Exponential Lower Bounds for History-Based Simplex Pivot Rules on Abstract Cubes*

Antonis Thomas

Department of Computer Science, Institute of Theoretical Computer Science, ETH Zürich, Zürich, Switzerland
athomas@inf.ethz.ch

Abstract

The behavior of the simplex algorithm is a widely studied subject. Specifically, the question of the existence of a polynomial pivot rule for the simplex algorithm is of major importance. Here, we give exponential lower bounds for three history-based pivot rules. Those rules decide their next step based on memory of the past steps. In particular, we study Zadeh's least entered rule, Johnson's least-recently basic rule and Cunningham's least-recently considered (or round-robin) rule. We give exponential lower bounds on Acyclic Unique Sink Orientations of the abstract cube, for all of these pivot rules. For Johnson's rule our bound is the first superpolynomial one in any context; for Zadeh's it is the first one for AUSO. Those two are our main results.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems, G.2.1 Combinatorics

Keywords and phrases pivot rule, lower bound, exponential, unique sink orientation, zadeh

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.69

1 Introduction

The existence of a polynomial time pivot rule for the simplex algorithm is a major open problem in the theory of optimization. Most known rules have superpolynomial lower bounds by now. For deterministic rules, in particular, it is the case that many of them admit exponential lower bounds. Klee and Minty with their seminal paper [16], already in 1972, gave an exponential lower bound for Dantzig's original pivot rule. Their construction has been heavily studied ever since (for example [10],[3]) and inspired many later lower bounds.

In this paper, we are interested in a family of deterministic pivot rules known as history-based (or having memory). For those, superpolynomial lower bounds seemed to be elusive until recently. Arguably, the most famous history-based rule is due to Zadeh. Known as the *least entered* rule, it was described in 1980 with a technical report that was reprinted in 2009 [26]. This rule keeps a history of how many times each improving direction has been used and, at every step, chooses one that minimizes this history (a tie-breaking rule takes care of ties). The least entered rule was specifically designed to attack constructions similar to the Klee-Minty by using the improving directions in a balanced way (note that in this regard, it is similar to a random walk). With a letter to Klee in the 80s, Zadeh offered a $1000 prize to anyone who can prove polynomial upper or superpolynomial lower bounds for the least entered rule. This prize was claimed in 2011, by Friedmann [6], with a superpolynomial lower bound on actual Linear Programs (LP). No non-trivial upper bounds are known for this rule.

* A full version of the paper is available at https://arxiv.org/abs/1706.09380.

Another interesting rule was suggested by Cunningham [4], known as the *least-recently considered* rule. It fixes an initial ordering on all improving directions and then selects one in a round-robin fashion, starting from the last direction selected. The history here is to remember which was the last used improving direction. Furthermore, the *least-recently basic* rule, which Cunningham attributes to Johnson, was also first discussed in the same paper [4]. That rule selects the improving direction that left the basis least recently (in other words the direction whose opposite was selected least recently). For a detailed exposition on those and many other history-based pivot rules, the interested reader should look at Aoshima *et al.* [1].

We provide *exponential lower bounds*, by means of Acyclic Unique Sink Orientations, for all three aforementioned history-based rules.

Unique Sink Orientations. (USO) is an abstract framework that generalizes LP (and other problems). It was originally described by Stickney and Watson [23] and later revived by Szabó and Welzl [24]. Such abstract frameworks have received lots of attention since the discovery of the Random Facet pivot rule: Kalai [15] and, independently, Matoušek, Sharir and Welzl [19] proved subexponential upper bounds for this rule on LP. It became evident that their analysis made use only of combinatorial properties of LP and, thus, it was possible to extend their upper bounds in a much more abstract setting [8].

The most well-studied such framework is that of USO (e.g. [18],[5],[11] and see also below). Intuitively, a USO is an orientation of the hypercube graph such that every non-empty face has a unique sink (vertex with only incoming edges). The computational problem is to discover the unique global sink by performing vertex evaluations (each one reveals the orientation of the edges incident to the vertex). Commonly, acyclic USO (AUSO) constructions have served as lower bounds for pivot algorithms (e.g. [17], [22], [20], [14]) and our lower bounds are also manifested as AUSO.

Prior work and open questions. Aoshima *et al.* [1] explore the possibility that there exist AUSO on which history-based pivot rules take a Hamiltonian path. They prove, with the help of computers, that Zadeh's pivot rule admits such Hamiltonian paths up to dimension 9 at least. On the contrary, they show that Johnson's rule (among others) does not admit Hamiltonian paths and, so, they ask if it admits exponential paths on AUSO.

Recently, Avis and Friedmann [2] gave the first exponential lower bound for history-based rules. Namely, they prove an exponential lower bound for Cunningham's rule on binary parity games (definitions in [2]). Their constructions translate immediately to linear programs and also AUSO, for which[1] the lower bound is $\Omega(2^{n/5})$. However, they are very complicated and, thus, the authors ask if it is possible to prove exponential lower bounds for this rule, on AUSO, in a simpler manner.

