McMaster University

Advanced Optimization Laboratory

Title:

The complexity of geometric scaling

Authors:

Antoine Deza, Sebastian Pokutta, and Lionel Pournin

AdvOL-Report No. 2022/1

May 2022, Hamilton, Ontario, Canada

THE COMPLEXITY OF GEOMETRIC SCALING

Antoine Deza

McMaster University, Hamilton, Ontario, Canada
e-mail: deza@mcmaster.ca

Sebastian Pokutta

Zuse Institute Berlin, Germany
e-mail: pokutta@zib.de

Lionel Pournin

Université Paris 13, Villetaneuse, France
e-mail: lionel.pournin@univ-paris13.fr

ABSTRACT

Geometric scaling, introduced by Schulz and Weismantel in 2002, solves the integer optimization problem $\max\{c{\cdot}x : x \in P \cap \mathbb{Z}^n\}$ by means of primal augmentations, where $P \subset \mathbb{R}^n$ is a polytope. We restrict ourselves to the important case when P is a 0/1 polytope. Schulz and Weismantel showed that no more than $O(n \log n \|c\|_\infty)$ calls to an augmentation oracle are required. This upper bound can be improved to $O(n \log \|c\|_\infty)$ using the early-stopping policy proposed in 2018 by Le Bodic, Pavelka, Pfetsch, and Pokutta. Considering both the maximum ratio augmentation variant of the method as well as its approximate version, we show that these upper bounds are essentially tight by maximizing over a n-dimensional simplex with vectors c such that $\|c\|_\infty$ is either n or 2^n.

1. Introduction

The computational performance of linear optimization algorithms is closely related to the geometric properties of the feasible region. The combinatorial properties can also play an important role, in particular for integer optimization algorithms. Starting with the Klee–Minty cubes [7] exhibiting an exponential number of simplex pivots, worst-case constructions have helped providing a deeper understanding of how the structural properties of the input affect the performance of linear optimization. Recent examples include the construction of Allamigeon et al. [1, 2] for which the primal-dual log-barrier interior point method performs an exponential number of iterations, and thus is not strongly polynomial. In a similar spirit, a lower bound on the number of simplex pivots required in the worst case to perform linear optimization on a lattice polytope has been recently established in [4, 5]. In turn, a preprocessing and scaling algorithm has been proposed by Del Pia and Michini in [8] to construct simplex paths that are short relative to these lower bounds. We focus on geometric scaling methods, introduced in [10], for integer optimization on 0/1 polytopes. For these methods, no worst-case instances have been proposed to the best of our knowledge. In contrast, a tight lower bound has been provided by Le Bodic et al. [3] for bit scaling methods [11].

A 0/1 polytope is the convex hull of a subset of the vertex set of the unit n-dimensional hypercube $[0,1]^n$. Given a 0/1 polytope $P \subset \mathbb{R}^n$ and a vector c in $\mathbb{Z}^n$, we are interested in the following optimization problem:

$$\max\{c\cdot x : x \in v(P)\},$$

where $v(P)$ denotes the vertex set of P.

We recall geometric scaling in Section 2 and refer the reader to [3, 9, 10] for comprehensive expositions. In Section 3, we show that the maximum-ratio augmentation variant of this method sometimes requires $n+\log n\|c\|_\infty+1$ steps and in Section 4 that any implementation can require $n/3+\log n\|c\|_\infty+1$ steps. In Section 5, we improve this number of steps to $n+\log n\|c\|_\infty+1$ by showing that the halving ratio is the reason for the gap between the two lower bounds obtained in Sections 3 and 4. This result highlights the tradeoff between the chosen amount of scaling and the accuracy of the feasibility oracle used in the implementation. Open questions are discussed in Section 6.

2. Geometric scaling

There are several variants of geometric scaling. In Section 3, we will focus on Algorithm 1, a practical implementation of Algorithm 3 from [3] based on the description from Sections 5.4 and 5.5 in the same article. It is important to keep in mind that this implementation is by no means canonical. For example, Line 3 could be replaced by the computation of a vertex x of P such that

$$c\cdot(x - \tilde{x}) > \mu\|x - \tilde{x}\|_1$$

and the condition in Line 4 by checking whether such a vertex exists: if there is such a vertex, then x gets assigned to $\tilde{x}$ and otherwise, μ is halved (see Algorithm 3 in [3]). This results in Algorithm 2, that we study in Section 4. In any case, the steps that end at Line 5 will be referred to as *halving steps* and the steps that end at Line 7 as *augmenting steps*. A series of consecutive augmentation steps performed with same the value of μ is referred to as a *scaling phase*, and a series of consecutive halving steps as an *halving phase*.

