Mathematical Programming Computation (2019) 11:211–235
https://doi.org/10.1007/s12532-018-0146-5

FULL LENGTH PAPER

The (not so) trivial lifting in two dimensions

Ricardo Fukasawa[1] · Laurent Poirrier[1] · Álinson S. Xavier[2]

Received: 20 June 2017 / Accepted: 29 June 2018 / Published online: 19 September 2018

Abstract

When generating cutting-planes for mixed-integer programs from multiple rows of the simplex tableau, the usual approach has been to relax the integrality of the non-basic variables, compute an intersection cut, then strengthen the cut coefficients corresponding to integral non-basic variables using the so-called trivial lifting procedure. Although of polynomial-time complexity in theory, this lifting procedure can be computationally costly in practice. For the case of two-row relaxations, we present a practical algorithm that computes trivial lifting coefficients in constant time, for arbitrary maximal lattice-free sets. Computational experiments confirm that the algorithm works well in practice.

Keywords Integer programming · Lifting · Cutting planes

Mathematics Subject Classification 90C11 · 90C57

The software that was reviewed as part of this submission was given the DOI (Digital Object Identifier) https://doi.org/10.5281/zenodo.1342770

The authors were partially supported by NSERC Discovery Grant RGPIN 2014-05623. Xavier and Poirrier were partially supported by Ontario Early Researcher Award ER 11-08-174. Poirrier was partially supported by NSERC Discovery Grant RGPIN 2018-04335.

✉ Álinson S. Xavier
axavier@anl.gov

Ricardo Fukasawa
rfukasawa@uwaterloo.ca

Laurent Poirrier
lpoirrier@uwaterloo.ca

1 Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada

2 Energy Systems Division, Argonne National Laboratory, Lemont, USA

1 Introduction

A recent topic of interest in the theory of general-purpose cutting planes for mixed-integer linear programming (MIP) has been cutting planes that can only be obtained by considering multiple rows of a simplex tableau simultaneously (see for instance [1,5,11]). More specifically, researchers have been interested in producing strong valid inequalities for the *corner relaxation* of a simplex tableau [14], defined as

$$\begin{aligned} x &= f + \sum_{r \in R} r s_r + \sum_{w \in W} w y_w \\ x &\in \mathbb{Z}^m \\ s_r &\in \mathbb{R}_+ \quad \forall r \in R \\ y_w &\in \mathbb{Z}_+ \quad \forall w \in W, \end{aligned} \tag{1}$$

where $f \in \mathbb{R}^m \setminus \mathbb{Z}^m$, $R \subseteq \mathbb{R}^m$ and $W \subseteq \mathbb{R}^m$. In this relaxation, the decision variables x, s and y correspond, respectively, to integral basic variables, continuous non-basic variables and integral non-basic variables, while f, R and W correspond to the right-hand side and to the columns of the tableau.

One approach to obtain valid inequalities for (1) is via *lifting*, a technique first introduced by Padberg [24] for a specific combinatorial problem and later generalized to other polyhedra (see Dey and Wolsey [11]). In this approach, a strong valid inequality for the following simplified model is first computed and then transformed into a strong valid inequality for (1):

$$\begin{aligned} x &= f + \sum_{r \in R} r s_r \\ x &\in \mathbb{Z}^m \\ s_r &\in \mathbb{R}_+ \quad \forall r \in R. \end{aligned} \tag{2}$$

Note that (2) is the model obtained from (1) by fixing all the integral non-basic variables in (1) to zero. This model is considerably simpler than (1) and, following a seminal paper by Andersen et al. [1], has received much attention in the last decade (see [8, Chapter 6] for a survey). It is well known that valid inequalities for (2) can be obtained from convex lattice-free sets in $\mathbb{R}^m$ (see Balas [3]). More specifically, if $B \subseteq \mathbb{R}^m$ is a convex lattice-free set containing f in its interior and $\psi : \mathbb{R}^m \to \mathbb{R}$ is the gauge function of the convex set $B - f$, then the inequality

$$\sum_{r \in R} \psi(r) s_r \geq 1 \tag{3}$$

is valid for (2). Furthermore, every non-trivial valid inequality for (2) can be written as (3), where ψ is the gauge function of some convex lattice-free set (see Borozan and Cornuéjols [7]). Obtaining strong valid inequalities for (2), therefore, can be done in practice by generating convex lattice-free sets in low dimensions.

In this paper, we focus on the practical problem of lifting valid inequalities for (2) into strong valid inequalities for (1). More specifically, given a valid inequality for (2) written as (3), we want to obtain a function $\pi : \mathbb{R}^m \to \mathbb{R}$ such that

$$\sum_{r \in R} \psi(r) s_r + \sum_{w \in W} \pi(w) y_w \geq 1 \tag{4}$$

is a valid inequality for (1).

In theory, the best possible sets of lifting coefficients can be computed by optimizing over the polar set of (1). This approach has been applied to corner relaxations by Louveaux et al. [22] in order to measure the strength of their different variations. However, it involves enumerating a subset of the points of (1), which is computationally expensive and cannot be expected to perform well in a general-purpose cut generator.

An alternative method, which is computationally cheaper, is to compute the so-called *trivial lifting coefficient*. Given a valid inequality (3), Gomory and Johnson [15] and Balas and Jeroslow [4] proved that, if π is defined as

$$\pi(w) = \inf_{k \in \mathbb{Z}^n} \psi(w + k), \tag{5}$$

then (4) is valid inequality for (1). Such function π is called the *trivial lifting* of ψ. Under the assumption that (4) must be a valid inequality for any choice of R and W, Dey and Wolsey [11] and Basu et al. [6] proved that, in some particular cases, any other definition of π leads to weaker inequalities. In other words, in those cases, the computationally cheaper trivial lifting actually yields the best possible lifting coefficients. Moreover, even when this is not the case, one important advantage of the trivial lifting is that it is *sequence independent*, which means that the order in which the coefficients are computed is irrelevant. The idea of potentially losing coefficient strength in order to obtain a more computationally tractable sequence-independent lifting has been applied to other valid inequalities for MIPs, like flow cover inequalities [16].

Despite its name, evaluating the trivial lifting function is far from trivial, particularly if put into the context of where the problem arises: it is solved once for every integer variable that needs to be lifted within a cut, so potentially thousands of times per cut. In addition, if one thinks that several cuts are to be generated and that the cutting-plane generation is just one small step in the whole solution process of a MIP, this can quickly become an impractical problem to solve.

A naive approach to solve (5) is to evaluate $\psi(w + k)$ for every $k \in \mathbb{Z}^n$ such that $\|k\|$ is smaller than a fixed constant, chosen before the cut generation procedure starts. For the two-row case, it has been proven that, if the constant is large enough, then this procedure finds the correct answer [11]. Alternatively, in one of the first computational experiments with multi-row cuts, instead of evaluating ψ at many points, Espinoza [12] evaluates this function exactly once, at a point selected heuristically, with no guarantee that the exact trivial lifting coefficient is obtained. This approach is computationally friendly, but produces inequalities that are not as strong as they could be. In later experiments, Dey and Wolsey [9,11] and Basu et al. [5] carefully choose the convex lattice-free sets that give rise to inequality (4), so that the trivial lifting function can be evaluated by using a closed formula. This approach, however, limits the range of lattice-free sets that can be experimented with. Finally, it is worth mentioning that, in principle, this problem can be solved in polynomial time for fixed n, since it is a minimization of a convex function over integer points in polyhedra [23], or modeled as a mixed-integer program by adding a continuous variable to handle the convex piecewise linear objective function which leads to another polynomial time algorithm for fixed n [2]. Such approaches, however, rely on the solution of the feasibility problem in fixed dimension as a subroutine to solve the optimization problem. Solving one such MIP for each cut coefficient would likely suffer from significant computational overhead, even if a fast fixed-dimension MIP solver implementation were available.

In this paper, we develop a more practical method for evaluating the trivial lifting function in two dimensions, which requires significantly fewer queries to the function ψ than the naive procedure and that does not have significant computational overhead. For maximal lattice-free sets, we prove that the algorithm is guaranteed to terminate in constant time, and for the cases where the closed formula described in [5] is applicable, we show that it requires the same number of evaluations to the function ψ as the closed formula. We also obtain an upper bound on the number of evaluations for non-maximal lattice-free sets, which depends on the lattice-width of the set and on its second covering minimum. Finally, we run computational experiments to confirm that the algorithm works well in practice. The proposed method can also easily be adapted to solve the lifting problem in higher dimensions, though in that context we do not have theoretical or computational evaluations of its performance.

The main idea of the method is to compute a unimodular transformation that is applied as a preprocessing step to the lattice-free set, which is then combined with a simple algorithm based on a one-dimensional convex optimization problem. We note that the idea is similar in spirit to Lenstra's algorithm for fixed-dimension IP [20], although our proposed transformation relies solely on very simple arithmetic calculation as opposed to computing ellipsoids. In addition, the transformed lattice-free set has no guarantee of "roundness" as is required by Lenstra's algorithm. It is only guaranteed not to be too wide in one direction. The simplicity of the algorithm is an important component on making its computational overhead very small (on the order of 0.1 ms runtime).

