

Lift-and-Project Cuts for Mixed Integer Convex Programs

Pierre Bonami

LIF, CNRS Aix-Marseille Université, 163 avenue de Luminy - Case 901 F-13288
Marseille Cedex 9 France
`pierre.bonami@lif.univ-mrs.fr`

Abstract. This paper addresses the problem of generating cuts for mixed integer nonlinear programs where the objective is linear and the relations between the decision variables are described by convex functions defining a convex feasible region. We propose a new method for strengthening the continuous relaxations of such problems using cutting planes. Our method can be seen as a practical implementation of the lift-and-project technique in the nonlinear case. To derive each cut we use a combination of a nonlinear programming subproblem and a linear outer approximation. One of the main features of the approach is that the subproblems solved to generate cuts are typically not more complicated than the original continuous relaxation. In particular they do not require the introduction of additional variables or nonlinearities. We propose several strategies for using the technique and present preliminary computational evidence of its practical interest. In particular, the cuts allow us to improve over the state of the art branch-and-bound of the solver Bonmin, solving more problems in faster computing times on average.

Keywords: Mixed Integer Nonlinear Programming, Disjunctive Programming, Lift-and-Project, Cutting Planes.

1 Introduction

In this paper we consider mixed integer nonlinear programs of the form

$$\begin{aligned} \min \quad & c^T x \\ g_i(x) & \leq 0 \quad i = 1, \dots, m \\ x_j & \in \mathbb{Z} \quad j = 1, \dots, p \\ x_j & \in \mathbb{R} \quad j = p + 1, \dots, n \end{aligned} \tag{MICP}$$

where $1 \leq p \leq n$, $c \in \mathbb{R}^n$ and for $i = 1, \dots, m$, $g_i : \mathbb{R}^n \rightarrow \mathbb{R} \cup \{+\infty\}$ is a continuously differentiable convex function. Because of the convexity of the function g_i we call this problem a mixed integer convex program. Note that our problem formulation does not have bounds on the variables. If any are present, we assume that they are among the constraints $g_i(x) \leq 0$. Note also that any

mixed-integer minimization problem with a convex objective function and lower or equal constraints described by convex functions can be put into the form of (MICP) by adding a variable and putting the objective in the constraints.

MICP can be seen as a generalization of Mixed Integer Linear Programming. It has received a sustained level of attention in recent years. In particular, a successful line of research has been to try and extend techniques that have been successful in the solution of MILPs to MICP. This has led to the design of algorithms and solvers that can effectively solve MICPs of relatively large size [1,12] (see also [14] for a recent survey of these efforts). One aspect that is still missing in that context is efficient techniques for strengthening the continuous relaxation of (MICP).

Cutting plane techniques are one of the main ingredients used to achieve that in modern mixed-integer linear programming solvers. Among the most widely used cuts are Gomory's Mixed Integer cuts [22,6], Mixed Integer Rounding cuts [27], Extended covers [10]. . .

Inspired by these methods, several cutting plane approaches have been proposed for MICP. In particular, Ceria, Soares [17] and Stubbs, Mehrotra [31] have proposed generalizations of disjunctive programming [3] and the lift-and-project approach [5] to nonlinear convex sets and MICP. Unfortunately these approaches require the solution of nonlinear programs in extended spaces and which feature so called perspective functions that are not handled well by general purpose nonlinear programming algorithms (these functions are not differentiable everywhere). Probably because of these difficulties, to the best of our knowledge, these methods have never been tested on a large scale and are not included in solvers. Some other approaches have been developed for specific classes of MICPs. In particular, several methods have been proposed for MICPs where the nonlinear constraints are conic constraints. Cezik and Iyengar [18] have generalized the Chvátal-Gomory and lift-and-project cuts. Atamtürk and Narayanan [2] have proposed a generalization of MIR. These approaches have given encouraging computational results but they are restricted to specific types of problems.

Our goal in this paper is to propose a cutting plane technique for MICP that can be applied to the general case and that is computationally good enough to be used in solvers as an effective tool for strengthening continuous relaxations. Our approach is closely related to lift-and-project techniques in MILP, in particular it can be seen as a generalization of [9,11] to MICP.

Suppose that we are given to cut a point $\bar{x} \in \mathbb{R}^n$ that does not satisfy the integrality requirements of (MICP). Similarly to what is done in MILP, our approach tries to generate one cut for each variable x_j , $1 \leq j \leq p$, such that $\bar{x}_j \notin \mathbb{Z}$. For each x_j , the cut is built in two phases. In the first phase, we solve a nonlinear program (NLP for short) that tells us if a cut that separates \bar{x} by using the integrality of x_j exists. If the answer to the first phase is yes, the second phase builds an outer approximation of (MICP) and derives a cut from it by Linear Programming (LP). An important property is that a cut is always generated in the second phase if the answer to the first phase is yes and a constraint qualification holds. Another desirable property is that the nonlinear program to

solve in the first phase is defined in the same space as the original problem and is typically not more difficult to solve than the continuous relaxation of (MICP). We propose two different strategies to use this cutting plane technique and test them within the solver Bonmin [12,13]. Our conclusions are positive in that we are able to close a significant proportion of the integrality gap on several classes of problems and this helps to effectively solve some hard problems faster with branch-and-bound.