Algorithm 1: MRA-based geometric scaling

Input: a 0/1 polytope P contained in $\mathbb{R}^n$,
a vector c in $\mathbb{Z}^n$,
a vertex x^0 of P, and
a number μ_0 greater than $\|c\|_\infty$.

Output: A vertex $x^\star$ of P that maximizes $c\cdot x$.

1 $\mu \leftarrow \mu_0$, $\tilde{x} \leftarrow x^0$
2 **repeat**
3   **compute** a vertex x of P that maximizes $c\cdot(x - \tilde{x})/\|x - \tilde{x}\|_1$
4   **if** $x = \tilde{x}$ *or* $c\cdot(x - \tilde{x}) < \mu\|x - \tilde{x}\|_1$ **then**
5         $\mu \leftarrow \mu/2$ (halving step)
6   **else**
7         $\tilde{x} \leftarrow x$ (augmenting step)
8   **end**
9 **until** $\mu < 1/n$;
10 Return $\tilde{x}$

The following remarks about geometric scaling hold for both the variants described in Algorithm 1 and Algorithm 2; for details we refer the interested reader to [3]. In particular, the combination of these two remarks provides a slightly differentiated picture on the complexity we study here.

Remark 2.1 (Solutions are strictly monotonically increasing with respect to c): The sequence of points $x_1, x_2, \ldots$ generated by geometric scaling satisfies

$$c \cdot x_1 < c \cdot x_2 < \ldots$$

Note that this is very different from bit scaling, another augmentation-based optimization approach for 0/1 polytopes introduced in [11], where points can be revisited in successive scaling phases and the sequence of generated points is not strictly increasing with respect to the original objective c. This fact also impacts the structure of our lower bounds: for bit scaling it was shown in [3] that the number of required augmenting steps can depend on $\log \|c\|_\infty$ by making bit scaling revisit points. It will not be possible to do the same here and, in contrast to the bounds obtained for bit scaling, we will only be able to show that the total number of steps (the sum of the number of augmenting steps and the number of halving steps) depends on $\log \|c\|_\infty$. Our bounds for the number of required augmenting steps do not exceed n.

Remark 2.2 (μ provides an upper bound on the primal gap): Consider the value of μ taken before a halving update. Either $\mu = \mu_0$ and then by definition this is a lower bound or μ arose from a previous halving step. In that halving iteration, before the actual halving, we had for some iterate $\tilde{x}$:

$$\max_{y \in v(P)} c\cdot(y - \tilde{x}) \leq \mu \|y - \tilde{x}\|_1 \leq \mu n.$$

The worst-case complexity in the number of total steps for geometric scaling on 0/1 polytopes is $O(n \log n \|c\|_\infty)$. The above two remarks allow to improve the worst-case complexity of geometric scaling slightly in the case of 0/1 polytopes as shown in [3]. Observe that geometric scaling requires $O(n \log \|c\|_\infty)$ iterations until $\mu \leq 1/2$. According to Remark 2.2, we know that the primal gap at that point in time is at most

$$\max_{y \in v(P)} c\cdot(y - \tilde{x}) \leq \mu \|y - \tilde{x}\|_1 \leq 2\mu n \leq n$$

and by Remark 2.1, we know that we improve by at least 1 in each iteration, so that the total number of iterations can be bounded as

$$O(n \log \|c\|_\infty) + n = O(n \log \|c\|_\infty)$$

iterations; we assume here that one would simply stop the algorithm after (at most) n additional steps and does not continue performing unnecessary halving steps as we are guaranteed to be optimal. In the following, we will refer to these improved bounds as *early stopping* bounds. With this we obtain the following upper bounds that we compare against.