2 Main algorithm

In this section, we describe an algorithm for the computation of trivial lifting coefficients on two-dimensional lattice-free sets. More specifically, let $B \subseteq \mathbb{R}^2$ be a convex lattice-free set containing $f \in \mathbb{R}^2$ in its interior, and suppose $\psi : \mathbb{R}^2 \to \mathbb{R}$ is the gauge function of $B - f$, i.e. $\psi(r) = \inf\{t > 0 : f + \frac{1}{t}r \in B\}$. For any $w \in \mathbb{R}^2$, our goal is to solve the minimization problem (5).

The following propositions give us the two main ideas behind the algorithm. Recall that $k \in \mathbb{Z}^2$ is the variable in the minimization problem (5). The first proposition shows that, if one component of k is fixed at any value (without loss of generality, we fix k_2), then the minimization problem becomes significantly easier.

Proposition 1 *Let $\bar{k}_2 \in \mathbb{R}$. If k_1^* is a solution for*

$$\min_{k_1 \in \mathbb{R}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \tag{6}$$

then a solution for

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \tag{7}$$

is either $\lfloor k_1^* \rfloor$ or $\lceil k_1^* \rceil$.

Proof This follows directly from the fact that ψ is a univariate convex function. □

Therefore, given a solution for the continuous problem (6), we can easily determine an optimal solution for the integer problem (7). Note, however, that (6) can be solved efficiently, for example by modeling it as an LP if B is polyhedral. Since these ideas will be used throughout, it will be convenient to define the following notation

$$g(\bar{\alpha}_2) := \min_{\alpha_1 \in \mathbb{R}} \psi \begin{pmatrix} \alpha_1 \\ \bar{\alpha}_2 \end{pmatrix}$$
$$h(\bar{k}_2) := \min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix}.$$

The second proposition shows that, if the second component of k is fixed at a number with very large magnitude, either positive of negative, then the optimal value also becomes very large. Therefore, these values of k_2 may be safely ignored. Note that the constants ζ^+ and ζ^- that appear on the statement of the proposition do not depend on w or k, but only on the definition of the function ψ.

Proposition 2 *If $\bar{k}_2$ is a positive integer such that $\bar{k}_2 > |w_2|$, then*

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \geq \zeta^+(w_2 + \bar{k}_2)$$
$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2-\bar{k}_2 \end{pmatrix} \geq \zeta^-(\bar{k}_2 - w_2)$$

where $\zeta^+ = \min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ 1 \end{pmatrix}$ and $\zeta^- = \min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ -1 \end{pmatrix}$.

Proof First, note that

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \geq \min_{k_1 \in \mathbb{R}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} = \min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ w_2+\bar{k}_2 \end{pmatrix}$$

since the integer problem is a restriction of the continuous one, and we can let $\alpha := w_1 + k_1$. Then, because ψ is positively homogeneous and $w_2 + \bar{k}_2 > 0$, we have

$$\min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ w_2+\bar{k}_2 \end{pmatrix} = (w_2 + \bar{k}_2) \min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ 1 \end{pmatrix} = \zeta^+(w_2 + \bar{k}_2).$$

To obtain the second inequality, we proceed similarly. Since $\bar{k}_2 - w_2 > 0$, we have

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2-\bar{k}_2 \end{pmatrix} \geq \min_{\alpha \in \mathbb{R}} \psi \begin{pmatrix} \alpha \\ w_2-\bar{k}_2 \end{pmatrix} = \zeta^-(\bar{k}_2 - w_2).$$

Given B, f and w, the function TrivialLifting described in Algorithm 3 computes the optimum value of (5). At each iteration, it solves the two optimization problems

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \text{ and } \min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2-\bar{k}_2 \end{pmatrix}$$

for some fixed value $\bar{k}_2$, starting from zero, and going up. By Proposition 1, these two problems can be easily solved. The algorithm also keeps track of the smallest optimal value found so far, in the variable η^*. It stops when $\bar{k}_2$ is such that three conditions are satisfied:

$$\bar{k}_2 > |w_2| \textbf{ and } w_2 + \bar{k}_2 > \frac{\eta^*}{\zeta^+} \textbf{ and } \bar{k}_2 - w_2 > \frac{\eta^*}{\zeta^-} \tag{8}$$

This is justified by Proposition 2. Indeed, if $\bar{k}_2$ is such that condition (8) holds, then

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix} \geq \zeta^+(w_2 + \bar{k}_2) \geq \zeta^+ \frac{\eta^*}{\zeta^+} = \eta^*$$

$$\min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2-\bar{k}_2 \end{pmatrix} \geq \zeta^-(\bar{k}_2 - w_2) \geq \zeta^- \frac{\eta^*}{\zeta^-} = \eta^*$$

Therefore, by considering any such $\bar{k}_2$, the incumbent value η^* can never be improved. Also note that, if $\bar{k}_2$ is sufficiently large, then condition (8) is automatically satisfied. Indeed, since $\eta^* \geq h(0)$, then

$$\bar{k}_2 > \max \left\{ \frac{h(0)}{\zeta^+} - w_2, \frac{h(0)}{\zeta^-} + w_2, |w_2| \right\}$$

implies that the condition holds. Therefore, the algorithm will always terminate.

Algorithm 3 Trivial Lifting

function TrivialLifting(B, f, w)
  Let us denote $g(\bar{\alpha}_2) := \min_{\alpha_1 \in \mathbb{R}} \psi \begin{pmatrix} \alpha_1 \\ \bar{\alpha}_2 \end{pmatrix}$
  Let us denote $h(\bar{k}_2) := \min_{k_1 \in \mathbb{Z}} \psi \begin{pmatrix} w_1+k_1 \\ w_2+\bar{k}_2 \end{pmatrix}$
  $\zeta^+, \zeta^- \leftarrow g(1), g(-1)$
  $\eta^* \leftarrow h(0)$
  $\bar{k}_2 \leftarrow 1$
  repeat
    $\eta^* \leftarrow \min \{\eta^*, h(\bar{k}_2), h(-\bar{k}_2)\}$
    $\bar{k}_2 \leftarrow \bar{k}_2 + 1$
  until $\left(\bar{k}_2 > |w_2| \textbf{ and } w_2 + \bar{k}_2 > \frac{\eta^*}{\zeta^+} \textbf{ and } \bar{k}_2 - w_2 > \frac{\eta^*}{\zeta^-}\right)$
  return η^*

3 Preprocessing step

Although finite, the algorithm described in Sect. 2 may require a large number of iterations to terminate. In this subsection, we describe a preprocessing step that, when executed prior to the algorithm, greatly increases its worst-case performance.

To illustrate how pre-processing can improve the efficiency of Algorithm 3, consider the following example.

Example 4 Let B and $w, f \in \mathbb{R}^2$ be defined as

$$B = \text{conv}\left\{\begin{pmatrix} 22 \\ \frac{69}{7} \end{pmatrix}, \begin{pmatrix} -3 \\ -\frac{11}{7} \end{pmatrix}, \begin{pmatrix} -8 \\ -\frac{26}{7} \end{pmatrix}\right\} \quad f = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{6} \end{pmatrix} \quad w = \begin{pmatrix} \frac{2}{3} \\ \frac{1}{3} \end{pmatrix}$$

The set B, illustrated in Fig. 1a, is very long and thin, which causes performance problems for both the naive algorithm (i.e., searching all points in a predetermined box), as well as the algorithm described previously. In fact, Algorithm 3 requires seven iterations of the main loop to output the optimal value of $\frac{4}{5}$. Before feeding this data into the algorithm however, we could apply to B, w and f the affine unimodular transformation $\tau : \mathbb{R}^2 \to \mathbb{R}^2$ defined as

$$\tau(x) = \begin{pmatrix} 1 & -2 \\ -5 & 11 \end{pmatrix} x + \begin{pmatrix} 4 \\ 2 \end{pmatrix}.$$

Let $\bar{B}$, $\bar{f}$, $\bar{w}$ be the set and vectors obtained, namely

$$\bar{B} = \text{conv}\left\{\begin{pmatrix} \frac{16}{7} \\ \frac{3}{7} \end{pmatrix}, \begin{pmatrix} \frac{1}{7} \\ -\frac{2}{7} \end{pmatrix}, \begin{pmatrix} -\frac{4}{7} \\ \frac{8}{7} \end{pmatrix}\right\} \quad \bar{f} = \begin{pmatrix} \frac{1}{3} \\ \frac{1}{2} \end{pmatrix} \quad \bar{w} = \begin{pmatrix} \frac{0}{3} \\ \frac{1}{3} \end{pmatrix}.$$

Note that the integral part of $\bar{w}$ was discarded. The set $\bar{B}$, as Fig. 1b illustrates, is much smaller, and in particular, not very wide in the vertical direction. Feeding this new data into Algorithm 3, we obtain the same optimal value of $\frac{4}{5}$ as before, but now after a single iteration of the main loop. □