Moreover, they compare their construction to the one for Zadeh's rule [6]. The latter gives a family of non-binary parity games (which correspond to linear programs), where Zadeh's rule takes a subexponential number of steps, of the form $2^{\Omega(\sqrt{n})}$ (where n is the number of variables of the LP). Although binary parity games correspond directly to AUSO, the same is not known for non-binary ones. Hence, Avis and Friedmann ask [2] if superpolynomial lower bounds for Zadeh's rule exist also on AUSO. In addition, Friedmann's lower bound [6] is based on a tie-breaking rule which is *artificial* in the sense that it always works in favor of

[1] The exact translation of binary Parity Games to AUSO is explained in [2]. Roughly, their constructed binary parity game translates to a cube of dimension $5n'$, where the path that the algorithm will take is of length $2^{n'}$. Thus, for n-dimensional AUSO, that is a lower bound of the form $\Omega(2^{n/5})$.

the lower bound designer. It is not described in the paper because, as the author writes, it is "not a natural one". Thus, he raises the question [6] of whether it is possible to obtain a lower bound with a *natural* tie-breaking rule.

Finally, Avis and Friedmann write [2]: "More generally it is of interest to determine whether all of the history based rules mentioned in [1] have exponential behaviour on AUSO".

Our results. With Theorem 4, we give an exponential lower bound for Johnson's rule. This is the *first superpolynomial* lower bound for this algorithm. Moreover, we give an exponential lower bound for Zadeh's rule, with Theorem 8. This has a number of advantages compared to the known construction: Firstly, it is exponential, whereas Friedmann's lower bounds [6] are subexponential (also *not* known to translate to AUSO). Secondly, our constructions are much simpler to describe. Finally, it is based on a tie-breaking rule that is essentially as *simple* as possible: a fixed ordered list. These two lower bounds constitute the *main results* of this paper. With Theorem 3, we give an exponential lower bound for Cunningham's rule. The advantage here is that the construction is significantly simpler; the lower bound also happens to be slightly improved. Theorem 3 serves as a warm-up to the main results by introducing the techniques and notation we use for our constructions.

Therefore, we answer to the positive all the questions described in the previous paragraph. Due to space constraints, many details and some proofs are missing from this extended abstract; a full version with all the details and a more complete analysis can be found at [25].

Our methods. The constructions in this paper are based on the building tools originally developed by Schurr and Szabó [21]; we do, however, introduce some novel ideas needed to deal with history-based pivot rules. Most known inductive lower bound constructions (e.g. [21],[22],[20],[14]) embed copies of the previous construction into the next one, in such a way that the algorithm gets trapped in the previous construction twice. For Zadeh's rule this does not work: it balances the directions being used and it inevitably escapes the second trap (at the next inductive step). To overcome this, we build a trap that consists of a small number of copies, being connected in a careful way which ensures that the algorithm uses the improving directions in a *balanced* fashion: it follows the path of the previous construction, up to making additional "balancing moves" between different copies.

Lower bounds on AUSO. It is not clear if AUSO lower bound constructions (including ours) can be realized as LP. However, the abstract setting allows for simpler proofs that are easy to communicate. We, thus, believe that such constructions are relevant for understanding the behavior of the pivot rules; the ideas could be used for the design of LP-based exponential lower bounds. For example, the first subexponential lower bounds for Random Facet [17] (tight to the upper bound; see also [9]) and for Random Edge [20] (at every step chooses one improving direction at random) were both proved by AUSO constructions. Indeed, for these two rules, subexponential lower bounds have been later proved on actual LP [7]. The most recent lower bound on AUSO was by Hansen and Zwick in 2016 [14], where they improve the subexponential lower bound for Random Edge. Note that for this rule non-trivial exponential upper bounds are known in the general case [13] and under assumptions [12].

2 Preliminaries

Let $[n] = \{1, \ldots, n\}$ and $\pm[n] = \{-n, \ldots, -1, 1, \ldots, n\}$. Let $Q^{[n]} = 2^{[n]}$ be the set of vertices of the n-dimensional hypercube over coordinates in $[n]$. Often we write Q^n (the superscript indicates the dimension). A vertex of the hypercube $v \in Q^n$ is denoted by the set of coordinates it contains. Generally, with $C \subseteq [n]$ we denote a set of coordinates. Consider two vertices $v, u \in Q^n$. With $v \oplus u$ we denote the symmetric difference of the two sets. Now, let $C \subseteq 2^{[n]}$ and $v \in Q^n$. A *face* of the hypercube, $F(C, v)$, is defined as the set of vertices that are reached from v over the coordinates defined by any subset of C, i.e. $F(C, v) = \{u \in Q^n | v \oplus u \subseteq C\}$. The dimension of the face is $|C|$. We call edges the faces of dimension 1, e.g. $F(\{j\}, v)$. For $k \leq n$ we call a face of dimension k a k-face.