Proposition 2.3 ([3]): *Given a* 0/1 *polytope and an objective function* c*, geometric scaling (either variant) solves*

$$\max_{x \in v(P)} c \cdot x$$

in no more than $O(n \log n \|c\|_\infty)$ *iterations using Algorithm 1 and Algorithm 2 and no more than* $O(n \log \|c\|_\infty)$ *iterations via early stopping.*

Note that in the following we state the lower bounds for the exact forms of Algorithm 1 and Algorithm 2. In light of the above discussion and Proposition 2.3, using the early stopping variants reduces the number of required halving steps, and thus the lower bounds, by the n term under log.

3. Worst-case instances for geometric scaling via MRA

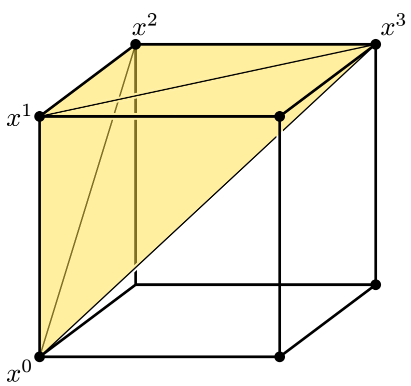

Figure 1. The simplex S when $n = 3$.

For any integer i such that $0 \le i \le n$, denote by x^i the point in $\mathbb{R}^n$ whose last i coordinates are equal to 1 and whose other coordinates are equal to 0. Note that x^0 is the origin of $\mathbb{R}^n$. This point will be our initial vertex for Algorithm 1, and is therefore the same x^0 as in the input of that algorithm. Consider the n-dimensional simplex S illustrated in Fig. 1 when $n = 3$, whose vertices are the points x^0 to x^n. Further consider the vector c whose ith coordinate is i:

$$c = (1, 2, \ldots, n).$$

In the remainder of the section, S and c are fixed as above, and we study how Algorithm 1 behaves in the case when P is equal to S.

LEMMA 3.1: *If, during the execution of Algorithm 1, $\tilde{x}$ is equal to x^i, then $\tilde{x}$ is set to x^{i+1} by the next augmentation step, regardless of the value of μ.*

Proof. Let us compute the value of

$$\frac{c{\cdot}(x^j - x^i)}{\|x^j - x^i\|_1} \tag{1}$$

where $j \neq i$. If j is less than i, then $x^j - x^i$ has no positive coordinate and at least one negative coordinate. As a consequence, $c{\cdot}(x^j - x^i)$ is negative, as well as the ratio (1). If j is greater than i, then

$$\frac{c{\cdot}(x^j - x^i)}{\|x^j - x^i\|_1} = \frac{1}{j-i}\sum_{k=i}^{j-1} c_{d-k},$$

where c_{d-k} is the $(d-k)$th coordinate of c. As $c_{d-i} > c_{n-k}$ when $k > i$,

$$\frac{c{\cdot}(x^j - x^i)}{\|x^j - x^i\|_1} \leq c_{n-i},$$

with equality if and only if $j = i + 1$. In other words, when $j > i + 1$

$$\frac{c{\cdot}(x^j - x^i)}{\|x^j - x^i\|_1} < \frac{c{\cdot}(x^{i+1} - x^i)}{\|x^{i+1} - x^i\|_1}.$$

Therefore, if at the beginning of a step during the execution of Algorithm 1, $\tilde{x}$ is equal to x^i where $i < n$, then x will be set to x^{i+1} in Line 3, and the next augmentation will set $\tilde{x}$ to x^{i+1} as announced. ∎

THEOREM 3.2: *Starting at x^0, Algorithm 1 requires n augmentation steps and $\log n\|c\|_\infty + 1$ halving steps in order to maximize $c{\cdot}x$ over S. With early stopping, the number of required halving steps decreases to $\log\|c\|_\infty + 1$.*

Proof. Note that the optimal solution of the problem is x^n. According to Lemma 3.1, Algorithm 1 performs n augmenting steps to reach x^n from x^0. As a consequence, it suffices to observe that this algorithm performs at least $\log n\|c\|_\infty + 1$ halving steps in order to scale μ down to less than $1/n$. ∎

4. Worst-case instances for feasibility-based geometric scaling

Let us now consider a more general description of geometric scaling by modifying Algorithm 1 as described in Section 2: the point x computed in Line 3 of Algorithm 1 can now be any vertex of P that satisfies

$$c \cdot (x - \tilde{x}) > \mu \|x - \tilde{x}\|_1.$$

In particular, x is possibly not a maximizer of the ratio

$$\frac{c \cdot (x - \tilde{x})}{\mu \|x - \tilde{x}\|_1}.$$

Moreover, the condition in Line 4 of Algorithm 1 should be modified in such a way that μ is halved when no such point exists (and otherwise, this is an augmentation step and $\tilde{x}$ is set to x). This results in Algorithm 2.