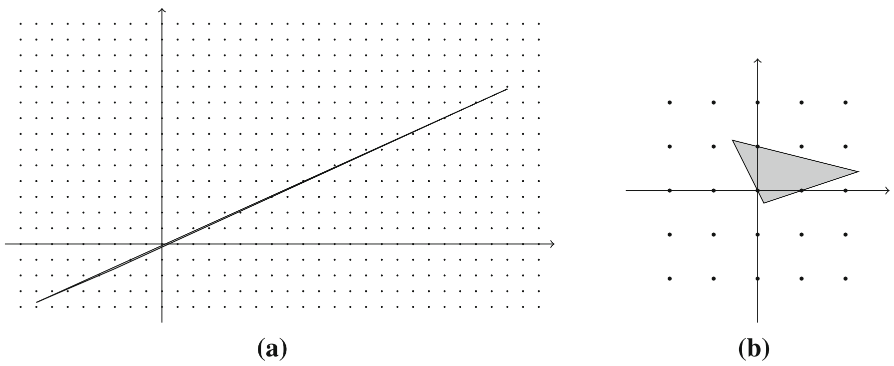

Fig. 1 Example of pre-processing. **a** Original set B, **b** modified set $\bar{B}$

In the following, we describe exactly what properties we would like the affine unimodular transformation τ to satisfy, and how such a transformation could be obtained for arbitrary maximal lattice-free sets. If $B \subseteq \mathbb{R}^2$ is a maximal lattice-free set, then B is either a split, a triangle or a quadrilateral. Following Dey and Louveaux [10], we further categorize maximal lattice-free triangles as follows:

Definition 5 [10]

(i) A type-1 triangle is a triangle with integral vertices and exactly one integral point in the relative interior of each facet (see Fig. 2b);
(ii) A type-2 triangle is a triangle with at least one fractional vertex v, exactly one integral point in the relative interior of two facets incident to v, and at least two integral points on the third facet (see Fig. 2c);
(iii) A type-3 triangle is a triangle with exactly three integral points on the boundary, one in the relative interior of each edge (see Fig. 2d).

Given $d \in \mathbb{R}^2$, we define the *width of* B *along* d as

$$\omega_d(B) = \max_{b\in B} d^T b - \min_{b\in B} d^T b.$$

First, we would like τ to be a transformation such that the width of $\tau(B)$ along the vertical direction $\binom{0}{1}$ is small. Algorithms 10 and 11 show how to obtain such τ when B is a maximal lattice-free triangle or quadrilateral, respectively. Both algorithms make use of the following lemma, which is implied by the proofs presented by Hurkens [17].

Lemma 6 [17]

(i) If B is a maximal lattice-free triangle such that $\binom{0}{0}, \binom{0}{1}, \binom{1}{0}$ are in the relative interiors of distinct faces of B, then there exists $d \in \left\{\binom{0}{1}, \binom{1}{0}, \binom{1}{1}\right\}$ such that

$$\omega_d(B) \le 1 + \frac{2}{3}\sqrt{3}.$$

(ii) If B is a maximal lattice-free quadrilateral such that $\binom{0}{0}, \binom{0}{1}, \binom{1}{0}$ are in the relative interiors of distinct faces of B, then there exists $d \in \left\{\binom{0}{1}, \binom{1}{0}\right\}$ such that

$$\omega_d(B) \le 2.$$

The bound given by Lemma 6 can be slightly improved when B is a maximal lattice-free triangle of type 1 or 2, as the next lemma shows.

Lemma 7 *If B is a maximal lattice-free triangle of type 1 or 2 such that $\binom{0}{0}, \binom{0}{1}, \binom{1}{0}$ are in the relative interiors of distinct faces of B, then there exists $d \in \left\{\binom{0}{1}, \binom{1}{0}, \binom{1}{1}\right\}$ such that*

$$\omega_d(B) \le 2.$$

Proof If B is a type-1 triangle, then B must be the triangle depicted in Fig. 2b. Clearly, $d = \binom{1}{0}$ satisfies the condition of the lemma. Now suppose that B is a type-2 triangle. We have three subcases, depending on which facet of B contains multiple lattice points.

For the first subcase, suppose that the facet of B that contains multiple lattice points is the one containing $\binom{1}{0}$ in its relative interior. In this case, the vertices of B are

$$(1, \alpha), (1, -\beta), \left(\frac{-1}{\alpha+\beta-1}, \frac{\beta}{\alpha+\beta-1}\right),$$

for some $\alpha, \beta \in \mathbb{R}_+$. If $\omega_{(1,0)}(B) \leq 2$, we are done. Suppose $\omega_{(1,0)}(B) > 2$. Then $\alpha + \beta > 2$, and we have

$$\omega_{(0,1)}(B) = 1 + \frac{1}{\alpha+\beta-1} \leq 1 + \frac{1}{2-1} = 2.$$

In any case, there exists $d \in \left\{\binom{0}{1}, \binom{1}{0}\right\}$ satisfying the condition of the lemma.

It is known that we can use a unimodular transformation to reduce the other two subcases to the first one [11]. We now determine this unimodular transformation exactly, since we will need it to specify Algorithm 10.

For the second subcase, suppose that the facet of B that contains multiple lattice points is the one containing $\binom{0}{1}$. Let $\bar{B}$ be defined as

$$\bar{B} = \left\{\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} b : b \in B\right\}$$

Clearly, $\bar{B}$ satisfies the conditions of the first subcase. Therefore, there exists $\bar{d} \in \left\{\binom{0}{1}, \binom{1}{0}\right\}$ such that $\omega_{\bar{d}}(\bar{B}) \leq 2$, which implies that there exists $d \in \left\{\binom{1}{0}, \binom{0}{1}\right\}$ such that $\omega_d(B) \leq 2$.

Finally, for the third subcase, suppose that the facet of B that contains multiple lattice points is the one containing $\binom{0}{0}$. We proceed similarly to the previous subcase. Let $\bar{B}$ be defined as

$$\bar{B} = \left\{\begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix} b + \begin{bmatrix} 0 \\ 1 \end{bmatrix} : b \in B\right\}$$

Note that $\bar{B}$ is a lattice-free triangle, since the transformation is unimodular. Also, the point $\binom{0}{0}$ is mapped to $\binom{0}{1}$. Therefore, $\bar{B}$ satisfies the conditions of the second subcase, and there exists $\bar{d} \in \left\{\binom{1}{0}, \binom{0}{1}\right\}$ such that $\omega_{\bar{d}}(\bar{B}) \leq 2$. We conclude that there exists $d \in \left\{\binom{1}{0}, \binom{1}{1}\right\}$ such that $\omega_d(B) \leq 2$. □

Propositions 8 and 9 show that when Algorithm 10 or 11 is applied to an arbitrary maximal lattice-free triangle or quadrilateral, it produces a transformed set that satisfies the conditions to apply Lemmas 6 and 7. Specifically, our proposed preprocessing step generates lattice-free sets $\bar{B}$ that have the points $\left\{\binom{0}{0}, \binom{1}{0}, \binom{0}{1}\right\}$ in the relative interiors of distinct faces of B, and a small width $\omega_d(\bar{B})$ for $d = \binom{0}{1}$.

Note that in the preprocessing step, we also transform the given vector w (corresponding to the variable to be lifted) in a specific way, resulting in points (ii) of Propositions 8 and 9, which we will exploit in Sect. 4. In this process, we introduce the variable ε, which is the middle point of $\bar{B}$ along the vertical coordinate x_2.

Proposition 8 *Let $B \subseteq \mathbb{R}^2$ be a maximal lattice-free triangle containing $f \in \mathbb{R}^2$ in its interior, and let $w \in \mathbb{R}^2$. Suppose $v^1, v^2, v^3 \in \mathbb{Z}^2$ are lattice points in the relative interiors of three distinct facets of B. If $\bar{B}$, $\bar{f}$ and $\bar{w}$ are the values returned by Algorithm 10, and if λ is the width of $\bar{B}$ along the vertical direction, then:*

(i) $\lambda \leq 1 + \frac{2}{3}\sqrt{3}$ if B is a type-3 triangle, and $\lambda \leq 2$ otherwise.
(ii) $|\bar{f}_2 + \bar{w}_2 - \bar{b}_2| \leq \frac{\lambda+1}{2}$ for all $\bar{b} \in \bar{B}$.

Proof (i) It is clear that τ^1 is an affine unimodular function that maps v^1, v^2, v^3 to the points $\binom{0}{0}, \binom{1}{0}, \binom{0}{1}$, respectively. Furthermore, $\tau^1(B)$ satisfies the conditions for item (i) of Lemma 6, so there exists d such that $\omega_d(\tau^1(B)) \leq 1 + \frac{2}{3}\sqrt{3}$. Additionally, by Lemma 7, if B is a maximal lattice-free triangle of types 1 or 2, then $\omega_d(\tau^1(B)) \leq 2$. If $d = \binom{0}{1}$, then the direction that minimizes the width of $\tau^1(B)$ is already the vertical direction. In that case, $\tau^2 = \tau^1$, and we are done. If $d = \binom{1}{0}$, then the direction that minimizes the width of $\tau^1(B)$ is the horizontal direction. To obtain τ^2, the algorithm composes τ^1 with a transformation that flips the two coordinates. Finally, if $d = \binom{1}{1}$, then the direction that minimizes the width of $\tau^1(B)$ is perpendicular to the line connecting $\binom{1}{0}$ and $\binom{0}{1}$. To obtain τ^2, the algorithm composes τ^1 with a transformation that maps $\binom{0}{0}, \binom{0}{1}, \binom{1}{0}$ to $\binom{0}{1}, \binom{0}{0}, \binom{1}{0}$, respectively. The direction that minimizes the width of $\tau^2(B)$, therefore, is perpendicular to the line that connects $\binom{0}{0}, \binom{1}{0}$, which is the vertical direction, as desired.