Let us note that very recently, another proposal for a practical cutting plane approach for (MICP) has been made by Linderoth, Kılınç and Luedtke [23]. This approach and the one presented here bear some similarities but also have some major differences. The main one, is that the approach in [23] does not use nonlinear programming but solely relies on an outer approximation of (MICP) refined by constraint generation. The advantage of [23] is that it may be more lightweight in some cases but the disadvantage is that the constraint generation can have very slow convergence properties. We benefit from the good convergence properties offered by sophisticated nonlinear programming algorithms.

The paper is organized as follows. In Sect. 2, we outline our procedure for generating each cut. In Sect. 3, we present in details the NLP used in the first phase of our algorithm and establish some of its main properties. In Sect. 4, we present the LP used in the second phase of our algorithm. In Sect. 5, we describe two strategies for using our cutting planes. Finally, in Sect. 6, we report on the experimental testing of our method.

2 Outline of the Approach

We denote by $C := \{x \in \mathbb{R}^n : g_i(x) \leq 0, i = 1, \dots, m\}$ the feasible set of the continuous relaxation of (MICP) and by $X := \text{conv}(C \cap (\mathbb{Z}^p \times \mathbb{R}^{n-p}))$ the convex hull of the feasible solutions of (MICP). Let $\bar{x} \in C \setminus (\mathbb{Z}^p \times \mathbb{R}^{n-p})$ be the fractional point that we want to separate from X . Note that we do not make further assumptions on \bar{x} . In particular, we do not assume that it is an extreme or an exposed point of C and it may even belong to X (of course in such a case our approach would not find any cut).

To generate inequalities, we use a *split relaxation* of X which is defined as follows. We call split an integral vector $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ such that $\pi_j = 0$, $j = p+1, \dots, n$. We then define the split relaxation corresponding to (π, π_0) as

$$C^{(\pi, \pi_0)} := \text{conv}(C \cap (\{x \in \mathbb{R}^n : \pi^T x \leq \pi_0\} \cup \{x \in \mathbb{R}^n : \pi^T x \geq \pi_0 + 1\})).$$

Clearly $X \subseteq C^{(\pi, \pi_0)}$ since, for any $x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$, $\pi^T x \in \mathbb{Z}$ and $\mathbb{Z} \cap]\pi_0, \pi_0 + 1[= \emptyset$.

Given a split (π, π_0) , our procedure decides if $\bar{x} \in C^{(\pi, \pi_0)}$ and finds a separating hyperplane if this not the case. In this paper, we only use elementary splits of the form $(e_k, \lfloor \bar{x}_k \rfloor)$ (where $1 \leq k \leq p$ and e_k is the k -th unit vector). Nevertheless our method applies to any split. In this section we give a geometric outline of the approach using a general split.

Our approach is based on the following simple lemma that states a necessary and sufficient condition for \bar{x} to belong to $C^{(\pi, \pi_0)}$.

Lemma 1. *Let $(\pi, \pi_0) \in \mathbb{Z}^{n+1}$ be a valid split and let $\bar{x} \in C$ be such that $\pi^T \bar{x} - \pi_0 \in]0, 1[$. $\bar{x} \in C^{(\pi, \pi_0)}$ if and only if there exists $x^1 \in C$ such that*

- (i) $\frac{\bar{x} - (\pi^T \bar{x} - \pi_0)x^1}{1 - \pi^T \bar{x} + \pi_0} \in C$,
- (ii) $\pi^T x^1 \geq \pi_0 + 1$.

Proof. Suppose first that $\bar{x} \in C^{(\pi, \pi_0)}$, then there exists $x^0, x^1 \in C$ and $\lambda \in [0, 1]$ such that $\pi^T x^0 \leq \pi_0$, $\pi^T x^1 \geq \pi_0 + 1$ and $\bar{x} = \lambda x^1 + (1 - \lambda)x^0$. Note that, without loss of generality, we can assume that $\pi^T x^0 = \pi_0$ and $\pi^T x^1 = \pi_0 + 1$ (just take the intersections of the initial segment $[x^0, x^1]$ with the two hyperplanes $\pi^T x = \pi_0$ and $\pi^T x = \pi_0 + 1$). Condition (ii) clearly holds. Also, since $\pi^T(x^1 - x^0) = 1$, $\pi^T \bar{x} = \lambda \pi^T x^1 + (1 - \lambda)\pi^T x^0 = \lambda + \pi^T x^0$, and $\lambda = \pi^T \bar{x} - \pi_0$. Condition (i) follows.

