\ \text{s.t.} \ & g_{jp}(x) \leq y_{jp} \quad \forall j \in J \quad \forall p \in \{1, \ldots, p_j\}, \end{aligned} \tag{Ext-MINLP}$$

$$\sum_{p=1}^{p_j} y_{jp} \le 0 \quad \forall j \in J,$$

$$x \in X, \quad x_I \in \mathbb{Z}^{|I|}.$$

(Ext-MINLP) is a convex MINLP problem since the functions $g_{jp}$ are convex. Tawarmalani and Sahinidis [54] have shown that the outer approximation for (Ext-MINLP) is a tighter relaxation than the outer approximation for (MINLP) derived using linearization cuts at the same set of linearization points.

In some cases, a function $h$ may not be convex-separable, but separability can be induced using an affine change of variables. For example, consider the convex quadratic function $h : \mathbb{R}^n \to \mathbb{R}$

$$h(x) = x^T Qx$$

where $Q$ is $n \times n$ symmetric positive-definite matrix. By applying a Cholesky decomposition, $Q$ can be decomposed as $Q = LL^T$ where $L$ is lower triangular matrix. Using this decomposition $h$ can be written as $h(x) = x^T LL^T x$. Thus, if new variables $y$ are introduced with the constraints $y = L^T x$, then $g(y) = y^T y = h(x)$, and thus we can use the separable function $g(y)$ in place of $h(x)$ in the formulation.

## 6 Computational experiments

In this section, we investigate the computational impact of the lift-and-project cuts described in Sects. 3 and 4 (and their strengthened counterpart described in Sect. 2.1) and the effectiveness of combining the cuts with the extended formulations reviewed in Sect. 5.

### 6.1 Test sets

The strength of (strengthened) lift-and-project cuts are tested on a suite of 222 convex MINLPs intances covering a wide range of applications such as multi-product batch plant design problems (`Batch`) [50,56], layout design problems (`CLay, FLay, SLay, fo-m-o`) [19,51], synthesis design problems(`Syn`) [28,55], retrofit planning (`Rsyn`) [51], stochastic service system design problems(`sssd`) [29], cutting stock problems (`trimloss, tls`) [37], quadratic uncapacitated facility location problems(`uflquad`) [35], network design problems (`nd`) [9,16,17] and portfolio optimization (`MV`) [33]. The test is set collected from the MacMINLP collection [44], GAMS MINLP World [18], the collection on the website of the IBM-CMU research group [52] and instances we created ourselves. The instances we created are the instance families `sssd`, `uflquad`, `nd` and `MV` and they are generated as they described in [36].

In order to test extended formulations, we examined instances to determine if they contain convex-separable functions. The instance families that contain convex-separable functions were `Batch`, `MV`, `SLay`, `trimloss` and `uflquad`, and they are reformulated as described in Sect. 5. All convex-separable functions in these instances

**Table 1** Characteristics of separable instances

| Instance family | NL Ob? | # of ins | Average | | | | Average | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | Var | Bin | LC | NLC | Var | LC | NLC |
| Batch | √ | 10 | 334.6 | 123 | 1089.1 | 1 | 363.8 | 1090.1 | 29.3 |
| MV | √ | 10 | 400 | 200 | 402 | 0 | 800 | 602 | 200 |
| SLay | √ | 14 | 336 | 92 | 437 | 0 | 350 | 437 | 14 |
| trimloss | | 6 | 279.2 | 227.5 | 133.3 | 6 | 324.8 | 139.3 | 45.7 |
| uflquad | √ | 15 | 1227.7 | 22.7 | 1260.3 | 0 | 2432.7 | 1260.3 | 1205 |

**Table 2** Characteristics of non-separable instances

| Instance family | NL Ob? | # of ins | Average | | | |
|---|---|---|---|---|---|---|
| | | | Var | Bin | LC | NLC |
| CLay | | 12 | 116.7 | 35.3 | 138.3 | 40 |
| FLay | | 10 | 158 | 28 | 183 | 4 |
| fo-m-o | | 9 | 112.2 | 41.6 | 194.3 | 13.6 |
| nd | | 5 | 574 | 37.6 | 283.8 | 37.6 |
| RSyn | | 48 | 922.3 | 251 | 1716.3 | 34.2 |
| safetyLay | | 3 | 120.7 | 38 | 111 | 34.7 |
| sssd | | 14 | 162.4 | 135.5 | 50 | 20.1 |
| Syn | | 48 | 366.3 | 95 | 660 | 34.2 |
| tls | | 6 | 279.2 | 227.5 | 133.3 | 6 |
| others | √ | 12 | 205.4 | 86.4 | 206 | 3.3 |

were the sum of separable functions except for the MV instances, which contained convex quadratic functions of the form $h(x) = x^T Qx$ and were reformulated using the change of variables described in Sect. 5.

In Table 1, we give characteristics of the separable instances. The first seven columns in Table 1 list the instance family, whether or not the instance has a nonlinear objective function, the number of instances in the family, the average number of variables, the average number of binary variables, the average number of linear constraints excluding bounds on variables, and the average number of nonlinear constraints for the instances in the family. In the next three columns, we give the average number of variables, the average number of linear constraints, and the average number of nonlinear constraints of the extended formulations of these same instances. In Table 2, we provide the same set of statistics for the remaining families of instances, that are not separable. Instances that had a nonlinear objective function were transformed by moving the objective into the constraints using an auxiliary variable as described in Sect. 1.