Let ψ denote an orientation of the edges of the hypercube Q^n. Consider two vertices $v, u \in Q^n$ and a coordinate $j \in [n]$. The notation $v \xrightarrow{j} u$ (w.r.t ψ) means that $F(\{j\}, v) = \{v, u\}$ and that the corresponding edge is oriented from v to u in ψ. Sometimes we write $v \rightarrow u$, when the coordinate is irrelevant. An edge $v \xrightarrow{j} u$ is forward if $j \in u$ and otherwise we say it is backward. We use $v \rightsquigarrow w$ to denote (that there is) a directed path from v to w.

We now define the concept of *direction*; the algorithms that we study here have memory of the directions that have been used so far. A direction is a signed coordinate. Let $c \in C$ be a coordinate; two different directions correspond to c, $+c$ and $-c$. At a vertex v the direction $+c$ corresponds to a forward edge incident to v and $-c$ to a backward edge. We say that a direction is *available* at vertex v if the corresponding edge is outgoing. Thus, at each vertex if a coordinate is incoming then none of the directions is available and if a coordinate is outgoing then exactly one of the directions is available. Similarly to above, we write $v \xrightarrow{d} u$, for some direction d. Note that if we have $v \xrightarrow{+c} v'$ (similarly $-c$) then at v' neither $+c$ nor $-c$ can be available. Generally, we denote with $D \subseteq \pm[n]$ a set of directions. Given a set of coordinates C, we say that D is the set of directions that corresponds to C to mean $D = \{-c, +c \mid c \in C\}$. Often, we use d to denote a direction without specifying its sign.

Then, ψ is a *Unique Sink Orientation* (USO) of Q^n when every non-empty face has a unique sink. USO can be either cyclic or acyclic (for these we write AUSO). n-AUSO means an AUSO over Q^n. For a USO ψ, we define s_ψ, the outmap function: for every $v \in Q^n$, $s_\psi(v) = \{j \in [n] | v \xrightarrow{j} (v \oplus \{j\})\}$, that is the set of coordinates on which v has an outgoing edge. A sink of a face $F(C, v)$ is a vertex $u \in F(C, v)$, such that $s_\psi(u) \cap C = \emptyset$. The whole cube is a face of itself; thus, there is a unique vertex v, the *global sink* with $s(v) = \emptyset$. In the rest, we write $s(v)$ to denote the outmap of v (ψ will be clear from the context).

The computational problem associated with a USO is to find the global sink. The computational model is the *vertex oracle* model. We have access to an oracle such that when we give it a vertex v, it replies with the outmap $s(v)$ of v. This is the standard computational model in USO literature and all the lower and upper bounds are with respect to it.

We are now ready to state the Product and Reorientation lemmas (due to [21]) which are the building tools for our constructions. The following constitutes an intuitive description of the Product lemma which is relevant to us: Consider an n-AUSO A (oriented hypercube graph) and take 2^m copies of A. For every vertex $v \in A$, take an m-AUSO A_v, the *connecting frame* for v. Each copy of A corresponds to a vertex of the frame A_v and v in this copy of A is connected according to that vertex of A_v. The result is an $(n + m)$-AUSO. Formally:

▶ **Lemma 1** (Product [21]). *Let C be a set of coordinates, $C' \subseteq C$ and $\bar{C}' = C \setminus C'$. Let $\tilde{s}$ be a USO outmap on $Q^{C'}$. For each vertex $u \in Q^{C'}$ we have a USO outmap s_u on $Q^{\bar{C}'}$. Then, the orientation defined by the outmap $s(v) = \tilde{s}(v \cap C') \cup s_{v \cap C'}(v \cap \bar{C}')$ on Q^C is a USO. Furthermore, if $\tilde{s}$ and all s_u are acyclic so is s.*

The Reorientation lemma, which follows, can be intuitively explained this way: if we have a USO and there is a face, such that all the vertices in this face have the same outmap on the edges external to the face, then we can reorient this face according to any other USO.