Suppose now that $x^1 \in C$ satisfies (i) and (ii). Let $x^0 := \frac{\bar{x} - (\pi^T \bar{x} - \pi_0)x^1}{1 - \pi^T \bar{x} + \pi_0}$. By (i), $x^0 \in C$. Furthermore, we clearly have $\bar{x} = (\pi^T \bar{x} - \pi_0)x^1 + (1 - \pi^T \bar{x} + \pi_0)x^0$. Since $\pi^T x^1 \geq \pi_0 + 1$ by hypothesis, we just need to verify that $\pi^T x^0 \leq \pi_0$ to prove the result. Indeed, $\pi^T x^0 = \frac{\pi^T \bar{x} - (\pi^T \bar{x} - \pi_0)\pi^T x^1}{1 - \pi^T \bar{x} + \pi_0} \leq \frac{\pi^T \bar{x} - (\pi^T \bar{x} - \pi_0)(\pi_0 + 1)}{1 - \pi^T \bar{x} + \pi_0} = \pi_0 \frac{1 - \pi^T \bar{x} + \pi_0}{1 - \pi^T \bar{x} + \pi_0}$. \square

We are now ready to describe the two phases of our separation procedure for $C^{(\pi, \pi_0)}$. In the first phase, we find $\bar{x}^1 \in C$, satisfying condition (i) of the lemma and maximizing $\pi^T x$. In Sect. 3, it will be shown that this can be accomplished by solving a convex NLP formulated in \mathbb{R}^n . If $\pi^T \bar{x}^1 \geq \pi_0 + 1$, \bar{x}^1 proves that $\bar{x} \in C^{(\pi, \pi_0)}$ and we can not generate a cut using this split. Now suppose that $\pi^T \bar{x}^1 < \pi_0 + 1$. Then $\bar{x} \notin C^{(\pi, \pi_0)}$ and we enter the second phase of the procedure. We build an outer approximation $P(\bar{x}^0, \bar{x}^1)$ of C by taking the first order approximations of the constraints of C in the points \bar{x}^1 and $\bar{x}^0 := \frac{\bar{x} - (\pi^T \bar{x} - \pi_0)\bar{x}^1}{1 - \pi^T \bar{x} + \pi_0}$:

$$P(\bar{x}^0, \bar{x}^1) = \left\{ x \in \mathbb{Z}^p \times \mathbb{R}^{n-p} : \nabla g_i(\bar{x}^1)^T (x - \bar{x}^1) + g_i(\bar{x}^1) \leq 0, i = 1, \dots, m, \right. \\ \left. \nabla g_i(\bar{x}^0)^T (x - \bar{x}^0) + g_i(\bar{x}^0) \leq 0, i = 1, \dots, m \right\}.$$

Similarly to $C^{(\pi, \pi_0)}$, we define the split relaxation $P(\bar{x}^0, \bar{x}^1)^{(\pi, \pi_0)}$ of $P(\bar{x}^0, \bar{x}^1)$. A cut separating \bar{x} from $P(\bar{x}^0, \bar{x}^1)^{(\pi, \pi_0)}$ can then be sought using classical polyhedral disjunctive programming techniques. We will show in Sect. 4 that, such a cut can typically be found by solving an LP in \mathbb{R}^n . In particular, we show that, if the optimal solution of the NLP solved in the first phase satisfies a constraint qualification, a cut is always found by solving this LP.

The fact that the solution of the NLP solved in the first phase has to satisfy a constraint qualification for our procedure to generate a cut in the second phase is not to be overlooked. In particular, we will show that whenever \bar{x} is an extreme point of C the NLP for finding \bar{x}^1 as defined above leads to a degenerate NLP where no constraint qualification holds. The procedure will be slightly amended to solve this issue.

3 Phase I: Testing If \bar{x} Belongs to a Split Relaxation

In this section, we derive the NLP solved in the first phase of our procedure to decide if $\bar{x} \in C^{(\pi, \pi_0)}$. Since in practice here, we only separate cuts using elementary splits of the form $(e_k, \lfloor \bar{x}_k \rfloor)$ (where $1 \leq k \leq p$), from now on, for simplicity's sake, we restrict ourselves to this case (the procedure remains valid for any split). We denote the fractional part of \bar{x}_k by $f_0 := \bar{x}_k - \lfloor \bar{x}_k \rfloor$ and assume $f_0 > 0$.

Consider the optimization problem

$$\begin{aligned} \max \quad & y_k - f_0 \lfloor \bar{x}_k \rfloor \\ & g_i \left(\frac{y}{f_0} \right) \leq 0 \quad i = 1, \dots, m, \\ & g_i \left(\frac{\bar{x} - y}{1 - f_0} \right) \leq 0 \quad i = 1, \dots, m. \end{aligned} \quad (\text{MNLP})$$

Let y be an optimal solution to (MNLP), $x^1 := \frac{y}{f_0}$ satisfies $x^1 \in C$ and condition (i) of Lemma 1. If furthermore, the objective value is non-negative, x^1 satisfies condition (ii) of the lemma and therefore $\bar{x} \in C^{(e_k, \lfloor \bar{x}_k \rfloor)}$. Therefore, solving (MNLP) solves the phase I of our procedure as defined in Sect. 2.

Note that (MNLP) is a convex program: by definition, all the functions g_i are convex and the nonlinear functions describing the constraints of (MNLP) are obtained by composing g_i with an affine function (either $\frac{y}{f_0}$ or $\frac{\bar{x}-y}{1-f_0}$). (MNLP) has the same number of variables as our original problems and twice as many constraints. Note that if there are any linear or bound constraints in the original problem (MNLP) can be made smaller in practice. Indeed if g_i is an affine function, $g_i(\frac{y}{f_0}) \leq 0$ and $g_i(\frac{\bar{x}-y}{1-f_0}) \leq 0$ can be reformulated as $g_i(\bar{x}) + (1 - f_0)g_i(0) \leq g_i(y) \leq (1 - f_0)g_i(0)$. Therefore, (MNLP) can be reformulated as a problem that has as many linear constraints as (MICP) and twice as many nonlinear constraints.