(ii) Let $\varepsilon := \frac{\max\{b_2:(b_1,b_2)\in B\}+\min\{b_2:(b_1,b_2)\in B\}}{2}$. By definition, $|\varepsilon - \bar{b}_2| \leq \frac{\lambda}{2}$, for all $\bar{b} \in \bar{B}$. We also claim that $|\bar{f}_2 + \bar{w}_2 - \varepsilon| \leq \frac{1}{2}$. This is true, since, for every $x \in \mathbb{R}$, if we let $\bar{x} \leftarrow x + \lfloor \varepsilon + \frac{1}{2} - x \rfloor$ then $|\bar{x} - \varepsilon| \leq \frac{1}{2}$, and it is the exact transformation applied to w_2' in the algorithms. The result then follows, since, for every $\bar{b} \in \bar{B}$, we have:

$$\begin{aligned}|\bar{f}_2 + \bar{w}_2 - \bar{b}_2| &= |\bar{f}_2 + \bar{w}_2 - \varepsilon + \varepsilon - \bar{b}_2| \\ &\leq |\bar{f}_2 + \bar{w}_2 - \varepsilon| + |\varepsilon - \bar{b}_2| \\ &\leq \frac{1}{2} + \frac{\lambda}{2}\end{aligned}$$

□

Proposition 9 *Let $B \subseteq \mathbb{R}^2$ be a maximal lattice-free quadrilateral with $f \in \mathbb{R}^2$ in its interior, and let $w \in \mathbb{R}^2$. Suppose $v^1, \ldots, v^4 \in \mathbb{Z}^2$ are lattice points in the relative interiors of four distinct facets of B. If $\bar{B}$, $\bar{f}$ and $\bar{w}$ are the values returned by Algorithm 11, then (i) the width of $\bar{B}$ along the vertical direction is at most 2, and (ii)*

$$|\bar{f}_2 + \bar{w}_2 - \bar{b}_2| \leq \frac{3}{2} \quad \forall \bar{b} \in \bar{B}.$$

Algorithm 10 Preprocessing step for triangles

function PREPROCESS($B, f, w, v^1, \ldots, v^3$)

Let $\tau^1(x) = \begin{bmatrix} v^2 - v^1 & v^3 - v^1 \end{bmatrix}^{-1} \left(x - v^1\right)$

Let $d \in \left\{\binom{1}{1}, \binom{1}{0}, \binom{0}{1}\right\}$ such that $\omega_d(\tau^1(B))$ is minimum.

Let $\tau^2(x) = \begin{cases} \tau^1(x) & \text{if } d = \binom{0}{1} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tau^1(x) & \text{if } d = \binom{1}{0} \\ \begin{bmatrix} 1 & 0 \\ -1 & -1 \end{bmatrix} \tau^1(x) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} & \text{if } d = \binom{1}{1} \end{cases}$

Let $\bar{B} = \{\tau^2(x) : x \in B\}$, $\bar{f} \leftarrow \tau^2(f)$, $w' \leftarrow \tau^2(w)$

Let $\varepsilon \leftarrow \frac{1}{2}\left(\max\{b_2 : (b_1, b_2) \in B\} + \min\{b_2 : (b_1, b_2) \in B\}\right)$

Let $\bar{w}_1 \leftarrow w'_1$ and $\bar{w}_2 \leftarrow w'_2 + \left\lfloor \varepsilon + \frac{1}{2} - \bar{f}_2 - w'_2 \right\rfloor$

Return $\bar{B}, \bar{f}, \bar{w}$

Algorithm 11 Preprocessing step for quadrilaterals

function PREPROCESS($B, f, w, v^1, \ldots, v^4$)

Let $\tau^1(x) = \begin{bmatrix} v^2 - v^1 & v^3 - v^1 \end{bmatrix}^{-1} \left(x - v^1\right)$

Let $\bar{v}^i \leftarrow \tau^1(v^i)$ for $i = \{1, \ldots, 4\}$

Let $\tau^2(x) = \begin{cases} \tau^1(x) & \text{if } \bar{v}^4 = \binom{1}{1} \\ \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \tau^1(x) & \text{if } \bar{v}^4 = \binom{1}{-1} \\ \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tau^1(x) & \text{if } \bar{v}^4 = \binom{-1}{1} \end{cases}$

Let $d \in \left\{\binom{1}{1}, \binom{1}{0}, \binom{0}{1}\right\}$ such that $\omega_d(\tau^1(B))$ is minimum.

Let $\tau^3(x) = \begin{cases} \tau^2(x) & \text{if } d = \binom{0}{1} \\ \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tau^2(x) & \text{if } d = \binom{1}{0} \end{cases}$

Let $\bar{B} = \{\tau^2(x) : x \in B\}$, $\bar{f} \leftarrow \tau^2(f)$, $w' \leftarrow \tau^2(w)$

Let $\varepsilon \leftarrow \frac{1}{2}\left(\max\{b_2 : (b_1, b_2) \in B\} + \min\{b_2 : (b_1, b_2) \in B\}\right)$

Let $\bar{w}_1 \leftarrow w'_1$ and $\bar{w}_2 \leftarrow w'_2 + \left\lfloor \varepsilon + \frac{1}{2} - \bar{f}_2 - w'_2 \right\rfloor$

Return $\bar{B}, \bar{f}, \bar{w}$

Proof Similarly to the proof of Proposition 8, it is clear that τ^1 is an affine unimodular function that maps v^1, v^2, v^3 to the points $\binom{0}{0}, \binom{1}{0}, \binom{0}{1}$, respectively. Since $\bar{v}^4 \in \mathbb{Z}^2$, since the area of the quadrilateral defined by $\bar{v}^1, \ldots, \bar{v}^4$ is one, and since no $\bar{v}^i$ is a convex combination of the others, then $\bar{v}^4$ can only be either $\binom{1}{1}, \binom{-1}{1}$ or $\binom{1}{-1}$. The transformation τ^2 maps $\bar{v}^1, \ldots, \bar{v}^4$ to $\binom{0}{0}, \binom{0}{1}, \binom{1}{0}$ and $\binom{1}{1}$. Furthermore, $\tau^2(B)$ satisfies the conditions for item (ii) of Lemma 6, therefore there exists d such that $\omega_d(\tau^1(B)) \leq 2$. To finish, we proceed similarly to the proof of (ii) in Proposition 8, replacing λ by its upper bound 2. □

4 Complexity analysis

In this section we study the worst case complexity of Algorithm 3. First, in Sect. 4.1, we assume only that the convex lattice-free set $B \subseteq \mathbb{R}^2$ is bounded and full-dimensional (i.e. we do not assume maximality at this point). We obtain an upper bound on the number of iterations of the algorithm, which depends on the second covering minimum of B and the width of B along the vertical direction. We highlight that the algorithm does not rely on computing either of these quantities, they are just used to bound the worst-case runtime of the algorithm in terms of characteristics of the lattice-free set B. Next, in Sect. 4.2, we focus on the case where B is a maximal lattice-free triangle or quadrilateral. Assuming that B has been preprocessed, we prove that Algorithm 3 requires at most a small number of iterations to finish.

4.1 Convex lattice-free sets in general

Let $B \subseteq \mathbb{R}^2$ be a bounded and convex lattice-free set containing the point $f \in \mathbb{R}^2$ in its interior. We do not assume that B is maximal. In this section, for any $\gamma \in \mathbb{R}_+$, we denote by γB the set obtained by scaling B by a factor of γ, using f as the origin. That is,

$$\gamma B = \{\gamma(b - f) + f : b \in B\} = \left\{x \in \mathbb{R}^2 : \frac{x}{\gamma} + \frac{f(\gamma - 1)}{\gamma} \in B\right\}.$$

We also denote by $\psi_{\gamma B}$ the gauge function of the convex set $\gamma B - f$. The following lemma proves that, if the union of all integer translations of γB cover $\mathbb{R}^2$ then the value of the trivial lifting is at most γ.