Algorithm 2: Geometric Scaling via a feasibility test

Input: a 0/1 polytope P contained in $\mathbb{R}^n$,
a vector c in $\mathbb{Z}^n$,
a vertex x^0 of P, and
a number μ_0 greater than $\|c\|_\infty$.

Output: A vertex $x^\star$ of P that maximizes $c \cdot x$.

$\mu \leftarrow \mu_0$, $\tilde{x} \leftarrow x^0$
repeat
  compute a vertex x of P such that $c \cdot (x - \tilde{x}) > \mu \|x - \tilde{x}\|_1$
  if *there is no such vertex of* P **then**
    $\mu \leftarrow \mu / 2$ (halving step)
  else
    $\tilde{x} \leftarrow x$ (augmenting step)
  end
until $\mu < 1/n$;
Return $\tilde{x}$

A close inspection of the behavior of that new algorithm shows that, while it might not use the ratio-maximal solution contained in P, these solutions approximately maximize the ratio and in fact provide a 2-approximation of the maximum ratio. More precisely,

$$\frac{1}{2}\max_{y\in v(P)\setminus\{\tilde{x}\}}\frac{c\cdot(y-\tilde{x})}{\mu\|y-\tilde{x}\|_1}\leq\frac{c\cdot(x-\tilde{x})}{\mu\|x-\tilde{x}\|_1}\leq\max_{y\in v(P)\setminus\{\tilde{x}\}}\frac{c\cdot(y-\tilde{x})}{\mu\|y-\tilde{x}\|_1}.$$

We show in this section that Algorithm 2 sometimes requires

$$n/3+\log n\|c\|_\infty+1$$

steps to reach optimality. In order to do that, we will use the same simplex S as in Section 3, with vertices x^0 to x^n but a different vector c whose coordinates are exponential. More precisely, c is the vector whose ith coordinate is 2^i:

$$c=(2,4,\ldots,2^n).$$

Note that, as in Section 3, we will start the algorithm at vertex x^0.

LEMMA 4.1: *Assume that, at the start of some step during the execution of Algorithm 2, $\tilde{x}$ is equal to x^i. If, in addition, $\mu<c_{n-i}\leq 2\mu$, that step ends with an augmentation that sets $\tilde{x}$ to either x^{i+1}, x^{i+2}, or x^{i+3}.*

Proof. We proceed as in the proof of Lemma 3.1 by computing

$$\frac{c\cdot(x^j-x^i)}{\|x^j-x^i\|_1} \tag{2}$$

when $j\neq i$. If $j<i$, this ratio is negative because x^j-x^i has at least one negative coordinate and none of its coordinates is positive. In particular, the next augmentation cannot set $\tilde{x}$ to x^j. Now assume that $j>i$. In this case,

$$\begin{aligned}\frac{c\cdot(x^j-x^i)}{\|x^j-x^i\|_1} &= \frac{1}{j-i}\sum_{k=i}^{j-1}c_{n-k},\\ &= \frac{1}{j-i}\left(\sum_{k=i}^{n}2^{n-k}-\sum_{k=j}^{n}2^{n-k}\right),\\ &= \frac{2^{n-i+1}-2^{n-j+1}}{j-i},\\ &= 2^n\frac{2^{1-i}-2^{1-j}}{j-i}.\end{aligned}$$

If in addition $\mu<c_{n-i}\leq 2\mu$, then

$$2^i\mu<2^n\leq 2^{i+1}\mu.$$

As a consequence,

$$2\frac{1-2^{i-j}}{j-i}\mu < \frac{c\cdot(x^j - x^i)}{\|x^j - x^i\|_1} \leq 4\frac{1-2^{i-j}}{j-i}\mu.$$

As the ratio $(1-2^{-t})/t$ is less than $1/4$ when t belongs to $[4,+\infty[$, the step cannot end with an augmentation that sets $\tilde{x}$ to x^j where $j \geq i+4$. Now observe that this ratio is equal to $1/2$ when t is equal to 1. Hence,

$$\frac{c\cdot(x^{i+1} - x^i)}{\|x^{i+1} - x^i\|_1} > \mu.$$

This proves that the step will end by an augmentation that sets $\tilde{x}$ to one of the vertices x^{i+1}, x^{i+2}, or x^{i+3}, as desired. ∎