We note that (MNLP) is a generalization in the nonlinear setting of a separation problem recently proposed [20,11]. In that respect, (MNLP) is related to the famous Cut Generation LP used in disjunctive programming [5] in that, in the linear case, the dual of (MNLP) is equivalent to the standard CGLP of lift-and-project procedures with a particular normalization condition (see [11] for the precise equivalence). A well-known property of the linear version of (MNLP) is that, if \bar{x} is an extreme point of the continuous relaxation, the solution of (MNLP) is unique (and actually corresponds to the GMI cut) [19,11]. Next lemma generalizes this result to our nonlinear setting.

Lemma 2. *Let $\bar{x} \in C$, be such that $\bar{x}_k \notin \mathbb{Z}$. If \bar{x} is an extreme point of C , then the unique solution to (MNLP) is $y = f_0 \bar{x}$.*

Proof. It follows directly from the fact that if \bar{x} is an extreme point, then by definition, the only $x^1 \in C$ satisfying condition (i) of Lemma 1 is \bar{x} itself. \square

Lemma 2 could be seen as positive. Indeed, if \bar{x} is an extreme point of C then (MNLP) can be solved by a closed form formula and furthermore (MNLP) shows that $\bar{x} \notin C^{(e_k, \lceil \bar{x}_k \rceil)}$ since $f_0(\bar{x}_k - \lceil \bar{x}_k \rceil) < 0$. Lemma 2 actually has rather dire implications. Indeed, if $y = f_0 \bar{x}$, then, for $i = 1, \dots, m$, the two nonlinear inequalities $g_i(\frac{y}{f_0}) \leq 0$ and $g_i(\frac{\bar{x} - y}{1 - f_0}) \leq 0$ are identical and therefore no constraint qualification holds in y . It follows, that no useful dual information exists. Without dual information, it is hard to imagine how a cut can be derived. Note that, in general, the optimal solution of the continuous relaxation of (MICP) should be expected to be an extreme point (this is at least true when the optimum is unique). Therefore it should be expected that the point \bar{x} we want to cut is an extreme point of C . Clearly, this issue needs to be addressed if we want to be able to strengthen the continuous relaxation in a systematic way.

To resolve this issue, we make a small modification to (MNLP). Consider the following nonlinear program:

$$\begin{aligned} \max \quad & y_k - f_0 \lceil \bar{x}_k \rceil \\ & g_i \left(\frac{y}{f_0} \right) \leq f_0 \lceil \bar{x}_k \rceil - y_k \quad i = 1, \dots, m, \\ & g_i \left(\frac{\bar{x} - y}{1 - f_0} \right) \leq f_0 \lceil \bar{x}_k \rceil - y_k \quad i = 1, \dots, m. \end{aligned} \quad (\text{MNLP}')$$

The difference between (MNLP') and (MNLP) is the term $f_0 \lceil \bar{x}_k \rceil - y_k$ that we add to the right-hand-side of the constraints. The effect of this term is to relax the constraints whenever $y_k - f_0 \lceil \bar{x}_k \rceil < 0$ (therefore in particular when $y = f_0 \bar{x}$) while keeping the property that if the optimal solution to (MNLP') is non-negative $x^1 := \frac{y}{f_0}$ satisfies the conditions of Lemma 1. We can not claim that a constraint qualification always holds at the optimum of (MNLP') but our practical experiment is that it is typically not an issue.

The modification made from (MNLP) to (MNLP') may seem puzzling. It is actually connected to a well known technique of lift-and-project. Indeed, if we again assume that all functions g_i are linear, it can be shown that (MNLP') is equivalent to the CGLP with the normalization condition $\sum(u_i + v_i) + u_0 + v_0 = 1$ first presented in [8]. Therefore, the modification made from (MNLP) to (MNLP') can be seen as a change in the normalization condition from $u_0 + v_0 = 1$ to $\sum(u_i + v_i) + u_0 + v_0 = 1$. In this light, the change can be seen as even more desirable since in the context of MILP several experiments have concluded that the latter typically gives better cuts [29,4,19]. Note that in practice, in this paper, we only add the term $f_0 \lceil \bar{x}_k \rceil - y_k$ to the nonlinear constraints of (MNLP). Doing this is enough to avoid the numerical difficulties and allows to keep (MNLP') compact (if the term is added to linear constraints then their number has to be doubled).

4 Phase II: Computing the Cut

We now turn to the description of the second phase of the procedure. At the beginning of Phase II, we have a point \bar{y} optimal solution of (MNLP') and

such that $\frac{\bar{y}}{f_0} - \lceil \bar{x}_k \rceil < 0$. Note that, because (MNLP') relaxed the constraints of (MNLP), $\frac{\bar{y}}{f_0} \notin C$ and condition (i) of Lemma 1 is not satisfied. But the maximality of \bar{y} still proves that $\bar{x} \notin C^{(\pi, \pi_0)}$.