Lemma 12 *If $\gamma B + \mathbb{Z}^2 = \mathbb{R}^2$ for some $\gamma \in \mathbb{R}_+$ then $\pi_B(w) \leq \gamma$ for all $w \in \mathbb{R}^2$.*

Proof Let $w \in \mathbb{R}^2$. Since $\gamma B + \mathbb{Z}^2 = \mathbb{R}^2$, there exist $b \in \gamma B$ and $k \in \mathbb{Z}^2$ such that $w + f = b + k$. This implies that $w - k + f$ belongs to γB and therefore $\frac{w-k}{\gamma} + f \in B$. By the definition of the gauge function, it is clear that $\psi_B(w - k) \leq \gamma$. Since $\pi_B(w) \leq \psi_B(w - k)$, for any integer vector k, we conclude that $\pi_B(w) \leq \gamma$. □

In the following, let μ be the smallest non-negative number such that $\mu B + \mathbb{Z}^2 = \mathbb{R}^2$. This number is also known as the *second covering minimum* (or *covering radius*) of B [18]. As we recall, in order to evaluate the function

$$\pi_B(w) = \min_{k \in \mathbb{Z}^2} \psi_B(w + k),$$

for a given $w \in \mathbb{R}^2$, we computed $\min_{k_1 \in \mathbb{Z}} \psi_B \begin{pmatrix} w_1 + k_1 \\ w_2 + \bar{k}_2 \end{pmatrix}$ for different values of $\bar{k}_2 \in \mathbb{Z}$. The next lemma shows that, if μB does not intersect the horizontal line at level $f_2 + w_2 + \bar{k}_2$, for a particular $\bar{k}_2 \in \mathbb{Z}$, then such $\bar{k}_2$ can be safely discarded.

Lemma 13 *Let $\bar{k}_2 \in \mathbb{Z}$. Suppose there does not exist $b \in \mu B$ such that $b_2 = f_2 + w_2 + \bar{k}_2$. Then*

$$\min_{k_1 \in \mathbb{Z}} \psi_B \binom{w_1+k_1}{w_2+\bar{k}_2} \geq \min_{\alpha \in \mathbb{R}} \psi_B \binom{\alpha}{w_2+\bar{k}_2} > \mu.$$

Proof Note that

$$\min_{k_1 \in \mathbb{Z}} \psi_B \binom{w_1+k_1}{w_2+\bar{k}_2} \geq \min_{k_1 \in \mathbb{R}} \psi_B \binom{w_1+k_1}{w_2+\bar{k}_2} = \min_{\alpha \in \mathbb{R}} \psi_B \binom{\alpha}{w_2+\bar{k}_2} = \mu \min_{\alpha \in \mathbb{R}} \psi_{\mu B} \binom{\alpha}{w_2+\bar{k}_2}$$

Since we have $f + \binom{\alpha}{w_2+\bar{k}_2} \notin \mu B$ for every $\alpha \in \mathbb{R}$, then $\psi_{\mu B} \binom{\alpha}{w_2+\bar{k}_2} > 1$ for every $\alpha \in \mathbb{R}$, and the result follows. □

A consequence of Lemma 13 is that, if μB is not very wide in the vertical direction, then we only need to consider few values of $\bar{k}_2$ when computing the trivial lifting. This motivates the choice of τ in Sect. 3. The next theorem shows that Algorithm 3 does not spend time considering such useless $\bar{k}_2$.

Theorem 14 *Let $B \subseteq \mathbb{R}^2$ be a lattice-free set with $f \in \mathbb{R}^2$ in its interior, with second covering minimum $\mu \in \mathbb{R}_+$. If $w \in \mathbb{R}^2$ is such that*

$$|f_2 + w_2 - b_2| \leq \sigma \quad \forall b \in \mu B, \tag{9}$$

for some $\sigma \geq 1$, then Algorithm 3 stops after at most $\lfloor \sigma \rfloor$ iterations of the main loop upon receiving B, f and w as input.

Proof If there exists $b \in \mu B$ satisfying (9) with equality, we can always increase σ very slightly to obtain

$$|f_2 + w_2 - b_2| < \sigma \quad \forall b \in \mu B,$$

while keeping $\lfloor \sigma \rfloor$ unchanged. We may assume, therefore, that every $b \in \mu B$ satisfies (9) strictly. If the algorithm stops before $\lfloor \sigma \rfloor$ iterations, we are done. Suppose, then, that it runs for at least $\lfloor \sigma \rfloor$ iterations. First, we prove that, at the end of iteration $\lfloor \sigma \rfloor$, we have $\eta^* = \pi_B(w) \leq \mu$. Let $\Sigma = \{- \lfloor \sigma \rfloor, \ldots, \lfloor \sigma \rfloor\}$. Clearly, at the end of iteration $\lfloor \sigma \rfloor$, the variable η^* has value $\min_{k_2 \in \Sigma} h(k_2)$. By definition, we also have

$$\pi_B(w) = \min_{k_2 \in \mathbb{Z}} h(k_2) = \min \left\{ \min_{k_2 \in \Sigma} h(k_2), \min_{k_2 \in \mathbb{Z} \setminus \Sigma} h(k_2) \right\} = \min \left\{ \eta^*, \min_{k_2 \in \mathbb{Z} \setminus \Sigma} h(k_2) \right\}.$$

By Lemma 12, $\pi_B(w) \leq \mu$. By Lemma 13, $\min_{k_2 \in \mathbb{Z} \setminus \Sigma} h(k_2) > \mu$. Therefore, $\eta^* = \pi_B(w) \leq \mu$. Now we prove that the algorithm stops at the end of iteration $\lfloor \sigma \rfloor$. First, we prove that $\bar{k}_2 > |w_2|$. At the end of iteration $\lfloor \sigma \rfloor$, the value of $\bar{k}_2$ is $\lfloor \sigma \rfloor + 1$. By assumption, $\sigma > |f_2 + w_2 - b_2|$ for every $b \in \mu B$. Since $f \in \mu B$, we have $\sigma > |w_2|$. Therefore, $\bar{k}_2 = \lfloor \sigma \rfloor + 1 \geq \sigma > |w_2|$. Now we prove that $w_2 + \bar{k}_2 > \frac{\eta^*}{\zeta^+}$. Since $\bar{k}_2$ has value $\lfloor \sigma \rfloor + 1$, there does not exist $b \in \mu B$ such that $b_2 - f_2 - w_2 = \bar{k}_2$. By Lemma 13, and since $\bar{k}_2 > |w_2|$, as proved earlier, we have

$$\begin{aligned}
&\min_{\alpha\in\mathbb{R}} \psi_B \begin{pmatrix} \alpha \\ w_2+\bar{k}_2 \end{pmatrix} > \mu \geq \eta^* \\
&\quad\Rightarrow (w_2+\bar{k}_2) \min_{\alpha\in\mathbb{R}} \psi_B \begin{pmatrix} \alpha \\ 1 \end{pmatrix} > \eta^* \\
&\quad\Rightarrow (w_2+\bar{k}_2)\zeta^+ > \eta^* \\
&\quad\Rightarrow w_2+\bar{k}_2 > \frac{\eta^*}{\zeta^*}
\end{aligned}$$

We can similarly prove that $\bar{k}_2 - w_2 > \frac{\eta^*}{\zeta^-}$. Therefore, at the end of iteration $\lfloor\sigma\rfloor$, the loop condition is satisfied, and the algorithm stops. □

4.2 Maximal lattice-free sets

Now we suppose that $B \subseteq \mathbb{R}^2$ is a bounded maximal lattice-free set containing f in its interior. In this case, B is either a maximal lattice-free triangle or a maximal lattice-free quadrilateral. Maximal lattice-free sets are interesting in practice, since these are the sets that generate the strongest valid inequalities for (2). Aside from triangles and quadrilaterals, maximal lattice-free sets include splits, but the latter can be lifted easily. We will show that Algorithm 3 requires at most one or two iterations of the main loop to finish, depending on whether B is a type-3 triangle or not.

It is well known that the second covering minimum of any lattice-free set must be at least one. The following lemma shows the second covering minimum of B is also bounded above by a constant.

Lemma 15 *Let $B \subseteq \mathbb{R}^2$ is a full-dimensional maximal lattice-free set. If B is a type-1 triangle, a type-2 triangle or a quadrilateral, then $\mu \leq 1$. If B is a type-3 triangle, then $\mu \leq 2$.*

Proof For any set $B \subseteq \mathbb{R}^n$, we know that $B+\mathbb{Z}^n = \mathbb{R}^n$ if and only if $\tau(B)+\mathbb{Z}^n = \mathbb{R}^n$, where τ is any unimodular affine transformation. Suppose B is a maximal lattice-free quadrilateral. By applying an unimodular affine transformation, we may assume that the points $\binom{0}{0}$, $\binom{0}{1}$, $\binom{1}{0}$ and $\binom{1}{1}$ are in the relative interiors of four distinct facets of B. Therefore, B contains the unit square, and clearly $B + \mathbb{Z}^2 = \mathbb{R}^2$. The same argument applies for type-1 and type-2 triangles. See Fig. 2 for an illustration. Now suppose B is a type-3 triangle. We may assume that the points $\binom{0}{0}$, $\binom{0}{1}$ and $\binom{1}{0}$ are

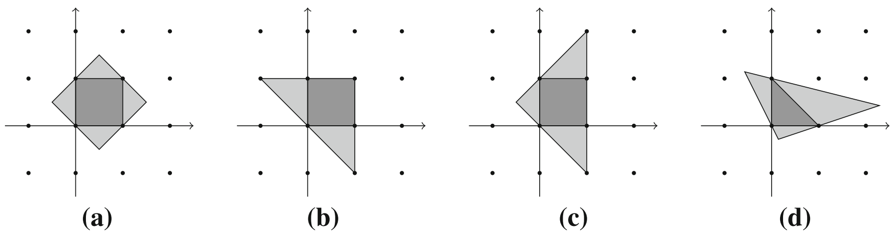

Fig. 2 Proof of Lemma 15. **a** Quadrilateral, **b** type-1 triangle, **c** type-2 triangle, **d** type-3 triangle

in the relative interiors of three distinct facets of B. The set $2B$, therefore, contains a type-1 triangle as a subset. By the previous case, $2B + \mathbb{Z}^2$ covers $\mathbb{R}^2$.