## 6.2 Implementation

We implemented our iterative cut generation method as a part of the convex MINLP solver FilMINT which is based on the LP/NLP-Based Branch-and-Bound algorithm of [49]. FilMINT is developed on top of the MILP solver MINTO [48] and uses FilterSQP [32] as the NLP solver and CPLEX [40] version 12.4 as the LP solver. We also use CPLEX to solve CGLP problems.

(Strengthened) lift-and-project cuts are added in rounds following the work of [5]. In each round, we solve the current LP relaxation ($Q_C(\mathcal{K})$) of the MINLP problem. Then, we create linearization cuts in order to strengthen $Q_C(\mathcal{K})$ and solve the LP relaxation again. This step is repeated up to 100 times in order to bring the LP relaxation solution close to the feasible region of nonlinear constraints. Then, for each fractional integer variable, we call Algorithm 1 to generate a lift-and-project cut. We use a tolerance of $1e^{-4}$ to determine whether or not the value of an integer variable is fractional. We conducted experiments both with and without the application of the monoidal strengthening procedure. Note that (strengthened) lift-and-project cuts generated by the algorithm are never included in the formulation of a CGLP, so the cuts we generate are always rank one.

We use the dual formulation of the CGLPs, where the cut coefficients can be readily obtained from the solution of the CGLP. In the iterative method, linearization cuts are added as columns to the dual CGLP similar to a column generation method. We tried both SNC and normalization constraints of the form $\|\alpha\| \leq 1$ in our computational experiments. When using SNC, we do not relax the lower and upper bounds on the decision variables in the CGLP (i.e., $Dx \leq d$ from Sect. 4 is defined by the variable bounds). We tested both $\|\cdot\|_\infty$ and $\|\cdot\|_1$ in the normalization $\|\alpha\| \leq 1$, but we give results only for normalization constraint of $\|\cdot\|_1$ since it performed better than $\|\cdot\|_\infty$ in our initial computational experiments.

In order to eliminate computational difficulties that might arise during generation of a cut, we use a tolerance of $1e^{-2}$ to check whether $\lambda^t \neq 0$ and $\mu^t \neq 0$ in Algorithm 1. We also add linearization cuts only for violated nonlinear constraints and use a tolerance of $1e^{-6}(1 + v_t(\bar{x}))$ for determining whether the nonlinear constraint is violated or not, where $v_t(\bar{x})$ is the amount by which $\bar{x}$ violates the current cut in the CGLP (which could be 0). Since the violation of the nonlinear constraint is the reduced cost of its corresponding linearization cut in the dual CGLP, this strategy attempts to limit the size of the CGLP by only adding linearization cuts that have a sufficiently large reduced cost relative to the current violation of the lift-and-project cut. Finally, we limit the number of iterations in Algorithm 1 to 10, since our initial computational experiments showed that the improvement of a cut's violation deteriorates after 10 iterations.

We measure the improvement in terms of the integrality gap closed. Specifically, let $z_R = \min_{x \in R}\{c^T x\}$ be objective value of the continuous relaxation and $z_{\mathcal{C}} = \min_{x \in \mathcal{C}}\{c^T x\}$ be objective value of the linear relaxation after (strengthened) lift-and-project cuts are added. Then, the percentage gap closed by the (strengthened) lift-and-project cuts is

$$100\left(\frac{z_{\mathcal{C}} - z_R}{z_{\text{MINLP}} - z_R}\right).$$

When reporting this improvement for instances for which the optimal solution value $z_{\text{MINLP}}$ is not known, we use instead the best known upper bound.

We use shifted geometric means to report solution times. The shifted geometric mean of the set $\{t_1, t_2 \ldots, t_N\}$ with shift $s \geq 0$ is defined as $\sqrt[N]{\prod_{i=1}^{N}(t_i + s)} - s$. Shifted geometric mean avoids the mean being dominated by outliers with large values and over-representation of differences among small values. If an instance is not solved within the time limit, then the time limit is used for that instance when calculating the shifted geometric mean.

We use performance profiles [26] to display the relative performance of different methods. A performance profile is a graph of the relative performance of a set of solvers on a fixed set of instances. In a performance profile graph, the $x$-axis is used for the performance metric. The $y$-axis gives the fraction of instances for which the performance of that solver is within a factor of $x$ of the best solver for that instance. In our experiments, we use the CPU solution time as the performance metric.

The computational experiments were run on a cluster of machines equipped with Intel Xeon E7-4850 microprocessors clocked at 2.00 GHz and 256 GB of RAM, using only one thread for each run.

### 6.3 Computation of the rank one lift-and-project closure

In this section, we investigate the strength of the lift-and-project cuts obtained using the strategies from Sects. 3–5. We are interested in understanding how much gap can possibly be closed by these methods, so we continue adding lift-and-project cuts in rounds as long as new cuts are being generated that cut off the current LP relaxation solution, up to an 8 h time limit. When the iterative procedure of Sect. 4 is used, this experiment approximates the strength of the *rank one lift-and-project closure* of $R$:

$$\mathcal{C} \overset{\text{def}}{=} \cap_{i \in I, \pi_0 \in \mathbb{Z}} \operatorname{conv}(R^{(e_i, \pi_0)}).$$

We also experimented with strengthened lift-and-project cuts obtained by applying monoidal strengthening. We found the gaps closed and computation times to be very similar, although slightly more gap is closed. These results are reported in Table 6 in the appendix. In order to isolate the effect of lift-and-project cuts, we disable preprocessing, cuts, and heuristics of FilMINT in these experiments.