As explained in Sect. 2, we now build an outer approximation $P(\bar{x}^0, \bar{x}^1)$ of C by taking linear approximations in $\bar{x}^1 := \frac{\bar{y}}{f_0}$ and $\bar{x}^0 := \frac{\bar{x} - \bar{y}}{1 - f_0}$ (note that the validity of this outer approximation does not depend on the fact that \bar{x}^1 and $\bar{x}^0 \in C$).

Once this outer-approximation is built, a cut can be found by classical disjunctive programming techniques. In our case, we chose to solve an LP which is just the linear version of (MNLP) formulated for $P(\bar{x}^0, \bar{x}^1)^{(e_k, \lceil \bar{x}_k \rceil)}$:

$$\begin{aligned} \max \quad & y_k - f_0 \lceil \bar{x}_k \rceil \\ \nabla g_i(\bar{x}^1)^T y & \geq \nabla g_i(\bar{x}^1)^T \bar{x} + (1 - f_0) (g_i(\bar{x}^1) - \nabla g_i(\bar{x}^1)^T \bar{x}^1) & i = 1, \dots, m, \\ \nabla g_i(\bar{x}^1)^T y & \leq -f_0 (g_i(\bar{x}^1) - \nabla g_i(\bar{x}^1)^T \bar{x}^1), & i = 1, \dots, m, \\ \nabla g_i(\bar{x}^0)^T y & \geq \nabla g_i(\bar{x}^0)^T \bar{x} + (1 - f_0) (g_i(\bar{x}^0) - \nabla g_i(\bar{x}^0)^T \bar{x}^0), & i = 1, \dots, m, \\ \nabla g_i(\bar{x}^0)^T y & \leq -f_0 (g_i(\bar{x}^0) - \nabla g_i(\bar{x}^0)^T \bar{x}^0), & i = 1, \dots, m, \\ x & \in \mathbb{R}^n. \end{aligned} \tag{MLP}(\bar{x}^0, \bar{x}^1)$$

It follows directly from linear programming duality that a cut can be constructed from any dual feasible basis of (MLP(\bar{x}^0, \bar{x}^1)) with negative primal cost (see for example [11] for details).

Note that other Cut Generation LPs could be used to find a cut separating \bar{x} from $P(\bar{x}^0, \bar{x}^1)^{(e_k, \lceil \bar{x}_k \rceil)}$ in this step. One advantage of (MLP(\bar{x}^0, \bar{x}^1)) is its size. Indeed, it does not require the introduction of extra variables and can typically be solved very quickly. Note also, that the cut obtained from the solution of (MLP(\bar{x}^0, \bar{x}^1)) can be strengthened by the classical monoidal strengthening method [7,5].

As the next theorem shows, a fundamental property of (MLP(\bar{x}^0, \bar{x}^1)) is that if a constraint qualification holds at the optimum of (MNLP') then the optimal value of (MLP(\bar{x}^0, \bar{x}^1)) is lower or equal to that of (MNLP') (and therefore a cut can be generated).

Theorem 1. *Let \bar{y} be an optimal solution of (MNLP') satisfying a constraint qualification. Let $\bar{x}^1 := \frac{\bar{y}}{f_0}$ and $\bar{x}^0 := \frac{\bar{x} - \bar{y}}{1 - f_0}$, and let \hat{y} be a solution to (MLP(\bar{x}^0, \bar{x}^1)). If $\bar{y}_k < f_0 \lceil \bar{x}_k \rceil$, then $\hat{y}_k \leq \bar{y}_k$*

The proof follows directly from the KKT conditions and the definition of (MLP(\bar{x}^0, \bar{x}^1)).

This theorem completes the description of our algorithm for finding a cut for $C^{(e_k, \lceil \bar{x}_k \rceil)}$. It is summarized in Algorithm 1.

5 Cutting Plane Strategies and Practical Considerations

In the next section, we present preliminary computational testing of Algorithm 1. First, we discuss here strategies for using it. Note that we did not specify yet

Algorithm 1. Cut Generation Algorithm

- Input.** A convex set C , $\bar{x} \in C$ such that $\bar{x}_k \notin \mathbb{Z}$, and a split $(e_k, \lfloor \bar{x}_k \rfloor)$.
- I. **Testing if $\bar{x} \in C^{(e_k, \lfloor \bar{x}_k \rfloor)}$.**
 Solve (MNLP'), if the objective value is non-negative **STOP**, otherwise let \bar{y} be the optimal solution and **go to Step II**.
 - II. **Computing the cut.**
 Let $\bar{x}^1 := \frac{\bar{y}}{f_0}$ and $\bar{x}^0 = \frac{\bar{x} - \bar{y}}{1 - f_0}$. Solve (MLP(\bar{x}^0, \bar{x}^1)), build a cut $\alpha^T x \geq \beta$ from the dual multipliers and strengthen it. **Return the strengthened cut $\alpha^T x \geq \beta$.**
-

how to choose the point to cut \bar{x} and the elementary disjunctions. We test two strategies that were previously proposed and extensively used in the context of MILP. We also discuss here the problem of cut validity.