▶ **Lemma 2** (Reorientation [21]). *Let C be a set of coordinates, $C' \subseteq C$ and $\bar{C}' = C \setminus C'$. Let s be a USO on Q^C and let $\mathcal{F} = F(C', u)$, for some $u \in Q^C$, be a face of Q^C. If, for any two vertices $v, w \in \mathcal{F}$, $s(v) \cap \bar{C}' = s(w) \cap \bar{C}'$ and $\tilde{s}$ is a USO on $Q^{C'}$, then the outmap $s'(v) = \tilde{s}(v \cap C') \cup (s(v) \cap \bar{C}')$ for $v \in \mathcal{F}$ and $s'(v) = s(v)$ otherwise is a USO on Q^C.*

3 A warm-up: Cunningham's Rule

▶ **Theorem 3.** *There exists n-AUSO such that Cunningham's rule, with a suitable starting vertex and list, takes a path of length at least $2^{n/4}$.*

We will sketch a proof for the above theorem. But let us start with some general comments and definitions that will apply to all our constructions. Firstly, they are inductive. Let A_i be the ith step of the induction. The base case is A_0. We call C_i a bundle of coordinates; that is the set of coordinates that was added at the ith step of induction (and C_0 are the coordinates of the base case). We also define $C_i^+ = \bigcup_{k=0}^{i} C_k$. Then, D_i denotes the set of directions that corresponds to C_i and similarly for D_i^+. Let v_0^i be the starting vertex for A_i. Consider that there is a token, which is initially on v_0^i, and at every step moves according to the direction that the given algorithm chooses. The path that the token takes from v_0^i to the unique sink, on A_i, is denoted with P_i; its length is denoted with $|P_i|$.

To construct A_{i+1} from A_i, we take 2^m copies of A_i and connect their vertices with m-dimensional connecting frames, for some constant m (Lemma 1). Afterwards, we perform one reorientation (Lemma 2), to install a simple balancing gadget. The token starts at the starting vertex v_0^{i+1} and walks on a path P (in A_{i+1}) until it reaches a vertex that has all coordinates from C_i^+ incoming. This vertex corresponds to the sink of A_i. If we project the path P to only the directions from D_i^+ we get exactly P_i. In the balancing gadget the token will be taken back to the vertex that corresponds to the starting vertex for A_i. The idea is to prove that if we project the rest of the token's path to the global sink, to only the directions from D_i^+, we get again P_i. Thus, $|P_{i+1}| > 2|P_i|$. Let $T(n)$ denote the length of the corresponding paths on an n-AUSO. The recursion we get then is $T(n+m) > 2T(n)$. This gives rise to exponential lower bounds of the form $2^{n/m}$. For Cunningham's and Johnson's rules the constant is $m = 4$ and for Zadeh's $m = 6$.

The lower bound. Consider that the algorithm runs on an n-AUSO. It has an ordered list L that contains all $2n$ directions; let $L[k]$ indicate the kth direction on the list. There is a marker μ of which direction was used last: if direction $L[k]$ was used at the last step then $\mu = k$. At the next step the algorithm will start checking the directions on the list from $L[\mu+1]$ in a cyclic order (so if it reaches $L[2n]$ it continues from $L[1]$) and it chooses the first available one. Initially, $\mu = 2n$ so that the first direction that the algorithm checks is $L[1]$.

We will now give a short sketch of the lower bound. Full details can be found in the full version [25]. Let A_0 be the base case and $L_0 = (+c_0^1, -c_0^2, +c_0^3, -c_0^1, +c_0^4, -c_0^3, +c_0^2, -c_0^4)$. To construct A_{i+1} from A_i we take 2^4 copies of A_i which we connect with three different frames. The two crucial ones F_1 and F_2 are given in Figure 1; F_3 can be found in [25]. The new set of coordinates will be $C_{i+1} = \{c_{i+1}^1, c_{i+1}^2, c_{i+1}^3, c_{i+1}^4\}$.

Let us define some notation in reference to Figure 1. An AUSO is given as a collection of 2-faces on the first two coordinates. All coordinates are labeled. Each square represents a face on coordinates C_i^+. All of these faces, except $\boxed{\text{B}}$, are internally oriented according to A_i (correspond to copies). The numbers are indicating in which order the token will visit them. We refer to these faces in the text; for example, we write $\boxed{1}$ to mean the face: $F(C_i^+, \{c_{i+1}^2\})$. Given a vertex v, we write $v\perp\boxed{1}$ to mean the vertex $v' \in \boxed{1}$, such that $v \cap C_i^+ = v' \cap C_i^+$.

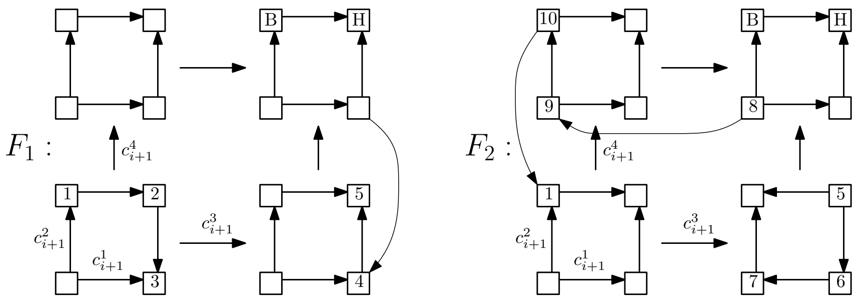

Figure 1 The orientations F_1 and F_2, used as connecting frames, are given in this figure. The 4-dimensional frames are split in 2-faces of coordinates c^1_{i+1} and c^2_{i+1}. The arrows on the other coordinates indicate the orientation of all the edges on this coordinate except when noted differently. For example, in F_2 all edges on coordinate c^3_{i+1} are oriented from left to right except the edge $\boxed{9} \leftarrow \boxed{8}$. Each $\boxed{\cdot}$ represents a face on C_i^+. This notation is valid also for the next figures.