THEOREM 4.2: *Considering the n-dimensional simplex S, the vector*

$$c = (2, 4, \ldots, 2^n),$$

and starting at the origin, Algorithm 2 requires $n/3$ augmentation steps and $\log n\|c\|_\infty + 1$ *halving steps in order to maximize $c{\cdot}x$ over S. With early stopping, the number of required halving steps decreases to* $\log\|c\|_\infty + 1$.

Proof. Observe again that Algorithm 2 performs at least $\log n\|c\|_\infty + 1$ halving steps. Theorem 4.2 then follows from Lemma 4.1 and from the observation that, after a halving step where $\tilde{x}$ is equal to x^i, either c_{n-i} is less than μ (in which case the next step is also a halving step) or satisfies $\mu < c_{n-i} \leq 2\mu$. ∎

5. The tradeoff between scaling and oracle accuracy

In this section, we consider a generalization of Algorithm 2 where, in Line 5, μ is divided by α instead of by 2. This modified algorithm will be referred to as Algorithm 2_α. Note that Algorithm 2 is recovered simply by setting $\alpha = 2$. Whole μ is no longer halved, we still refer to this operation as a halving step. The parameter α controls the amount of both augmenting and halving steps performed by the algorithm. If α is close to 1, then only a small region is made feasible after each halving step. In this case, the feasibility oracle in Line 3 of Algorithm 2 has few choices for feasible solutions and its ability to find the best possible feasible point is not important. If, on the contrary α is large, then many new points will be feasible after each halving step. In fact, for large enough values of α, Algorithm 2_α will be completely *descaled* as all the vertices of the polytope will be made feasible after the first halving step. In this case, the number of steps required to reach an optimal solution is completely determined by the ability of the feasibility oracle (called in Line 3 in Algorithm 2) to reach optimality. In other words, α also controls whether the complexity of the procedure is mainly due to the augmenting steps or to the accuracy of the feasibility oracle.

It turns out that α also explains the gap between the lower bounds provided by Theorems 3.2 and 4.2 on the complexity of geometric scaling. In particular, we will show how the term $n/3$ in the latter lower bound depends on α.

We consider, again, the same simplex S as in Sections 3 and 4 but use an objective vector whose ith coordinate is $\lceil\alpha\rceil^i$:

$$c = (\lceil\alpha\rceil, \lceil\alpha\rceil^2, \ldots, \lceil\alpha\rceil^n).$$

LEMMA 5.1: *Assume that, at the start of some step during the execution of Algorithm 2_α, $\tilde{x}$ is equal to x^i. If, in addition, $\mu < c_{n-i} \leq \alpha\mu$, that step ends with an augmentation that sets $\tilde{x}$ to x^j where $j > i$ and*

$$\alpha\lceil\alpha\rceil\frac{1-\lceil\alpha\rceil^{i-j}}{j-i} > 1. \tag{3}$$

Proof. Let us compute the ratio

$$\frac{c\cdot(x^j - x^i)}{\|x^j - x^i\|_1} \tag{4}$$

when $j \neq i$. As in the proof of Lemma 4.1, this ratio is negative when $j < i$. In that case, the next augmentation will not set $\tilde{x}$ to x^j. If, on the contrary, $j > i$ then the same calculation as in the proof of Lemma 4.1 yields

$$\frac{c\cdot(x^j - x^i)}{\|x^j - x^i\|_1} = \lceil\alpha\rceil^n\frac{\lceil\alpha\rceil^{1-i} - \lceil\alpha\rceil^{1-j}}{j-i}.$$

Now assume that $\mu < c_{n-i} \leq \alpha\mu$. In that case,

$$\lceil\alpha\rceil^i\mu < \lceil\alpha\rceil^n \leq \alpha\lceil\alpha\rceil^i\mu,$$

and it immediately follows that

$$\lceil\alpha\rceil\frac{1-\lceil\alpha\rceil^{i-j}}{j-i}\mu < \frac{c\cdot(x^j - x^i)}{\|x^j - x^i\|_1} \leq \alpha\lceil\alpha\rceil\frac{1-\lceil\alpha\rceil^{i-j}}{j-i}\mu.$$