5.1 Separation by Rounds

Our first strategy consists in separating cuts by rounds following the approach first proposed in [5]. Let \bar{x} be the optimal solution of the continuous relaxation of (MICP). If $\bar{x} \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$ the problem is solved and we stop. Otherwise, a round of cuts consists in applying Algorithm 1 for each $k \in \{1, \dots, p\}$ such that $\bar{x}_k \notin \mathbb{Z}$. At the end of the round all the cuts found are added to the formulation of (MICP). The process is then iterated recursively with the strengthened formulation of (MICP).

It is important to note that in this approach, the sub-problems (MNLP') and (MLP(\bar{x}^0, \bar{x}^1)) are augmented in each new round with the cuts found in previous rounds. This approach has desirable properties: by Lemma 2 and Theorem 1 if \bar{x} is an extreme point of C , it is guaranteed that cuts will be found. As noted in the context of MILP it can also have undesirable consequences. First increasing the rank of the cuts at each iteration may lead to numerical difficulties and invalid cuts. Second increasing the sizes of (MNLP') and (MLP(\bar{x}^0, \bar{x}^1)) can make them very difficult to solve. In practice, we only do a very limited number of rounds. We test this approach with 1 and 10 rounds of cuts.

5.2 Rank-1 Optimization

In this other strategy, we limit the rank of the cut we generate to 1. Essentially, this means that we apply the same procedure as before except that the cuts found are never added to (MNLP') and (MLP(\bar{x}^0, \bar{x}^1)). Of course, this means that contrary to before the procedure may stop with $\bar{x} \notin \mathbb{Z}^p \times \mathbb{R}^{n-p}$ because no rank-1 cut exists anymore. Nevertheless, it has been experimentally observed in the context of MILP that this strategy sometimes allows to close more gap faster [15,11]. It also gives a feeling of the strength of the closure obtained by using only rank 1 cuts.

An aspect that has to be dealt with in this strategy is to try to avoid solving too many (MNLP') that do not generate cuts. To deal with this, we follow the strategies proposed in [11].

Finally, although this algorithm may eventually stop because no rank-1 cut can be generated anymore, it might take a very large amount of time and iterations. Here, we limit the computations to 10 minutes of CPU time and 100 rounds.

5.3 Cut Validity and Stability

The problem of generating inequalities that cuts off integer feasible solutions is a well known issue in MILP. It should be expected to be even more acute when some constraints are nonlinear. Here, we make a few observations on our experience of dealing with this issue during the experiments presented in the next section. First, let us state that we can not guarantee that all the cuts that we generate are valid. The only thing that we can say is that the integer optimal solutions were never cut during our experiments.

We actually do not believe our approach to be more prone to generating invalid cuts than other algorithms for (MICP). Note that the cuts that we generate are solely computed through a linear outer approximation of (MICP). In particular they do not depend on the precision of the solution of (MNLP'). As long as the outer approximation $P(\bar{x}^0, \bar{x}^1)$ is built with care (in practice, as is customary, we try to avoid difficulties there by slightly relaxing each constraint), the approach is not more dangerous than any other Outer Approximation based algorithm.

This does not mean that we did not experience troubles. Most of the difficulties we experienced came from numerical stability issues when solving the continuous relaxation of (MICP) augmented with many cuts. We used the solver filterSQP [21] to solve this NLP and experienced that, as it grows larger, it can rapidly become unsolvable (note that filterSQP is an active set method, it is well known that such methods may not be best suited when problems sizes grow). To avoid those difficulties, we had to take drastic measures in rejecting some cuts based on their numerical properties. A criterion that seemed to play a particularly important role is the ratio between the largest and smallest absolute values of the non-zero coefficients of a cut. It is usual in MILP to accept cuts until this ratio is as high as 10^8 without experiencing much difficulties (see for example [28]). Here, to stand clear of numerical troubles, we had to reject all cuts for which this ratio was greater than 10^4 . In the rank 1 experiments, we also had to clean (MICP) from any inactive cuts regularly (those cuts were kept in a pool in case they would be needed later on).

6 Computational Testing

Algorithm 1 and the two strategies outlined in Sect. 5.1 and 5.2 were implemented in C++ using the interfaces of the Bonmin [13,12] framework from COIN-OR [26]. In particular, our code allows to use any LP solver with an OSI [26] interface to solve (MLP(\bar{x}^0, \bar{x}^1)) and any NLP solver with a TNLP interface [32] to solve (MNLP'). Here we used CPLEX 12.1 and FilterSQP respectively. All experiments were conducted on a machine equipped with Intel Quad Core Xeon

2.93GHz processors and 120 GiB of RAM, using only one thread for each run. The test set consists of instances collected from different sources [24,16,30,25] (see within [14] for references of each instances types). Among about 150 instances collected, we selected all instances that took more than 1000 nodes to solve with a basic branch-and-bound algorithm. We then removed all instances that had a nonlinear objective and all instances that could not be solved in three hours with any of the methods tested here. As a result we obtain a test set of 80 instances. As noted in the introduction, instances with a nonlinear objective can be put into the form of (MICP) and therefore treated in a systematical way by our approach. Nevertheless, we chose to exclude them because we are unsure if this is the correct treatment for nonlinear objectives (it posed some stability issues in practice) and we felt there were enough interesting instances to report without them.