Table 3 compares three versions of our lift-and-project implementation. The first version, denoted as `SNC Simple`, implements the simple lift-and-project cut generation method described in Sect. 3 and uses the standard normalization condition. The second version (`SNC Iterative`) implements the iterative technique for generating lift-and-project cuts explained as in Sect. 4 and also uses the standard normalization condition. The third version (`Alpha Iterative`) is the same as `SNC Iterative` except that it uses the normalization constraint $\|\alpha\|_1 \leq 1$. For each version, we present the average results for each instance family in three columns. The first column denotes the average percentage gap closed by lift-and-project cuts at the root node, the second column denotes the average CPU time, and the last column indicates how many instances hit the 8 h time limit for the instances of that family. The table is also divided horizontally into two where the top half gives the results for

**Table 3** Lift-and-project closure results

| Instance family | SNC simple | | | SNC iterative | | | Alpha iterative | | |
|---|---|---|---|---|---|---|---|---|---|
| | Gap Clsd | CPU time | Hit limit | Gap Clsd | CPU time | Hit limit | Gap Clsd | CPU time | Hit limit |
| Batch | 48.2 | 1657 | 0 | 56.9 | 9013 | 4 | 52.8 | 13,740 | 9 |
| Batch-ext | 54.6 | 866 | 0 | 76.0 | 2804 | 0 | 75.5 | 7974 | 3 |
| MV | 0.0 | 14,479 | 10 | 0.0 | 14,513 | 10 | 0.0 | 14,584 | 10 |
| MV-ext | 82.7 | 12,935 | 6 | 97.5 | 4961 | 0 | 94.3 | 10,499 | 1 |
| Slay | 41.2 | 1944 | 1 | 45.9 | 4266 | 3 | 57.4 | 3544 | 2 |
| Slay-ext | 69.4 | 51 | 0 | 86.1 | 83 | 0 | 86.1 | 307 | 0 |
| trimloss | 3.8 | 2006 | 0 | 5.8 | 7182 | 2 | 6.1 | 7518 | 2 |
| Trimloss-ext | 4.8 | 166 | 0 | 9.5 | 2489 | 0 | 9.5 | 3241 | 1 |
| uflquad | 0.1 | 8738 | 8 | 2.2 | 12,523 | 12 | 8.4 | 14,087 | 14 |
| Uflquad-ext | 25.5 | 240 | 0 | 97.0 | 622 | 0 | 79.3 | 5551 | 4 |
| **AvgSepOrg** | 19.7 | 6031 | 19 | 23.3 | 9562 | 31 | 27.2 | 10,714 | 37 |
| **AvgSepExt** | 50.1 | 2606 | 6 | 81.0 | 1874 | 0 | 75.5 | 5304 | 9 |
| Clay | 40.8 | 156 | 0 | 40.8 | 82 | 0 | 40.8 | 41 | 0 |
| Flay | 16.7 | 18 | 0 | 50.7 | 1793 | 1 | 50.7 | 1490 | 1 |
| F-m-o | 2.0 | 3 | 0 | 2.2 | 7 | 0 | 1.7 | 24 | 0 |
| nd | 67.4 | 223 | 0 | 85.0 | 2511 | 0 | 85.0 | 5230 | 1 |
| Rsyn | 83.7 | 87 | 0 | 88.7 | 171 | 0 | 88.7 | 576 | 0 |
| safetyLay | 100.0 | 100 | 0 | 100.0 | 192 | 0 | 100.0 | 19 | 0 |
| sssd | 74.3 | 3 | 0 | 99.7 | 8 | 0 | 99.7 | 141 | 0 |
| Syn | 97.7 | 4 | 0 | 99.8 | 11 | 0 | 99.8 | 33 | 0 |
| tls | 4.4 | 2444 | 1 | 6.3 | 7828 | 2 | 6.6 | 7432 | 2 |
| others | 29.6 | 3642 | 3 | 46.3 | 3898 | 3 | 46.7 | 3967 | 2 |
| **AvgOthers** | 68.5 | 397 | 4 | 76.5 | 807 | 6 | 76.5 | 989 | 6 |

separable instances and the bottom half gives the results for non-separable instances. For separable instances, we include the results for extended formulations in the row after each instance family, with the name of the instance family appended with '-ext'. Table 1 in the Electronic Supplementary Material provides the results obtained on each individual instance.

For the separable instances, we find that all three versions obtain significantly improved average gap closed with significantly less time when using the extended formulation in place of the original formulation. For example, `SNC Iterative` closed 81.0% of the integrality gap on average over the separable instances when using the extended formulation whereas it closed only 23.3% with the original formulation. The reason for this improvement is twofold. First, the linearization cuts are much more effective with extended formulations [38,54]. In the iterative method, we observed that many fewer iterations were required to converge to the best possible cuts when using the extended formulations. Thus, the CPU time to calculate the lift-and-project closure and the number of problems that hit the time limit decreased considerably. The second reason is that the cuts generated in the extended space can be significantly stronger than the ones generated in the original space [11,46]. In many instances, `SNC Iterative` closes significantly more gap on the extended formulation than on the original formulation, even when it was not terminated due to the time limit for either formulation (e.g., 97.7% compared to 46.9% for instance SLay04M).