Now we proceed to obtain upper bounds on the number of iterations of Algorithm 3 for maximal lattice-free sets, by applying Theorem 14. We consider two distinct cases, depending on whether B is a type-3 triangle or not. In every case, we assume that B, f and w have been preprocessed either by Algorithm 10 or by Algorithm 11. First, suppose that B is either a type-1 triangle, or a type-2 triangle, or a quadrilateral. The next theorem shows that the algorithm stops after a single iteration of the main loop.

Theorem 16 *Let $B \subseteq \mathbb{R}^2$ be a maximal lattice-free quadrilateral or triangle of types 1 or 2 containing $f \in \mathbb{R}^2$ in its interior and having vertical width at most 2. If $w \in \mathbb{R}^2$ is such that*

$$|f_2 + w_2 - b_2| \leq \frac{3}{2} \quad \forall b \in B,$$

then, upon receiving B, f and w as input, Algorithm 3 stops after at most a single iteration of the main loop.

Proof Let μ be the second covering minimum of B. By Lemma 15, $\mu \leq 1$. Since every lattice-free set has second covering minimum at least one, then $\mu = 1$. Therefore,

$$|f_2 + w_2 - b_2| \leq \frac{3}{2} \quad \forall b \in \mu B.$$

By Theorem 14, we conclude that Algorithm 3 requires at most $\left\lfloor \frac{3}{2} \right\rfloor = 1$ iteration to finish. □

This case is closely related to the closed formula presented by Basu et al. [5], which can compute trivial lifting coefficients under the assumption that B is a type-1 or type-2 maximal lattice-free triangle. Although the formula itself is closed, it assumes that B is already in some standard form, thus requires the application of a pre-processing step. The formula works by evaluating the function ψ_B at six points, in the worst case, and taking the minimum.

Theorem 16 proves that Algorithm 3 requires constant time to finish in the worst case. More interestingly, however, this theorem shows that the algorithm requires at most a single iteration of the main loop, implying that at most six calls to evaluate the function ψ_B are needed. This matches the number of calls made by the closed formula presented by Basu et al. Also note that Algorithm 3 requires the same number of calls when B is a maximal lattice-free quadrilateral, while the aforementioned closed formula does not apply for quadrilaterals.

Now we consider the case where B is a type-3 triangle. In this case, since both the maximum width along the vertical direction, as well as the second covering minimum, can be higher than before, the algorithm may require more iterations to terminate. In the worst case, however, it still requires at most a low, constant number of iterations.

Lemma 17 *Let $B \subseteq \mathbb{R}^2$ be a lattice-free set with $f \in \mathbb{R}^2$ in its interior. Also, let $w \in \mathbb{R}^2$ and $\gamma \in \mathbb{R}_+$ such that*

$$|f_2 + w_2 - b_2| \leq \gamma \quad \forall b \in B.$$

Then, for any $\mu \geq 1$,

$$|f_2 + w_2 - \bar{b}_2| \leq \gamma(2\mu - 1) \quad \forall \bar{b} \in \mu B.$$

Proof Let $\bar{b} \in \mu B$. By definition, there exists $b \in B$ such that $\bar{b} = \mu(b - f) + f$. Also, since $f \in B$ and $|f_2 + w_2 - b_2| \leq \gamma$ for every $b \in B$, we have $|w_2| \leq \gamma$. Therefore,

$$\begin{aligned} |f_2 + w_2 - \bar{b}_2| &= |f_2 + w_2 - \mu(b_2 - f_2) - f_2| \\ &= |\mu(f_2 + w_2 - b_2) - (\mu - 1)w_2| \\ &\leq \mu|f_2 + w_2 - b_2| + (\mu - 1)|w_2| \\ &\leq \mu\gamma + (\mu - 1)\gamma \\ &= \gamma(2\mu - 1) \end{aligned}$$

□

Theorem 18 *Let $B \subseteq \mathbb{R}^2$ be a type-3 triangle containing $f \in \mathbb{R}^2$ in its interior and having vertical width at most $1 + \frac{2}{3}\sqrt{3}$. If $w \in \mathbb{R}^2$ is such that*

$$|f_2 + w_2 - b_2| \leq 1 + \frac{1}{3}\sqrt{3} \quad \forall b \in B,$$

then, upon receiving B, f and w as input, Algorithm 3 stops after at most four iterations of the main loop.

Proof Let μ be the second covering minimum of B. By Lemma 17,

$$|f_2 + w_2 - b_2| \leq \left(1 + \frac{1}{3}\sqrt{3}\right)(2\mu - 1) \quad \forall b \in \mu B.$$

Since $\mu \leq 2$ by Lemma 15, we have

$$|f_2 + w_2 - b_2| \leq 3 + \sqrt{3} \quad \forall b \in \mu B.$$

Therefore, by Theorem 14, we conclude that Algorithm 3 requires at most $\left\lfloor 3 + \sqrt{3} \right\rfloor = 4$ iterations to finish. □

Corollary 19 summarize the results from Propositions 8 and 9 and Theorems 16 and 18.

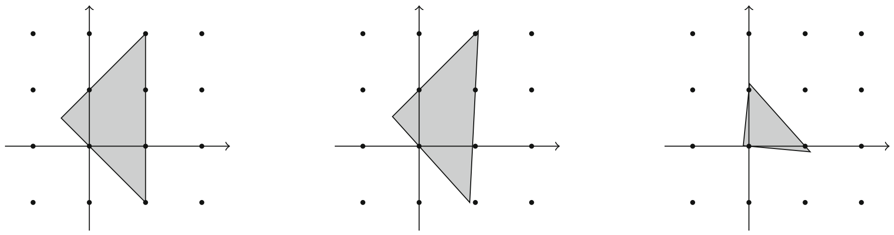

Fig. 3 Illustration of "good" and "bad" type 3 triangles. **a** Type-2 triangle, **b** "good" type-3 triangle, **c** "bad" type-3 triangle

Corollary 19 *Let $B \subseteq \mathbb{R}^2$ be any bounded maximal lattice-free set containing f in its interior. The lifting problem (5) can be solved by a combination of Algorithm 10 or 11 (depending on whether B is a triangle or a quadrilateral), and Algorithm 3. The total time necessary is the time taken by the linear algebra in Algorithm 10 or 11 and*

(i) a single iteration of Algorithm 3 if B is a quadrilateral or triangle of types 1 or 2, or
(ii) at most four iterations of Algorithm 3 if B is a type-3 triangle.

It is worth noting that the upper bound on the number of iterations obtained in Theorem 18 is a worst-case upper bound and that, the actual run time for some type 3 triangles can be smaller. For instance, the example shown in Fig. 3b is a type 3 triangle that is very close to being a type 2 triangle (Fig. 3a), so its trivial lifting coefficient and the runtime of the algorithm are likely similar to the type 2 case. On the other hand, the worst case runtime will be obtained by a type 3 triangle with high second covering minimum, like the one seen in Fig. 3c.

Another issue worth mentioning is that, computationally, it may be difficult to differentiate between the cases in Fig. 3a, b, due to numerical inaccuracy. This highlights a big advantage of the generic trivial lifting algorithm presented in this work, which computes the correct coefficient for any (even non-maximal) lattice-free set, as opposed to one that relies on the particular format of the maximal lattice-free set.

5 Computational experiments

In order to evaluate the practical efficiency of Algorithm 3, we implemented it and compared it against two variations of the naive method described in the introduction and against a black-box MIP solver given the formulation of Averkov and Basu [2].

5.1 Algorithms and variations

Two variations of Algorithm 3 were implemented and tested. The first variation (`bound`) applies the procedure directly to the input data, without performing any kind of preprocessing, while the second variation (`bound-pre`) applies the preprocessing step described in Sect. 3. Both variations were implemented in standard C, and do not make use of any external dependencies. The complete source code has been made available online [13]. In order to evaluate the function

$$\min_{\alpha_1 \in \mathbb{R}} \psi \begin{pmatrix} \alpha_1 \\ \bar{\alpha}_2 \end{pmatrix} \tag{10}$$

where $\bar{\alpha}_2$ is fixed, an ad-hoc method was used, instead of a generic LP solver. More precisely, if $B \subseteq \mathbb{R}^2$ is a polyhedron containing $f \in \mathbb{R}^2$ in its interior, then B can be written as

$$B = \{x \in \mathbb{R}^2 : d^i(x - f) \leq 1, i \in \{1, \ldots, t\}\}$$

where $d^1, \ldots, d^t \in \mathbb{R}^2$. Then (10) is equivalent to the linear program

$$\begin{array}{ll} \text{minimize} & \varepsilon \\ \text{subject to} & \varepsilon - d^i_1 \alpha_1 \geq d^i_2 \bar{\alpha}_2 \qquad i = 1, \ldots, t \\ & \varepsilon, \alpha_1 \text{ free} \end{array}$$

Since, in our experiments, B is always a maximal lattice-free set, this LP has at most a constant and very small number of bases. Instead of calling a generic LP solver, we simply enumerated all these bases, and found the one with best objective value.