Moreover, we write $\boxed{1} \rightsquigarrow \boxed{5}$ to mean a path from a vertex in $\boxed{1}$ to the corresponding vertex in $\boxed{5}$, using only directions from D_{i+1}. In this case, the exact vertex will be clear from the context. The face $\boxed{\text{B}}$ is the one that contains the balancing gadget, which is installed by use of Lemma 2. In this construction and the one of Section 4, $\boxed{\text{H}}$ is a hypersink (has all edges external to the face incoming). In the construction of Section 5 there is no hypersink.

Let us give a short description of the behavior of Cunningham's rule on A_{i+1}. Firstly, the starting vertex $v_0 = v_0^{i+1} = \{c_0^2, \ldots, c_{i+1}^2\}$. Thus, the token is initially placed in $\boxed{1}$. The list $L_{i+1} = L_i \cdot (+1, -2, +3, -1, +4, -3, +2, -4)$, where $\cdot$ represents concatenation. For simplicity, we write a number $\pm k$ instead of $\pm c^k_{i+1}$. So, the directions from bundle D_k have priority over the ones from bundle $D_{k'}$, if $k < k'$. Let $IN = D_i^+$ and $OUT = D_{i+1}$; that is the set of directions from the previous inductive steps and the current one respectively.

The connecting frame is F_1. The algorithm uses directions from IN and the token moves in $\boxed{1}$. When these are exhausted the token takes a path $\boxed{1} \rightsquigarrow \boxed{5}$. The directions from OUT are, then, exhausted. The algorithm uses a direction from IN, in $\boxed{5}$. At the next vertex the frame changes[2] to F_2. When the directions from OUT will be used again (after the ones from IN) the token will take a path $\boxed{5} \rightsquigarrow \boxed{10} \rightarrow \boxed{1}$. After one step in $\boxed{1}$, on a direction from IN, the frame changes to F_1. The conclusion is that the token will keep moving between $\boxed{1}$ and $\boxed{5}$ until it reaches a vertex v_{s_i}, such that $s(v_{s_i}) \cap C_i^+ = \emptyset$. This is not a cycle as the token keeps moving toward v_{s_i} on the IN directions (this was once P_i).

When the token is on s_i the frame will change to F_3. The token will walk to $\boxed{\text{B}}$ and the OUT directions will get exhausted exactly there. Inside $\boxed{\text{B}}$, the algorithm will be forced to take a path back to vertex $v_0 \perp \boxed{\text{B}}$ (the sink of $\boxed{\text{B}}$). The next time the directions from OUT will be used the token will go in the hypersink $\boxed{\text{H}}$. But there, it will be at vertex $v_0 \perp \boxed{\text{H}}$, the coordinates from C_{i+1} will be incoming from there on and, so, it will stay inside $\boxed{\text{H}}$ where it will perform P_i once again. We can conclude that $|P_{i+1}| > 2|P_i|$ can be proved for this construction, which also proves Theorem 3. Details in the full version [25].

[2] At this point the change of frame can be understood with an adversary argument. We play against the algorithm and we choose which frame to reveal at which vertex. This is consistent with Lemma 1.

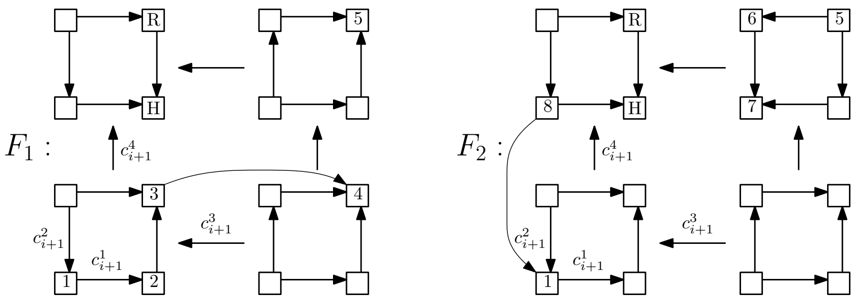

Figure 2 The orientations F_1 and F_2, used as connecting frames, are given in this figure.

4 Exponential lower bound for Johnson's rule

▶ **Theorem 4.** *There exists n-AUSO such that Johnson's rule, with a suitable starting vertex, takes a path of length at least* $2^{n/4}$.