First observe that, when $j = i+1$, the first inequality is

$$(\lceil\alpha\rceil - 1)\mu < \frac{c\cdot(x^{i+1} - x^i)}{\|x^{i+1} - x^i\|_1}.$$

As $\alpha > 1$, it follows that the step will end by an augmentation. Moreover that augmentation can set $\tilde{x}$ to x^{i+1}. Finally, if the augmentation sets $\tilde{x}$ to x^j, then j must satisfy (3) by the second inequality. ∎

Now denote by ω_α the number of integers t such that

$$\alpha\lceil\alpha\rceil\frac{1-\lceil\alpha\rceil^{-t}}{t} > 1.$$

As already noted in the proof of Lemma 5.1, that inequality is always satisfied when $t = 1$ because $\alpha > 1$, and thus $\omega_\alpha \geq 1$. One can check that the first few values of ω_α are $\omega_\alpha = 1$ when

$$1 < \alpha \leq \frac{4}{3},$$

$\omega_\alpha = 2$ when

$$\frac{4}{3} < \alpha \leq \frac{12}{7},$$

and $\omega_\alpha = 3$ when

$$\frac{12}{7} < \alpha \leq 2.$$

Then, ω_α jumps to 6 when

$$2 < \alpha \leq \frac{729}{364}$$

because $\lceil\alpha\rceil$ is no longer equal to 2, but to 3. Further note that ω_α grows like α^2 when α goes to infinity.

THEOREM 5.2: *Considering the n-dimensional simplex S, the vector*

$$c = (\lceil\alpha\rceil, \lceil\alpha\rceil^2, \ldots, \lceil\alpha\rceil^n),$$

and starting at the origin, Algorithm 2_α requires n/ω_α augmentation steps and $\log n\|c\|_\infty + 1$ halving steps to maximize $c{\cdot}x$ over S. With early stopping, only $\log \|c\|_\infty + 1$ halving steps are required.

Proof. Recall that Algorithm 2_α is identical to Algorithm 2, except that μ is divided by α in Line 5, and thus still performs $\log n\|c\|_\infty + 1$ halving steps. Theorem 5.2 then follows from Lemma 5.1. Indeed, as $\lceil\alpha\rceil \geq \alpha$, after an α-halving step where $\tilde{x}$ is equal to x^i, either c_{n-i} is less than μ (in which case the next step is also an halving step) or satisfies $\mu < c_{n-i} \leq \alpha\mu$ (in which case the next step is an augmenting step) and in the latter case, it is a consequence of Lemma 5.1 that at most ω_α vertices of S are feasible. ∎

Note that Theorem 4.2 is the special case of Theorem 5.2 obtained when $\alpha = 2$. Indeed, in this case, ω_α is equal to 3 and, therefore at most three new vertices are made feasible after each halving step. However, choosing $\alpha = 4/3$ (or, in fact, any α satisfying $1 < \alpha \leq 4/3$) provides Corollary 5.3 because in that case, ω_α is only equal to 1. More precisely, just as Algorithm 1 requires n augmentation steps with the vector

$$c = (1, 2, \dots, n),$$

Algorithm $2_{4/3}$ requires n augmentation steps with the vector

$$c = \left(\left\lceil \frac{4}{3} \right\rceil, \left\lceil \frac{4}{3} \right\rceil^2, \dots, \left\lceil \frac{4}{3} \right\rceil^n\right) = (2, 4, 8, \dots, 2^n)$$

in order to maximize $c{\cdot}x$ over S .

COROLLARY 5.3: *Considering the n-dimensional simplex S, the vector*

$$c = (2, 4, 8, \dots, 2^n),$$

and starting at the origin, Algorithm 2_α with $\alpha = 4/3$ requires n augmentation steps and $\log n\|c\|_\infty + 1$ halving steps to maximize $c{\cdot}x$ over S. With early stopping, only $\log \|c\|_\infty + 1$ halving steps are required.

6. A few simple upper bounds and open questions

We complement the above by providing a few simple upper bounds for geometric scaling that indicate the potential structure of stronger lower bound instances. Without loss of generality we assume from now on that $c \in \mathbb{Z}^n$ and we use the stronger upper bounds via early stopping.