We next observe that `SNC Iterative` can close significantly more gap than `SNC Simple`. This is due to the reasoning given in Example 1, which demonstrates that the iterative method can cut off points which can not be cut off with simple method. There is a tradeoff, however, as `SNC Iterative` uses more CPU time for most instances. On the other hand, we see that `SNC Iterative` is faster than `SNC Simple` for the extended formulation of separable instances. The main reason for this is that the number of iterations required to converge is much smaller with the extended formulation.

The comparison between `SNC Iterative` and `Alpha Iterative` demonstrates that the methods perform similarly in terms of integrality gap closed (as would be expected in the absence of a time limit, since both approximate the lift-and-project closure). But, `SNC Iterative` is faster than `Alpha Iterative` and hits the time limit in fewer instances. This is compatible with the computational experiments from the MILP literature where SNC is favored against other normalizations [31].

In summary, rank one lift-and-project cuts on average close 81.0% of the gap when using the extended formulation of the separable instances and 76.5% of the gap for the non-separable instances. For specific instance families the results are even more striking. For example, lift-and-project cuts close nearly the entire gap for instances in the `MV-ext`, `uflquad-ext`, `safetyLay`, `sssd`, and `Syn` families.

## 6.4 Branch-and-cut results

We next report our experience using our lift-and-project cut separation procedure within the branch-and-bound method of FilMINT to solve the test instances to optimality. Since monoidal strengthening yields slight improvements in gap closed in comparable time (Table 6 in the appendix) we use strengthend lift-and-project cuts in the implementation. In our experiments, we adopted a cut-and-branch approach where lift-and-project cuts are only added at the root node of the branch-and-bound tree. After FilMINT preprocesses the instance, lift-and-project cuts are concurrently generated with FilMINT's default cuts. Finally, we only use the extended formulation of the separable instances in the computational experiments of this section.

When using strengthened lift-and-project cuts, we disabled the addition of linearization cuts coming from the extended cutting plane (ECP)-based method or the Fixfrac method in FilMINT (see [1]). These linearization cuts are used to capture the nonlinear structure of the problem within the LP/NLP-Based Branch-and-Bound algorithm. Our computational experience indicated that the strengthened lift-and-project cuts provide sufficient information about the nonlinear structure, and that generating these extra linearization cuts was typically not helpful in conjunction with strengthened lift-and-project cuts.

Since our computational experiments from the previous section showed that SNC is superior to the $\alpha$ normalization, we compare default FilMINT with simple and iterative versions of our algorithm that use only SNC. The default version of FilMINT does not create any strengthened lift-and-project cuts and is denoted as `DEFAULT` in the results. In order to reduce the tailing-off effect when adding cuts in rounds at the root node, we stop generating cuts if the objective function value was improved by less than $\epsilon\%$ in the last round of cut generation. By decreasing $\epsilon$, we generate strengthened lift-and-projects more aggressively. In the normal setting, we set $\epsilon = 1$ for the simple and iterative version of our algorithm and they are denoted as `SIMPLE` and `ITERATIVE`, respectively. In the aggressive setting, we set $\epsilon = 1e^{-5}$, and the corresponding methods are denoted as `SIMPLE++` and `ITERATIVE++`.

In Fig. 4, we present the performance profile comparing the relative improvement obtained by using strengthened lift-and-project cuts on all 222 instances. From the full set of 222 instances we collected, we excluded 14 that could not be solved by any of the settings within the time limit of 2 h. In Table 4, we summarize the results for the remaining 208 instances. The first row denotes how many instances are solved to optimality with each method and the second row denotes the number of instances the method did not solve within the time limit. Additionally, we list the number of instances that can be solved with each method in 1, 10, 100 and 1000 s, respectively. Finally, in the last row, we present the shifted geometric mean of solution time with a shift of 10 s. Table 2 in the Electronic Supplementary Material provides the results for each individual instance.

The results from Table 4 and the performance profile in Fig. 4 show that strengthened lift-and-project cuts help FilMINT to solve more instances. `SIMPLE` and `ITERATIVE` are able to solve 15 and 17 more instances to optimality than `DEFAULT`, respectively. The average solution time is improved significantly for both `SIMPLE` and `ITERATIVE` where `ITERATIVE` is performing slightly better than `SIMPLE`. When looking at the number of problems that can be solved with increasing time limits, we see that both `SIMPLE` and `ITERATIVE` consistently solve more instances than