In this section we will prove the above theorem. Let us define Johnson's least-recently basic rule. Consider that the algorithm runs on an n-AUSO. It maintains a history function h which is defined on all $2n$ directions. Let v be the current vertex. Intuitively, the algorithm keeps the following history: Say direction d was used at step x and let $-d$ be the opposite of that direction. Then, at step x we have $h(-d) = x$; this will stay intact until $-d$ is used. On the other hand, $h(d)$ will keep increasing to the current step until $-d$ is used. Formally, for a direction d, $h(d)$ is the last step number when $|d| \in v$ if d is positive and the last step number when $|d| \notin v$ if d is negative. Here, $|d|$ denotes the coordinate that corresponds to direction d. Ties are possible and we assume that they are broken lexicographically. The algorithm chooses from the set of available directions, direction d which minimizes $h(d)$.

The construction. The construction is inductive. Let A_i denote the ith inductive step. The base case A_0 is the 4-dimensional AUSO F_1, shown in Figure 2. The initial set of coordinates is $C_0 = \{c^1_0, c^2_0, c^3_0, c^4_0\}$. The starting vertex is $v_0 = \emptyset$, which is at the vertex labeled $\boxed{1}$ in the figure. Then, the algorithm will go over directions $(+c^1_0, +c^2_0, +c^3_0, +c^4_0, -c^3_0, -c^2_0)$ and will find the sink at $\{c^1_0, c^4_0\}$. Let us now describe how to construct AUSO A_{i+1} from AUSO A_i. Every inductive step adds 4 dimensions. As before, $C^+_j = \bigcup_{k=0}^{j} C_k$. Let the new bundle of coordinates be $C_{i+1} = \{c^1_{i+1}, c^2_{i+1}, c^3_{i+1}, c^4_{i+1}\}$. We take 16 copies of A_i and connect them with the 4-AUSO F_1 and F_2 that appear in Figure 2. In this section, a reset-AUSO (thus, the $\boxed{\text{R}}$ in the figure) will take the role of the balancing gadget.

Let us define what we mean by lexicographic order here: $+c^k_j$ comes before $-c^{k'}_j$ for any k and k'; $+c^k_j$ comes before $+c^{k'}_j$ and $-c^k_j$ comes before $-c^{k'}_j$ if $k < k'$. Finally, d^k_j comes before $d^{k'}_{j'}$ for any k and k' and positive/negative sign, if $j < j'$.

The starting vertex will be $v_0 = \emptyset$ for every inductive step. Then, every positive direction $+c$ initially has $h(+c) = 0$ (until $+c$ is used by the algorithm). At step number 1, one of the positive directions will be used at which point every negative direction $-c$ has $h(-c) = 1$.

With this construction we want to force the algorithm to the following behavior: It starts at $\boxed{1}$ using all the positive directions in lexicographic order. Then it is in $\boxed{5}$, where it will use all the negative directions in lexicographic order. This will continue until the sink of A_i has been reached. It follows that directions from D^+_i are only used when the algorithm is in $\boxed{1}$ or $\boxed{5}$, before the sink of A_i has been discovered. We will later show that this is the case.

Then, we consider the construction as an adversary argument. Firstly, the starting vertex of A_i and the sink of A_i both use F_1 as the connecting frame. Every time the algorithm is in

$\boxed{1}$ and uses a direction from D_i^+, we change (or keep) the connecting frame to F_1. Similarly, when the algorithm arrives in $\boxed{5}$ and uses a direction from D_i^+, we change the connecting frame to F_2. This operation is consistent with Lemma 1: Every vertex is connected with the corresponding frame. The result of this operation is A'_{i+1} which is not the final AUSO.

The final step for the construction of A_{i+1} is to use Lemma 2 to embed a reset-AUSO to the face $\boxed{\text{R}}$. For every $i > 0$, R_i is a $4i$-AUSO (whereas A_i is a $4(i+1)$-AUSO). For the construction of A_{i+1} for A_i we embed R_{i+1} to the face $\boxed{\text{R}}$. R_{i+1} is designed such that it has its sink at vertex $\emptyset$. In addition, it has a path from vertex $\{c_0^1, c_0^4, \ldots, c_i^1, c_i^4\}$ to the vertex $\emptyset$ such that every vertex on this path has only one outgoing edge and the path goes through the negative directions in lexicographic order: $(-c_0^1, -c_0^4, \ldots, -c_i^4, -c_i^4)$.

This reorientation concludes the construction of A_{i+1}. It does not introduce any cycles in A_{i+1}. A formal description of the reset-AUSO and an argument on the acyclicity of A_i can be found in the full version [25]. Note that in $\boxed{\text{H}}$ we still have A_i.

The behavior of Johnson's rule. The behavior of Johnson's rule on the AUSO constructed as above will be described here. We give as much detail as space allows; the rest can be found in the full version [25]. Firstly, we define the tools that we are going to use for this analysis.