6.1 Gap Closed

In this first experiment we used the various cut generation strategies to only strengthen the continuous relaxations. The three strategies tested were 1 round of cuts, 10 rounds of cuts, and the rank 1 optimization. Our goal was to measure the CPU time of each method and the percentage of the integrality gap it closes¹. The results are summarized in Table 1.

Table 1. Number of instances in each class and then for each method and each instances class: averages cut generation times in seconds, average number of cuts and percentage of gap closed

Type	#	1 round			10 rounds			rank 1		
		CPU	# cuts	gap	CPU	# cuts	gap	CPU	# cuts	gap
Batch	4	2.85	24.25	20.64	40.65	89.5	24.01	60.08	112.75	24.06
CLay	7	1.64	40.14	1.55	16.16	219.71	8.55	39.71	152.29	33.71
FLay	5	0.35	29.75	1.25	7.74	225.75	3.85	48.19	400.5	34.98
RSyn	16	9.15	64.63	21.32	107.77	406.13	47.61	247.81	584.94	59.96
SLay	9	12.92	82.33	4.60	182.76	254.56	7.58	230.25	207.44	29.65
Syn	18	3.78	58.22	23.58	92.03	357.67	56.36	110.91	622.67	84.52
fo-m	5	0.27	32.4	0.00	7.93	139.8	0.00	12.79	283.2	0.00
nd	3	2.31	22.33	6.32	33.95	52.67	11.05	251.79	312.67	69.16
sssd	12	0.12	24.92	3.75	3.42	219.83	43.12	1.85	141.67	97.97
trimloss	1	0.27	11	2.70	3.35	106	12.15	10.02	96	12.32
all instances	80	4.57	48.99	12.18	69.07	276.01	32.30	120.08	377.24	58.05

The results shows that our various methods are able to close significant portions of the integrality gap in reasonable computing times. In particular, on average on our test set, the rank 1 approach closes 58% of the gap in about two minutes.

¹ If z_C is the optimal value of the initial continuous relaxation of (MICP), z_X is the optimal value of (MICP) and z_S is the optimal value of the continuous relaxation strengthened by cutting planes, the percentage of integrality gap closed is $100 \left(1 - \frac{z_X - z_S}{z_X - z_C}\right)$.

6.2 Complete Resolutions

In this second experiment we want to assess if the cuts can be useful to effectively solve the test instances to optimality faster. To this end, after the cut generation procedure is finished, we keep in the formulation all cuts that are tight at the new continuous optimum and solve the formulation with Bonmin's NLP branch-and-bound algorithm B-BB. We then compare the solution process with B-BB without cuts. The results are reported in Table 2.

Table 2. For each method and each instances type: number of instances solved (#I), average total solution times (include cut generation) and branch-and-bound nodes (averages figures taken only on the subsets of instances solved by all methods)

Type	no cuts			1 round			10 rounds			rank-1		
	#I	CPU	nodes	#I	CPU	nodes	#I	CPU	nodes	#I	CPU	nodes
Batch	4	101.6	1681	4	75.2	1069	4	127.6	1316	4	137.2	1110
Clay	7	106.3	13435	7	117.1	10727	7	171.5	11616	7	199.0	10875
Flay	5	1175.5	51773	4	1190.7	48522	4	1260.8	49964	4	1979.3	55611
Rsyn	10	2689.5	38914	11	2378.8	19422	12	1535.0	6082	16	1368.1	988
Slay	9	152.2	1455	9	177.4	1395	9	329.8	1274	9	392.5	1055
Syn	12	546.9	21476	12	710.2	21534	13	310.4	5309	18	94.5	54
fo-m	4	2271.2	315700	4	2019.4	278629	4	2017.5	209537	5	2764.0	263231
nd	3	1397.3	7946	3	1436.0	7763	3	1427.1	6821	3	524.5	527
sssd	8	362.8	69713	8	115.9	19207	6	200.4	34544	10	207.7	36877
tls	1	2071.1	1298667	1	1998.5	1092360	1	3636.2	1336885	1	4889.5	1517275
solved	63	980	66130	63	928	51946	63	776	47299	77	792	52190

The results show that the cuts are indeed helping to solve more instances faster. The rank-1 procedure can solve 77 instances, when all others solve only 63. On instances solved by all approaches, rank-1 and 10 rounds are the two most competitive. It should be noted that the usefulness of the cuts varies considerably between different types of instances. Cuts are particularly useful for solving the RSyn, Syn, nd and sssd instances. On the other hand they seem to only slow down the solution of CLay, FLay, SLay and trimloss.

Acknowledgments. Research supported by ANR grant ANR06-BLAN-0375 and by a Google Focused Research Award.