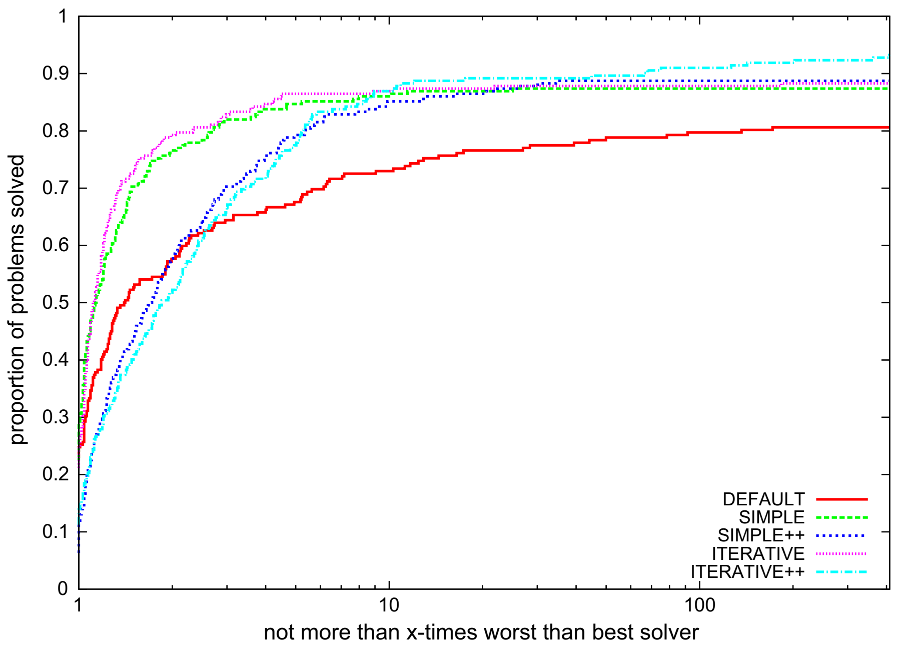

**Fig. 4** Performance profile comparing different strengthened lift-and-project cut settings within FilMINT on all 222 convex test instances

**Table 4** Branch-and-cut results comparing different strengthened lift-and-project cut settings within FilMINT

| | DEFAULT | SIMPLE | ITERATIVE | SIMPLE++ | ITERATIVE++ |
|---|---|---|---|---|---|
| Solved | 179 | 194 | 196 | 197 | 207 |
| Timeout | 29 | 14 | 12 | 11 | 1 |
| <1 | 52 | 56 | 54 | 50 | 44 |
| <10 | 99 | 109 | 106 | 93 | 91 |
| <100 | 137 | 152 | 156 | 140 | 128 |
| <1000 | 162 | 181 | 185 | 182 | 182 |
| Time | 74.0 | 43.4 | 40.6 | 53.3 | 62.0 |

DEFAULT. ITERATIVE is slower than SIMPLE for easier instances, but is able to solve more instances than SIMPLE.

With aggressive cut generation, SIMPLE++ solves three more instances to optimality than its less aggressive counterpart, whereas ITERATIVE++ solves 11 more instances. On the other hand, when aggressively generating strengthened lift-and-project cuts, the extra time spent for cut generation leads to increased average solution times for easier instances.

In order to investigate the effect of strengthened lift-and-project cuts on harder instances, we next exclude the problems that could be solved within 1000 s by all

**Table 5** Branch-and-cut results on 52 hard instances (those for which at least one setting takes more than 1000 s)

| | DEFAULT | SIMPLE | ITERATIVE | SIMPLE++ | ITERATIVE++ |
|---|---|---|---|---|---|
| Solved | 23 | 38 | 40 | 41 | 51 |
| <100 | 2 | 9 | 8 | 7 | 5 |
| <1000 | 6 | 25 | 29 | 26 | 26 |
| Time | 3223.5 | 885.5 | 788.2 | 921.7 | 843.4 |

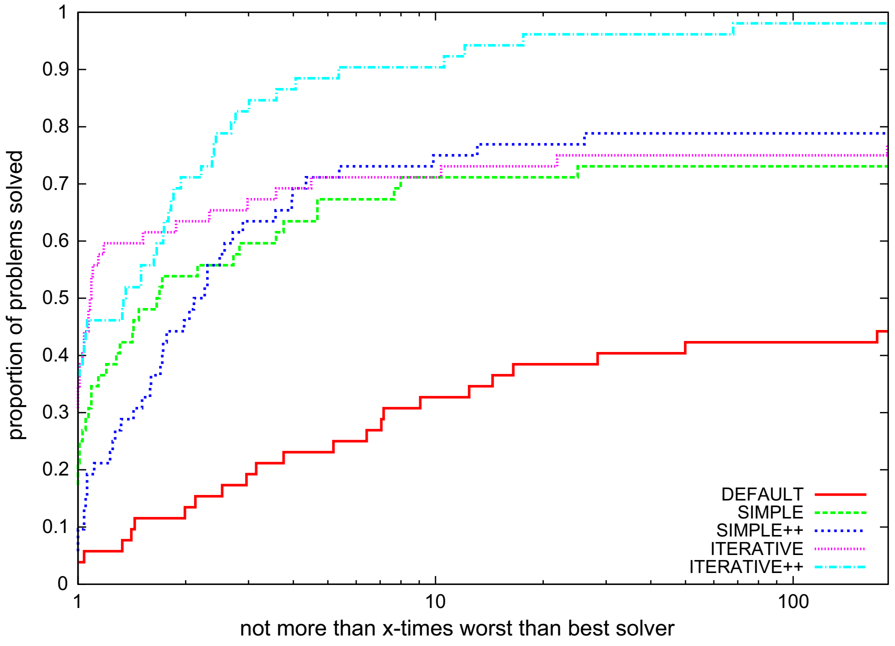

**Fig. 5** Performance profile comparing different strengthened lift-and-project cut settings within FilMINT on 52 hard instances (those for which at least one setting takes more than 1000 s)

settings. Table 5 is formed with the remaining 52 instances, and is designed similarly as Table 4. We find that the hard instances can be solved four times faster on average when using strengthened lift-and-project cuts. In addition, these results underestimate the impact of strengthened lift-and-project cuts, since the solution time for instances that are not solved in the time limit by a method is taken to be the time limit of 2 h when computing the averages. The extra gap closed with the iterative method also pays off on the hard instances, and we see significant reductions in solution time comparing ITERATIVE and ITERATIVE++ with SIMPLE and SIMPLE++, respectively. Interestingly, ITERATIVE++ can solve all ten MV-ext instances to optimality whereas none of other methods can solve any of them. The relative improvement obtained by using strengthened lift-and-project cuts on the hard instances can also be dramatically seen in the performance profile of Fig. 5.