We also implemented and tested two variations of the naive trivial lifting algorithm. The first variation (`naive-fixed`) simply evaluates the function $\psi(w + k)$ for all $k \in [-M, M]^2$, where M is a large constant which does not depend on any input data. During our tests, this value was fixed to 50. This variation is the simplest to implement, and probably the most widespread, but does not always produce the correct answers, since, for every fixed M, there is always a lattice-free set $B \subseteq \mathbb{R}^2$ such that M is not large enough for B. The second variation (`naive-bbox`) solves this problem by computing the bounding box for each lattice-free set B, and evaluating all $\psi(w + k)$ for all k such that $f + w + k$ is either inside or reasonably close to the bounding box. In order to avoid exceedingly long running times on some hard instances, the bounding box was intersected with a box $[-M, M]^2$ of fixed size, where M is a large constant which was fixed to 10^3 in our tests. Setting this constant to a smaller value reduces the maximum running time of `naive-bbox`, but increases the probability of generating incorrect answers.

We also modelled (5) as a mixed-integer linear program with a fixed number of variables, as described in [2], and we solved it using IBM ILOG CPLEX, version 12.6.0.0. We refer to this implementation as `mip`. Because we use a generic MIP solver (based on branch-and-bound and the simplex method), even in fixed dimension, the worst-case running time is exponential in the encoding size of the problem, while it could be polynomial in theory. This choice was made because, to the best of our knowledge, there is no currently-available MIP solver that is polynomial in fixed-dimension and competitive with CPLEX, for MIPs containing both continuous and integral variables.

Finally, since the preprocessing step described in Sect. 3 could also potentially improve the running times of `naive-bbound` and `mip`, we implemented two variations of these algorithms (`naive-bbound-pre` and `mip-pre`) where the preprocessing step is applied beforehand.

All computations, for all the variations described, were performed in floating point arithmetic, due to the observation that small arithmetical errors are not amplified by the algorithms, and therefore have no significant impact in the final cut coefficient. The code used to evaluate the function $\psi(w)$ for a certain $w \in \mathbb{R}^2$ was also exactly the same.

5.2 Instances

For our computational experiments, an instance of the trivial lifting problem consists of a convex lattice-free set $B \subseteq \mathbb{R}^2$, along with an interior point $f \in \mathbb{R}^2$, and a list of rays $w_1, \ldots, w_k$ that should be lifted. We use a list of rays, instead of a single ray, since, in practice, a cut generator usually needs to lift, for the same intersection cut, multiple rays corresponding to different integer variables. By receiving a list of rays in advance, the cut generator may perform a preprocessing step exactly once, before the trivial lifting computations begin, instead of one time for each ray. The decision of whether to apply some costly preprocessing step could also depend on the number of rays to be lifted, although we limited ourselves to a fixed choice here.

In our experiments, we used two lists of lattice-free sets. The first list was obtained by running the two-row intersection cut generator implemented by Louveaux and Poirrier [21] on the benchmark set of the MIPLIB 2010 [19] and capturing, for each intersection cut generated, the associated lattice-free set. This list was then filtered to exclude splits, since these can be lifted easily, and non-maximal lattice-free sets, since these can be transformed into maximal lattice-free sets. Finally, a sample of 1000 lattice-free sets from this list, picked randomly, was considered for our experiments. The second list of lattice-free sets was obtained by applying a shear transformation to each set on the first list. The precise transformation was given by

$$f(x) = \begin{bmatrix} 51 & 5 \\ 10 & 1 \end{bmatrix} x.$$

This transformed list has sets that resemble the set from Fig. 1a, and can be seen as a pathological scenario for trivial lifting algorithms.

For each lattice-free set on each list, we also randomly generated a fixed number of rays w_i, uniformly distributed inside the box $[0, 1)^2$. The lists of rays were generated randomly, since we observed that the choice of rays to be lifted had negligible impact in the performance of the algorithms considered. Most of the impact came, instead, from the choice of lattice-free sets. This also allowed us to evaluate the impact of lifting a different number of rays for each lattice-free set.

We conducted our experiments on a machine with Intel Core i5-3210M (2.50 GHz, dual core) processor and 16 GB of memory. In order to measure the running times more accurately, each instance was solved multiple times and the CPU running times were averaged. The precise number of samples varied according to algorithm being tested: 1000 samples for faster algorithms (`bound`) and either 10 or 100 samples for slower algorithms (`naive` and `mip`). CPLEX was configured to run in single-threaded mode, since this was the configuration that presented the best performance, and to use numerical emphasis.

5.3 Results and discussion

First, we focus on the unmodified list of lattice-free sets obtained from the cut-generator. Table 1 summarizes the CPU running times that each of the four algorithms took to process this list of lattice-free sets, with 100 randomly generated rays per set. The running time to process one lattice-free set includes any time spent on preprocessing the set, plus the time spent computing the lifted coefficients for all the rays. For each algorithm, the table shows the average, the median and the maximum running times in milliseconds to process each set. The table also shows the percentage of sets for which at least one cut coefficient was calculated incorrectly.

As Table 1 shows, algorithm `bound-pre` presented the fastest average running time among all variations for this set of instances, taking only 0.066 ms on average to process each set. This was more than 11, 180, 500 and 1800 times faster on average than `bound`, `naive-bbox`, `naive-fixed` and `mip`, respectively. It performed consistently well on all instances, having a maximum running time of only 0.164 ms. Algorithms `bound`, `naive-bbox` and `mip`, on the other hand, presented significant slowdown for some instances, having maximum running times of 43, 2372 ms and 3315 ms respectively. Applying the pre-processing step to `naive-bbox` and `mip` reduced the maximum running times of these algorithms to 98 ms and 161 ms, which was still 600 and 980 times slower than `bound-pre`. Although algorithm `naive-fixed` was consistent and had a better maximum running time than `naive-bbox`, we note that it failed to compute some coefficients correctly. Because of our restriction on the maximum size of the bounding boxes, algorithm `naive-bbox` also produced some incorrect coefficients.

Table 1 also shows that, for each instance, the best running time was obtained either by `bound-pre` or `bound`, and never by the other algorithms. Algorithms `bound-pre` and `bound` presented average ratio-to-best of 1.052 and 11.518 respectively. Although `bound` was faster than `bound-pre` for some instances, it was only very slightly so. For instances where `bound-pre` was faster than `bound`, however, the difference in running times was much more significant.

Now we consider the second list of lattice-free sets, obtained by applying a shear transformation to the sets of the first list, with 100 randomly generated rays per set. Table 2 summarizes the running times for all algorithms. The transformation had virtually no impact on the performance of algorithm `bound-pre`. Its average, median and maximum running times remained almost unchanged. The median running times for algorithms `bound`, `naive-bbox` and `mip`, on the other hand, became on average 6, 10 and 2 times slower than before, respectively. For some instances, `naive-bbox` and `mip` required minutes of processing time, while `bound-pre` required only a fraction of millisecond. The table also shows that failure rate of algorithm `naive-fixed` increased considerably, to 10%. Algorithm `mip` also calculated some coefficients

Table 1 Running times statistics: original lattice-free sets, 100 rays per set

	`bound-pre`	`bound`	`naive-bbox-pre`	`naive-bbox`	`naive-fixed`	`mip-pre`	`mip`
Mean (ms)	0.066	0.750	6.098	12.011	32.964	102.687	124.124
Minimum (ms)	0.048	0.044	2.360	2.360	29.880	75.200	80.400
Median (ms)	0.064	0.084	2.840	3.080	32.560	98.400	109.200
Maximum (ms)	0.164	43.756	98.480	2372.320	42.400	161.600	3315.200
Failure rate (%)	0.0	0.0	0.0	0.1	0.3	0.0	0.0
Best (%)	77.6	34.0	0.0	0.0	0.0	0.0	0.0
Avg ratio to best	1.052	11.518	97.095	191.156	535.318	1668.691	2016.075

Table 2 Running times statistics: transformed lattice-free sets, 100 rays per set

	`bound-pre`	`bound`	`naive-bbox-pre`	`naive-bbox`	`naive-fixed`	`mip-pre`	`mip`
Mean (ms)	0.066	6.150	6.341	460.164	32.962	112.370	1335.648
Minimum (ms)	0.048	0.112	2.440	5.600	29.800	58.000	144.000
Median (ms)	0.060	0.576	2.960	30.400	32.560	108.000	232.000
Maximum (ms)	0.120	641.000	101.200	9281.200	42.000	224.000	107,756.000
Failure rate (%)	0.0	0.0	0.0	0.5	10.0	0.0	2.1
Best (%)	100.0	0.0	0.0	0.0	0.0	0.0	0.0
Avg ratio to best	1.000	89.831	96.764	6927.182	516.400	1764.603	19,667.757

incorrectly, due to insufficient numerical precision. Algorithm `bound-pre` presented the best performance for every instance in this set.