Remark 6.1 (Upper bound induced by few objective values): As the objective is strictly increasing, the number of required augmentation steps is at most the number of different values that $c{\cdot}x$ can take over $v(P)$ and in particular at most $|v(P)| - 1$ as we start from a point $x^0 \in v(P)$.

Remark 6.2: All our lower bounds are obtained by maximizing $c{\cdot}x$ over a n-dimensional simplex S where c is a vector such that $\|c\|_\infty$ is equal to n or to 2^n. In that case, $|v(S)| - 1$ is precisely n. Thus, a natural upper bound for the number of required total steps for geometric scaling over S, assuming early stopping, is $O(n + \log \|c\|_\infty)$. In this setting, our bounds are essentially tight for both Algorithm 1 and Algorithm 2.

A close inspection of our lower bounds reveals that the dependance of the complexity on $\|c\|_\infty$ is exclusively reflected in the number of halving steps but not in the number of augmenting steps which depend exclusively, and linearly, on the dimension n. In fact, with early stopping, the situation is

$$\underbrace{n}_{\text{augmenting steps}} + \underbrace{\log \|c\|_\infty}_{\text{halving steps}} \qquad \text{vs.} \qquad \underbrace{n}_{\text{augmenting steps}} \cdot \underbrace{\log \|c\|_\infty}_{\text{halving steps}},$$

where the left-hand side is our lower bound and the right-hand side is the best-known upper bound for geometric scaling over 0/1 polytopes, leaving a challenging gap. As mentioned earlier these bounds are also in stark contrast to bit scaling, where instances with an arbitrary number of augmenting steps can be created by forcing the algorithm to revisit points in the various scaling phases. It remains open whether it is possible to create a geometric scaling instance for which the number of required augmentation steps depends on the encoding length $\log \|c\|_\infty$ of the vector c.

The polynomial time preprocessing given by Frank and Tardos in [6] allows to preprocess the vector c in such a way that $\log \|c\|_\infty = O(n^3)$ without changing the optimal solution. As a consequence, an overall complexity that is polynomial in n can be achieved for the total number of steps. However, this does not resolve the complexity question for a moderately sized vector c and in particular it also changes the considered instance.

Acknowledgments. We thank the 2021 HIM program Discrete Optimization during which part of this work was developed.

References

[1] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig, *Log-barrier interior point methods are not strongly polynomial*, SIAM Journal on Applied Algebra and Geometry **2** (2018), no. 1, 140–178.

[2] Xavier Allamigeon, Pascal Benchimol, Stéphane Gaubert, and Michael Joswig, *What tropical geometry tells us about the complexity of linear programming*, SIAM Review **63** (2021), no. 1, 123–164.

[3] Pierre Le Bodic, Jeffrey W. Pavelka, Marc E. Pfetsch, and Sebastian Pokutta, *Solving MIPs via scaling-based augmentation*, Discrete Optimization **27** (2018), 1–25.

[4] Antoine Deza and Lionel Pournin, *Primitive point packing*, Mathematika, *to appear*.

[5] Antoine Deza, Lionel Pournin and Noriyoshi Sukegawa, *The diameter of lattice zonotopes*, Proceedings of the American Mathematical Society **148** (2020), no. 8, 3507–3516.

[6] András Frank and Éva Tardos, *An application of simultaneous diophantine approximation in combinatorial optimization*, Combinatorica **7** (1987), no. 1, 49–65.

[7] Victor Klee and George J. Minty, *How good is the simplex algorithm?*, Inequalities III (Oved Shisha, ed.), Academic Press, New York, 1972, pp. 159–175.

[8] Alberto Del Pia and Carla Michini, *Short simplex paths in lattice polytopes*, Discrete & Computational Geometry **67** (2018), no. 2, 503–524.

[9] Sebastian Pokutta, *Restarting algorithms: sometimes there is free lunch*, Lecture Notes in Computer Science **12296** (2020), 22–38.

[10] Andreas S. Schulz and Robert Weismantel, *The complexity of generic primal algorithms for solving general integer programs*, Mathematics of Operations Research **27** (2002), no. 4, 681–692.

[11] Andreas S. Schulz, Robert Weismantel and Günter M. Ziegler, *0/1-integer programming: Optimization and augmentation are equivalent*, European Symposium on Algorithms '95, Springer, 1995, pp. 473–483.