## 7 Conclusion

We have introduced a computationally effective mechanism for generating lift-and-project cuts for convex MINLPs. The methodology relies only on solving a sequence of linear programs, giving it a significant computational advantage over separation approaches that rely on solving a nonlinear program. We have proved that in the limit our methodology can find a cut as strong as what would be obtained using the nonlinear program proposed in [20,53]. Using the proposed procedure allows us for the first time to report a significant computational study aimed at measuring the strength of lift-and-project cuts for convex MINLPs. For many families of convex MINLPs, the lift-and-project cuts close a significant fraction of the gap between the nonlinear relaxation and the optimal value. We also provide an empirical demonstration that using our proposed lift-and-project cut generation procedure on extended formulations derived from exploiting separable convex functions reduces significantly more integrality gap, and requires less time, in comparison to applying the procedure on the original formulation. Finally, the cuts have been successfully incorporated into the software FilMINT, resulting in often dramatic performance improvements. Further improvements to our implementation are possible. For example, by using the membership LP of [13], the size of CGLP used to generate a cut from a current polyhedral outer-approximation could be cut in half. Such a hybrid approach is now used in the commercial software CPLEX [15], and reported reductions in solution time on their convex MINLP test instances are consistent with the findings in this paper.

**Acknowledgements** This work was supported in part by the U.S. National Science Foundation (CCF-0830153) and by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Applied Mathematics program under contract numbers DE-AC02-06CH11357 and DE-FG02-08ER25861. The authors would like to thank Andrew Miller for his insightful comments on this work.

## Appendix

Table 6 presents the gaps closed when using monoidal strenghthening to obtain strengthened lift-and-project cuts in the closure experiment presented in Sect. 6.3. The structure of this table is identical to that of Table 3. Table 3 in the Electronic Supplementary Material provides the results for each individual instance.

**Table 6** Strengthened lift-and-project closure results summarized by instance family

| Instance family | SNC simple | | | SNC iterative | | | Alpha iterative | | |
|---|---|---|---|---|---|---|---|---|---|
| | Gap Clsd | CPU time | Hit limit | Gap Clsd | CPU time | Hit limit | Gap Clsd | CPU time | Hit limit |
| Batch | 48.6 | 873 | 0 | 58.3 | 9376 | 3 | 51.8 | 13,911 | 9 |
| Batch-ext | 56 | 1045 | 0 | 76.8 | 1921 | 0 | 75.3 | 10,044 | 5 |
| MV | 0 | 14, 501 | 10 | 0 | 14,522 | 10 | 0 | 14, 525 | 10 |
| MV-ext | 81.9 | 12, 350 | 5 | 97.7 | 4926 | 0 | 94.5 | 10,904 | 2 |
| Slay | 41.3 | 1470 | 1 | 45.9 | 4338 | 3 | 57.3 | 4454 | 3 |
| Slay-ext | 69.7 | 43 | 0 | 86.1 | 74 | 0 | 86.1 | 453 | 0 |
| trimloss | 5.2 | 932 | 0 | 12.8 | 6943 | 2 | 15 | 6571 | 2 |
| trimloss-ext | 10.4 | 201 | 0 | 23.7 | 1434 | 0 | 22.9 | 3288 | 1 |
| uflquad | 0.1 | 8706 | 8 | 2.2 | 12,414 | 12 | 8.1 | 14,245 | 14 |
| uflquad-ext | 25.5 | 218 | 0 | 96.9 | 637 | 0 | 76.9 | 6713 | 4 |
| **AvgSepOrg** | 19.9 | 5645 | 19 | 24.3 | 9592 | 30 | 27.8 | 10,906 | 38 |
| **AvgSepExt** | 50.9 | 2528 | 5 | 82.6 | 1594 | 0 | 76.3 | 6113 | 12 |
| Clay | 40.8 | 141 | 0 | 40.8 | 81 | 0 | 40.8 | 65 | 0 |
| Flay | 17.2 | 17 | 0 | 50.7 | 1856 | 1 | 50.7 | 1562 | 1 |
| f-m-o | 2 | 4 | 0 | 2.2 | 7 | 0 | 1.7 | 37 | 0 |
| nd | 67 | 224 | 0 | 85 | 2164 | 0 | 85 | 6113 | 1 |
| Rsyn | 84.5 | 78 | 0 | 89.2 | 149 | 0 | 88.8 | 840 | 0 |
| safetyLay | 100 | 121 | 0 | 100 | 121 | 0 | 100 | 91 | 0 |
| sssd | 76.8 | 2 | 0 | 99.7 | 6 | 0 | 99.7 | 229 | 0 |
| Syn | 98 | 4 | 0 | 99.7 | 10 | 0 | 99.8 | 59 | 0 |
| tls | 6.2 | 1441 | 0 | 12.8 | 7318 | 2 | 15.2 | 8468 | 3 |
| others | 36 | 3644 | 3 | 46.9 | 3845 | 3 | 47.3 | 4729 | 3 |
| **AvgOthers** | 69.6 | 358 | 3 | 76.9 | 770 | 6 | 76.9 | 1206 | 8 |