Similarly to the previous section, consider a token t. That is a token that starts at the initial vertex v_0 and moves according to the directions that the algorithm chooses. With slight abuse of notation we also use t to refer to the vertex where the token currently lies on. Moreover, we write t_j to mean the set $t \cap C_j^+$; that is, the projection of the vertex t to the set of coordinates C_j^+. Since $t_j \subseteq t$, we call t_j a subtoken.

We say that a coordinate bundle C_j is *active* when $s(t) \cap C_j \neq \emptyset$. Otherwise, we say that C_j is inactive. Note that for both the 4-dimensional frames that we have used, the sink is at the same vertex. This implies that C_j is inactive if and only if token t is such that $t \cap C_j = \{c_j^1, c_j^4\}$. Moreover, we say that token t is in $\boxed{1}_j$ to mean that $t \cap C_j = \emptyset$; t is in $\boxed{5}_j$ when $t \cap C_j = C_j$ and similarly for the rest of the faces $\boxed{\cdot}$ from Figure 2.

For each subtoken t_j, we say that it has reached *its sink* when all bundles in C_j^+ are inactive. This means that $t_j = \{c_0^1, c_0^4, \ldots, c_j^1, c_j^4\}$. *Resetting* C_j^+ is a process that happens when subtoken t_j is at its sink: It takes token t from a vertex where $t \cap C_j^+ = \{c_0^1, c_0^4, \ldots, c_j^1, c_j^4\}$ to a vertex where $t \cap C_j^+ = \emptyset$. Moreover, we say that C_j^+ is *resettable* when:

- t_j is at its sink.
- $h(-c_{j'}^1) < h(-c_{j'}^4) < h(-c_{j+1}^2)$, for every $0 \leq j' \leq j$.

The first bullet in the definition above equivalently means that all bundles in C_j^+ are inactive. Resetting C_j^+ is a process that takes place when (and only when) token t is in the reset-AUSO in $\boxed{\text{R}}_{j+1}$. For now assume that C_j^+ will be reset only when it is resettable; we will prove this later (with Lemma 7). Thus, t_j is on its sink when the resetting process starts. The second bullet of the definition of resettable ensures that token t will not go out of $\boxed{\text{R}}_{j+1}$ before C_j^+ has been reset. During the reset of C_j^+, token t will go over negative directions from D_j^+ in this order: $(-c_0^1, -c_0^4, \ldots, -c_j^1, -c_j^4)$. This is because of the construction of the reset-AUSO R_{j+1}; the algorithm has no other choice. We are ready to state the following lemma (a formal proof can be found in the full version [25]).

▶ **Lemma 5.** *Let $t \cap C_j^+ = \emptyset$. Then, all the positive directions from C_j^+ will be used in the lexicographic order: $(+c_0^1, +c_0^2, +c_0^3, +c_0^4, \ldots, +c_j^1, +c_j^2, +c_j^3, +c_j^4)$.*

The above lemma defines the path that token t will follow until subtoken t_j reaches its sink. For every bundle C_j, the positive directions are used in lexicographic order

$(+c_j^1, +c_j^2, +c_j^3, +c_j^4)$ and the token t goes to $\boxed{5}_j$. After this, we have $h(-c_j^1) < h(-c_j^2) < h(-c_j^3) < h(-c_j^4) < h(d)$ for any positive d from D_j. Then, some directions from C_{j-1}^+ will be used and the frame for C_j will change to F_2. When it is the turn of the negative directions from D_j to be used they will be used consecutively and in lexicographic order $(-c_j^1, -c_j^2, -c_j^3, -c_j^4)$; that is, assuming t_{j-1} has not reached its sink yet. Otherwise, the connecting frame for C_j would be F_1 and the directions $-c_j^1$ and $-c_j^4$ would not be available.

After this, t will be in $\boxed{1}_j$ and, moreover, $h(+c_j^1) < h(+c_j^2) < h(+c_j^3) < h(+c_j^4) < h(d)$ for any negative d from D_j. There, after some directions from C_{j-1}^+ are used, the connecting frame for C_j will be F_1. When it is the turn of the positive directions from D_j to be used they will be used consecutively and in in lexicographic order $(+c_j^1, +c_j^2, +c_j^3, +c_j^4)$ and the token t will go back to $\boxed{5}_j$. We can conclude that t will keep moving from $\boxed{1}_j$ to $\boxed{5}_j$ and reversely until subtoken t_{j-1} reaches its sink. The next corollary follows from this discussion.

▶ **Corollary 6.** *Let C_{j+1} be active.*

1. *If t is in $\boxed{1}_{j+1}$, then the positive directions from D_{j+1} will be used (when it is their turn) consecutively and in lexicographic order $(+c_{j+1}^1, +c_{j+1}^2, +c_{j+1}^3, +c_{j+1}^4)$.*
2. *If t is in $\boxed{5}_{j+1}$ and subtoken t_j has not reached its sink, then the negative directions from D_{j+1} will, similarly, be used (when it is their turn) consecutively as $(-c_{j+1}^1, -c_{j+1}^2, -c_{j+1}^3, -c_{j+1}^4)$.*

It follows that directions from D_j^+ are only used when t is in $\boxed{1}_{j+1}$ or in $\boxed{5}_{j+1}$.