References

1. Abhishek, K., Leyffer, S., Linderoth, J.: FilMINT: An Outer Approximation-Based Solver for Convex Mixed-Integer Nonlinear Programs. *INFORMS Journal on Computing* 22, 555–567 (2010)
2. Atamtürk, A., Narayanan, V.: Conic mixed integer rounding cuts. *Mathematical Programming* 122, 1–20 (2010)
3. Balas, E.: Disjunctive programming: Properties of the convex hull of feasible points. *Discrete Applied Mathematics* 89, 3–44 (1988); (originally MSRR # 348, Carnegie Mellon University, July 1974)

4. Balas, E., Bonami, P.: Generating lift-and-project cuts from the LP simplex tableau: open source implementation and testing of new variants. *Mathematical Programming Computations* 1, 165–199 (2009)
5. Balas, E., Ceria, S., Cornuéjols, G.: A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Math. Programming* 58, 295–324 (1993)
6. Balas, E., Ceria, S., Cornuéjols, G., Natraj, N.: Gomory cuts revisited. *Operations Research Letters* 19, 1–9 (1996)
7. Balas, E., Jeroslow, R.G.: Strengthening cuts for mixed integer programs. *European J. Oper. Res.* 4(4), 224–234 (1980)
8. Balas, E., Perregaard, M.: Lift and project for mixed 0-1 programming: Recent progress. *Discrete Applied Mathematics* 123(1-3), 129–154 (2002)
9. Balas, E., Perregaard, M.: A precise correspondence between lift-and-project cuts, simple disjunctive cuts, and mixed integer Gomory cuts for 0-1 programming. *Math. Program* 94(2-3, Ser. B), 221–245 (2003); *The Aussois 2000 Workshop in Combinatorial Optimization*
10. Balas, E., Zemel, E.: Facets of the knapsack polytope from minimal covers. *SIAM Journal on Applied Mathematics* 34, 119–148 (1978)
11. Bonami, P.: On optimizing over lift-and-project closures. Research Report HAL, CNRS (October 2010), <http://hal.archives-ouvertes.fr/hal-00529816/en/>
12. Bonami, P., Biegler, L.T., Conn, A.R., Cornuéjols, G., Grossmann, I.E., Laird, C.D., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. *Discrete Optimization* 5(2), 186–204 (2008)
13. Bonami, P., Forrest, J.J.H., Laird, C., Lee, J., Margot, F., Wächter, A.: BONMIN: Basic Open-source Nonlinear Mixed INteger programming (July 2006), <http://www.coin-or.org/Bonmin>
14. Bonami, P., Kılınç, M., Linderoth, J.: Algorithms and Software for Convex Mixed Integer Nonlinear Programs. Technical report, Technical Report #1664, Computer Sciences Department, University of Wisconsin-Madison (2009)
15. Bonami, P., Minoux, M.: Using rank-1 lift-and-project closures to generate cuts for 0–1 MIPs, a computational investigation. *Discrete Optimization* 2(4), 288–307 (2005)
16. Bussieck, M.R., Drud, A.S., Meeraus, A.: MINLPLib – A collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing* 15(1) (2003)
17. Ceria, S., Soares, J.: Convex programming for disjunctive optimization. *Mathematical Programming* 86, 595–614 (1999)
18. Cezik, M.T., Iyengar, G.: Cuts for mixed 0-1 conic programming. *Mathematical Programming* 104, 179–202 (2005)
19. Fischetti, M., Lodi, A., Tramontani, A.: On the separation of disjunctive cuts. *Mathematical Programming* (2009), doi: 10.1007/s10107-009-0300-y (in press)
20. Fischetti, M., Salvagnin, D.: An in-out approach to disjunctive optimization. In: Lodi, A., Milano, M., Toth, P. (eds.) CPAIOR 2010. LNCS, vol. 6140, pp. 136–140. Springer, Heidelberg (2010)
21. Fletcher, R., Leyffer, S.: User manual for filterSQP, University of Dundee Numerical Analysis Report NA-181 (1998)
22. Gomory, R.E.: An algorithm for integer solution solutions to linear programming. In: Graves, R.L., Wolfe, P. (eds.) *Recent Advances in Mathematical Programming*, pp. 269–302. McGraw-Hill, New York (1963)

23. Kılınç, M., Linderoth, J., Luedtke, J.: Effective separation of disjunctive cuts for convex mixed integer nonlinear programs. Technical Report Computer Sciences Department, University of Wisconsin-Madison (2010)
24. Leyffer, S.: MacMINLP: Test problems for mixed integer nonlinear programming (2003), <http://www.mcs.anl.gov/~leyffer/macminlp>
25. Linderoth, J., Kılınç, M.: Personal communication (2010)
26. Lougee-Heimer, R.: The common optimization interface for operations research. IBM Journal of Research and Development 47, 57–66 (2003), <http://www.coin-or.org>
27. Marchand, H., Wolsey, L.A.: Aggregation and mixed integer rounding to solve MIPs. Operations Research 49(3), 363–371 (2001)
28. Margot, F.: Testing cut generators for mixed integer linear programming. Mathematical Programming Computation 1, 69–95 (2009)
29. Perregaard, M.: Generative Disjunctive Cuts for Mixed Integer Programs. PhD thesis, Carnegie Mellon University (2003)
30. 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 (2006), <http://egon.cheme.cmu.edu/ibm/page.htm>
31. Stubbs, R., Mehrotra, S.: A branch-and-cut method for 0-1 mixed convex programming. Mathematical Programming 86, 515–532 (1999)
32. Wächter, A., Laird, C.D., Kawajir, Y.: Introduction to IPOPT: A tutorial for downloading, installing, and using IPOPT (2010), <http://www.coin-or.org/Ipopt/documentation/>