## References

1. Abhishek, K., Leyffer, S., Linderoth, J.T.: FilMINT: an outer-approximation-based solver for nonlinear mixed integer programs. INFORMS J. Comput. **22**, 555–567 (2010)
2. Atamtürk, A., Narayanan, V.: Conic mixed integer rounding cuts. Math. Program. **122**, 1–20 (2010)
3. Balas, E.: Disjunctive programming. Ann. Discret. Math. **5**, 3–51 (1979)
4. Balas, E.: A modified lift-and-project procedure. Math. Program. **79**, 19–31 (1997)
5. Balas, E., Ceria, S., Cornuejols, G.: Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Manag. Sci. **42**, 1229–1246 (1996)
6. Balas, E., Jeroslow, R.G.: Strengthening cuts for mixed integer programs. Eur. J. Oper. Res. **4**, 224–234 (1980)
7. Balas, E., Perregaard, M.: A precise correspondence between lift-and-project cuts, simple disjunctive cuts. Math. Program. Ser. B **94**, 221–245 (2003)
8. Balas, E., Saxena, A.: Optimizing over the split closure. Math. Program. **113**, 219–240 (2008)
9. Bertsekas, D., Gallager, R.: Data Networks, 2nd edn. Prentice-Hall Inc, Upper Saddle River (1992)
10. Bodur, M., Dash, S., Günlük, O.: Cutting planes from extended LP formulations. Math. Program. (2016). doi:10.1007/s10107-016-1005-7
11. Bodur, M., Dash, S., Günlük, O., Luedtke, J.: Strengthened Benders Cuts for Stochastic Integer Programs with Continuous Recourse. http://www.optimization-online.org/DB_HTML/2014/03/4263.html (2014)
12. Bonami, P.: Lift-and-project cuts for mixed integer convex programs. In: Günlük, O., Woeginger, G. (eds.) Integer Programming and Combinatoral Optimization, Lecture Notes in Computer Science, vol. 6655, pp. 52–64. Springer, Berlin (2011). doi:10.1007/978-3-642-20807-2_5
13. Bonami, P.: On optimizing over lift-and-project closures. Math. Program. Comput. **4**, 151–179 (2012)
14. Bonami, P., Minoux, M.: Using rank-1 lift-and-project closures to generate cuts for 0–1 MIPs, a computational investigation. Discret. Optim. **2**, 288–307 (2005)
15. Bonami, P., Tramontani, A.: Advances in CPLEX for mixed integer nonlinear optimization. International Symposium on Mathematical Programming. Pittsburgh. PA, USA (2015)
16. Boorstyn, R., Frank, H.: Large-scale network topological optimization. IEEE Trans. Commun. **25**, 29–47 (1977)
17. Borchers, B., Mitchell, J.E.: An improved branch and bound algorithm for mixed integer nonlinear programs. Comput. Oper. Res. **21**, 359–368 (1994)
18. 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)
19. Castillo, I., Westerlund, J., Emet, S., Westerlund, T.: Optimization of block layout design problems with unequal areas: a comparison of MILP and MINLP optimization methods. Comput. Chem. Eng. **30**, 54–69 (2005)
20. Ceria, S., Soares, J.: Convex programming for disjunctive optimization. Math. Program. **86**, 595–614 (1999)
21. Cezik, M.T., Iyengar, G.: Cuts for mixed 0–1 conic programming. Math. Program. **104**, 179–202 (2005)
22. Conforti, M., Cornuéjols, G., Zambelli, G.: Polyhedral approaches to mixed integer linear programming. In: Jünger, M., Liebling, T., Naddef, D., Pulleyblank, W., Reinelt, G., Rinaldi, G., Wolsey, L. (eds.) 50 Years of Integer Programming 1958–2008, pp. 343–385. Springer, New York (2009)
23. Cornuéjols, G.: Valid inequalities for mixed integer linear programs. Math. Program. Ser. B **112**, 3–44 (2008)
24. Dash, S., Günlük, O., Lodi, A.: MIR closures of polyhedral sets. Math. Program. **121**, 33–60 (2010)
25. Dash, S., Günlük, O., Vielma, J.: Computational experiments with cross and crooked cross cuts. INFORMS J. Comput. **26**, 780–797 (2014)
26. Dolan, E., Moré, J.: Benchmarking optimization software with performance profiles. Math. Program. **91**, 201–213 (2002)
27. Drewes, S.: Mixed Integer Second Order Cone Programming. Ph.D. thesis, Technische Universität Darmstadt (2009)
28. Duran, M.A., Grossmann, I.: An outer-approximation algorithm for a class of mixed-integer nonlinear programs. Math. Program. **36**, 307–339 (1986)
29. Elhedhli, S.: Service system design with immobile servers, stochastic demand, and congestion. Manuf. Serv. Oper. Manag. **8**, 92–97 (2006)