As a final experiment, we implemented the heuristic lifting algorithm proposed by Espinoza [12]. Since the algorithm has little variance on the runtimes, we do not include it in the tables. The runtimes of the heuristic were on average 0.003 ms, which is about 20 times faster than the average runtime of our best algorithm `bound-pre`. However, when it comes to computing the correct lifting coefficient, we note that for all test instances at least one coefficient computed was incorrect, which means that in practice all cuts computed using this lifting are strictly dominated by cuts computed using the `bound-pre` lifting. On average, around 60% of the coefficients computed were incorrect with at least half of those being more than 1.15 units away from the correct answer, which can often be at most 1, thus resulting in a significant error. We thus conclude that the loss in speed to an average of 0.066 ms per lifted coefficient will, in most cases, be worth the effort if compared to the heuristic approach.

6 Conclusion

In this work we considered an efficient method for computing the so-called trivial lifting coefficients in two dimensions. Even though this problem may be considered somewhat well understood in theoretical terms, an efficient implementation for it is still challenging, particularly if one considers that this is a problem that may need to be solved thousands of times for every cut that is added to an MIP.

We presented an algorithm that exploits the particular structure of the lifting function and that uses ideas from algorithms for integer programs in fixed-dimensions [20]. By a careful analysis, we show that the resulting algorithm has a very small upper bound on the number of iterations, where the precise value of that bound can be expressed in terms of the second covering minimum of the lattice-free set considered.

In practice, the algorithm performs at least two orders of magnitude faster than other methods for computing the trivial lifting function, while still guaranteeing that the correct trivial lifting coefficient is computed. Such speedup can mean the difference between the trivial lifting being used or not in practice.

We end by noting that essentially similar ideas can also be applied to the computation of the trivial lifting in higher dimensions. However, for those cases, we do not yet have such an efficient implementation nor a significant and careful analysis of worst-case running time.

A Source code

The source code for the trivial lifting algorithm presented in Sect. 2 has been made available online [13] under the GNU General Public License, version 3. The source code is written in ANSI C, having only IBM ILOG CPLEX as its external dependency (required only for benchmarking `mip` and `mip-pre` methods). The structure of the project is presented in Table 3. Complete instructions to compile the source code, run

Table 3 Directory structure and main files

Name	Description
`build/`	Directory that holds the compiled files
`cmake/`	Custom scripts used by CMake
`gtest/`	Google's C++ Test Framework
`run/`	Benchmark scripts and instances
`src/`	Source code for the lifting algorithm
`tests/`	Unit tests
`README.md`	Instructions for compiling, running and producing the tables

the computational experiments and generate the tables have been included in the file `README.md`.

Let $S \subseteq \mathbb{R}^2$ be a lattice-free set containing $f \in \mathbb{R}^2 \setminus \mathbb{Z}^2$ in its interior. In our code, we make the assumption that S is a bounded set, and therefore can be represented either as the convex combination of vertices $v^1, \ldots, v^k \in \mathbb{R}^2$, or as the as the intersection of half-spaces $\{x \in \mathbb{R}^2 : h^i(x - f) \leq 1, \text{for } i = 1, \ldots, q\}$. For maximal lattice-free sets, our code also assumes that a list of lattice points in the boundary of the set is given. This data is stored in the structure `LFreeSet2D`. The halfspace representation is needed for evaluating the gauge function and performing the trivial lifting. If this representation is not readily available, the source code includes the auxiliary function `LFREE_2D_compute_halfspaces`, which computes it, given the interior point f and the list of vertices. The source code also includes the function `LFREE_2D_preprocess`, which implements the preprocessing step described in Sect. 3. This function receives an `LFreeSet2D`, modifies it in-place and returns the transformation matrix. After the set is preprocessed, the trivial lifting function can be efficiently computed by calling `LIFTING_2D_bound`. A sample usage is presented below. For brevity, no preprocessing is performed.

```
int n_halfspaces = 5;
double halfspaces[] = {
    -1/2.0, 0,
    -1/2.0, -1/2.0,
    -1/4.0, 1/2.0,
    -1/6.0, -5/6.0,
    5/16.0, 1/8.0
};
double r[] = { 1/2.0, 1/2.0 }; // ray to be lifted
double value; // value of the trivial lifting
LIFTING_2D_bound(n_halfspaces, halfspaces, r, &value);
```

Finally, for benchmarking purposes, the source code also includes the functions `LIFTING_2D_naive` and `LIFTING_2D_mip`, which implement the naive and the MIP methods, respectively. Other auxiliary functions include `LFREE_2D_get_bounding_box`, which computes the bounding box for a given bounded lattice-free set, and `LIFTING_2D_psi`, which simply evaluates the gauge function. More complete examples, showing the usages of all these functions, can be found in the unit tests, which are included in the source code package.

References

1. Andersen, K., Louveaux, Q., Weismantel, R., Wolsey, L.: Inequalities from two rows of a simplex tableau. In: Fischetti, M., Williamson, D. (eds.) Integer Programming and Combinatorial Optimization, volume 4513 of Lecture Notes in Computer Science, pp. 1–15. Springer, Berlin (2007)
2. Averkov, G., Basu, A.: Lifting properties of maximal lattice-free polyhedra. Math. Program. **154**(1–2), 81–111 (2015)
3. Balas, E.: Intersection cuts—a new type of cutting planes for integer programming. Oper. Res. **1**(19), 19–39 (1971)
4. Balas, E., Jeroslow, R.G.: Strengthening cuts for mixed integer programs. Eur. J. Oper. Res. **4**(4), 224–234 (1980)
5. Basu, A., Bonami, P., Cornuéjols, G., Margot, F.: Experiments with two-row cuts from degenerate tableaux. INFORMS J. Comput. **23**, 578–590 (2011)
6. Basu, A., Campêlo, M., Conforti, M., Cornuéjols, G., Zambelli, G.: Unique lifting of integer variables in minimal inequalities. Math. Program. **141**(1–2), 561–576 (2013)
7. Borozan, V., Cornuéjols, G.: Minimal valid inequalities for integer constraints. Math. Oper. Res. **34**(3), 538–546 (2009)
8. Conforti, M., Cornuéjols, G., Zambelli, G.: Integer Programming, vol. 271. Springer, Berlin (2014)
9. Dey, S.S., Lodi, A., Tramontani, A., Wolsey, L.A.: On the practical strength of two-row tableau cuts. INFORMS J. Comput. **26**(2), 222–237 (2014)
10. Dey, S.S., Louveaux, Q.: Split rank of triangle and quadrilateral inequalities. Math. Oper. Res. **36**(3), 432–461 (2011)
11. Dey, S.S., Wolsey, L.A.: Two row mixed-integer cuts via lifting. Math. Program. **124**, 143–174 (2010)
12. Espinoza, D.G.: Computing with multi-row Gomory cuts. Oper. Res. Lett. **38**(2), 115–120 (2010)
13. Fukasawa, R., Poirrier, L., Xavier, Á.S.: The (not so) trivial lifting in two dimensions: source code, Aug 2018. https://doi.org/10.5281/zenodo.1342770
14. Gomory, R.E.: Some polyhedra related to combinatorial problems. Linear Algebra Appl. **2**(4), 451–558 (1969)
15. Gomory, R.E., Johnson, E.L.: Some continuous functions related to corner polyhedra, part I. Math. Program. **3**, 23–85 (1972)
16. Gu, Z., Nemhauser, G.L., Savelsbergh, M.W.P.: Sequence independent lifting in mixed integer programming. J. Comb. Optim. **4**, 109–129 (2000)
17. Hurkens, C.A.J.: Blowing up convex sets in the plane. Linear Algebra Appl. **134**, 121–128 (1990)
18. Kannan, R., Lovász, L.: Covering Minima and Lattice Point Free Convex Bodies, pp. 193–213. Springer, Berlin (1986)
19. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**(2), 103–163 (2011)
20. Lenstra Jr., H.W.: Integer programming with a fixed number of variables. Math. Oper. Res. **8**(4), 538–548 (1983)
21. Louveaux, Q., Poirrier, L.: An algorithm for the separation of two-row cuts. Math. Program. **143**(1–2), 111–146 (2014)
22. Louveaux, Q., Poirrier, L., Salvagnin, D.: The strength of multi-row models. Math. Program. Comput. **7**(2), 113–148 (2015)
23. Oertel, T., Wagner, C., Weismantel, R.: Convex integer minimization in fixed dimension. http://arxiv.org/pdf/1203.4175v1.pdf (2012). Accessed 9 Aug 2018
24. Padberg, M.W.: On the facial structure of set packing polyhedra. Math. Program. **5**(1), 199–215 (1973)