30. Fischetti, M., Lodi, A.: Optimizing over the first Chvátal closure. Math. Program. Ser. B **110**, 3–20 (2007)
31. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. Math. Program. **128**, 205–230 (2011)
32. Fletcher, R., Leyffer, S.: User Manual for FilterSQP. University of Dundee Numerical Analysis Report NA-181 (1998)
33. Frangioni, A., Gentile, C.: Perspective cuts for a class of convex 0–1 mixed integer programs. Math. Program. **106**, 225–236 (2006)
34. Gomory, R.E.: An Algorithm for the Mixed Integer Problem. Technical Report RM-2597, The RAND Corporation (1960)
35. Günlük, O., Lee, J., Weismantel, R.: MINLP strengthening for separable convex quadratic transportation-cost UFL. Technical Report RC24213 (W0703-042), IBM Research Division (2007)
36. Günlük, O., Linderoth, J.: Perspective relaxation of mixed integer nonlinear programs with indicator variables. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO, Lecture Notes in Computer Science, vol. 5035, pp. 1–16. Springer, New York (2008)
37. Harjunkoski, I., Westerlund, T., Porn, R., Skrifvars, H.: Different transformations for solving non-convex trim loss problems by MINLP. Eur. J. Oper. Res. **105**, 594–603 (1998)
38. Hijazi, H., Bonami, P., Ouorou, A.: An outer-inner approximation for separable mixed-integer nonlinear programs. INFORMS J. Comput. **26**, 31–44 (2014)
39. Hiriart-Urruty, J.B., Lemarechal, C.: Convex Analysis and Minimization Algorithms I: Fundamentals (Grundlehren Der Mathematischen Wissenschaften). Springer, New York (1993)
40. IBM: Using the CPLEX Callable Library, Version 12 (2009)
41. Kelley, J.: The cutting-plane method for solving convex programs. J. SIAM **8**, 703–712 (1960)
42. Kılınç, M., Linderoth, J., Luedtke, J.: Effective Separation of Disjunctive Cuts for Convex Mixed Integer Nonlinear Programs. Technical Report 1681, Computer Sciences Department, University of Wisconsin-Madison (2010)
43. Kılınç, M.R.: Disjunctive Cutting Planes and Algorithms for Convex Mixed Integer Nonlinear Programming. Ph.D. thesis, University of Wisconsin-Madison (2011)
44. Leyffer, S.: MacMINLP: Test Problems for Mixed Integer Nonlinear Programming. http://www-unix.mcs.anl.gov/~leyffer/macminlp (2003)
45. Modaresi, S., Kılınç, M., Vielma, J.P.: Intersection cuts for nonlinear integer programming: convexification techniques for structured sets. Math. Program. **155**, 575–611 (2016)
46. Modaresi, S., Kılınç, M.R., Vielma, J.P.: Split cuts and extended formulations for mixed integer conic quadratic programming. Oper. Res. Lett. **43**(1), 10–15 (2015)
47. Nemhauser, G., Wolsey, L.: A recursive procedure for generating all cuts for 0–1 mixed integer programs. Math. Program. **46**, 379–390 (1990)
48. Nemhauser, G.L., Savelsbergh, M.W.P., Sigismondi, G.C.: MINTO, a mixed INTeger optimizer. Oper. Res. Lett. **15**, 47–58 (1994)
49. Quesada, I., Grossmann, I.E.: An LP/NLP based branch-and-bound algorithm for convex MINLP optimization problems. Comput. Chem. Eng. **16**, 937–947 (1992)
50. Ravemark, D.E., Rippin, D.W.T.: Optimal design of a multi-product batch plant. Comput. Chem. Eng. **22**(1–2), 177–183 (1998)
51. Sawaya, N.: Reformulations, Relaxations and Cutting Planes for Generalized Disjunctive Programming. Ph.D. thesis, Chemical Engineering Department, Carnegie Mellon University (2006)
52. Sawaya, N., Laird, C.D., Biegler, L.T., Bonami, P., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Lee, J., Lodi, A., Margot, F., Wächter, A.: CMU-IBM Open Source MINLP Project Test Set. http://egon.cheme.cmu.edu/ibm/page.htm. Accessed 19 April 2016
53. Stubbs, R., Mehrotra, S.: A branch-and-cut method for 0–1 mixed convex programming. Math. Program. **86**, 515–532 (1999)
54. Tawarmalani, M., Sahinidis, N.: A polyhedral branch-and-cut approach to global optimization. Math. Program. **103**(2), 225–249 (2005)
55. Türkay, M., Grossmann, I.E.: Logic-based MINLP algorithms for the optimal synthesis of process networks. Comput. Chem. Eng. **20**(8), 959–978 (1996)
56. Vecchietti, A., Grossmann, I.E.: LOGMIP: a disjunctive 0–1 non-linear optimizer for process system models. Comput. Chem. Eng. **23**(4–5), 555–565 (1999). doi:10.1016/S0098-1354(98)00293-2. http://www.sciencedirect.com/science/article/B6TFT-3XY28Y0-B/2/2709e69a55450cf2263efcc5368850db

57. Vielma, J., Dunning, I., Huchette, J., Lubin, M.: Extended Formulations in Mixed Integer Conic Quadratic Programming. Technical Report. http://arxiv.org/abs/1505.07857 (2015)
58. Zhu, Y., Kuno, T.: A disjunctive cutting-plane-based branch-and-cut algorithm for 0–1 mixed-integer convex nonlinear programs. Ind. Eng. Chem. Res. **45**(1), 187–196 (2006). doi:10.1021/ie0402719