The next lemma is the last ingredient needed (proof can be found in the full version [25]).

▶ **Lemma 7.** *Let C_{j+1} be active. When t_j reaches its sink, C_j^+ is resettable.*

Now consider the path of the token t from v_0 to the sink of A_{i+1}. This AUSO is of dimension $n = 4(i+2)$. By Corollary 6, t will be moving back and forth between $\boxed{1}$ and $\boxed{5}$ until t_i reaches its sink. When that happens, C_i^+ is resettable (by Lemma 7) and, when t enters $\boxed{\text{R}}$, C_i^+ will be reset. Following, t enters $\boxed{\text{H}}$ at vertex $v_0 \bot \boxed{\text{H}}$.

When the algorithm started at $v_0 = \emptyset$, all the positive directions from D_{i+1}^+ where used in lexicographic order. Since it is deterministic, this defines completely the behavior of t. Consider the path P that t will follow from v_0 until t_i reaches its sink, but projected on the coordinates from C_i^+. From Lemma 5, we know that when token t is such that $t \cap C_i^+ = \emptyset$, all the positive directions from D_i^+ will be used in the lexicographic order. At this point C_{i+1} is inactive and t is in $\boxed{\text{H}}$. Therein, it will follow the same path P to the global sink.

Let $T(n)$ denote the length of the path that token t will take from v_0 until it reaches the global sink on a n-AUSO. With the above analysis, we have shown that the recursion $T(n+4) > 2T(n)$ holds. This recursion leads to the proof of Theorem 4.

5 Exponential lower bound for Zadeh's Rule

▶ **Theorem 8.** *There exists n-AUSO such that Zadeh's rule, with a suitable starting vertex and tie-breaking rule, takes a path of length at least $2^{n/6}$.*

In this section we will prove the above theorem. Firstly, let us define formally Zadeh's least entered rule. Consider that the algorithm runs on an n-AUSO. It maintains a history function h which is defined on all $2n$ directions. Given a direction d, $h(d)$ is the number of times the direction d has been used. At the beginning $h(d) = 0$, for all d. At every step the algorithm picks one direction from the set of available ones that minimizes the history function. In addition, there is a tie-breaking rule: this is an ordering of the directions and is invoked only in case more than one have the minimum history size. As we already mentioned in Section 1, our lower bound construction will have the *simplest* possible tie-breaking rule, an ordered list which will be given explicitly. This is in contrast to the lower bounds from [6].

Secondly, let us define some tools that we will use for the analysis of the algorithm. We have a *balance* function b, which is also defined on all the $2n$ directions. Let d_{max} be the most used direction; then, $b(d) = h(d_{max}) - h(d)$. This means that direction d has been used $b(d)$ less times compared to d_{max}. We say that a direction d is *imbalanced* when $b(d) > 0$ and that D is balanced when $b(d) = 0$, for all $d \in D$. We also define a balance function on any subset of directions: Given set D we define $b(D, d)$ to be the balance of direction d w.r.t. the directions from D, i.e. the defining coordinate is now $d_{max} \in D$.

Furthermore, we define the concept of *saturation.* This is with regards to history and the current vertex in the algorithm run. Given a set of directions $D \subseteq [\pm n]$ and a vertex v we say that v is D-saturated when for every $d \in D$ with $b(d) > 0$, the direction d is not available for v. It follows that if at vertex v set D is balanced, then v is D-saturated.

The construction. The construction is inductive. Let A_i denote the ith inductive step. The base case, A_0, is a 6-AUSO. Due to the lack of space we do not define it here, but we will mention and utilize some of its properties. A complete description of the base case can be found in the full version [25]. Every inductive step adds 6 new dimensions. As before the bundle of coordinates C_i is the one added at the ith step of induction (and C_0 are the ones of the base case). Also, with D_i we denote the directions that correspond to C_i and, similarly, for D_i^+. Thus, the AUSO A_i is $6(i+1)$-dimensional and the coordinates that describe it are in the set C_i^+. For each A_i, the starting vertex is $v_0^i = \{c_0^2, \dots, c_i^2\}$.

Let us now describe how to construct AUSO A_{i+1} from AUSO A_i. Let the new bundle of coordinates be C_{i+1}. We call IN the set of directions D_i^+ and OUT the set of directions D_{i+1}. At every inductive step the tie-breaking rule will be formed such that the directions from IN have priority over the ones from OUT. Thus, directions D_k have priority over the ones from $D_{k'}$, if $k < k'$. For simplicity, we write number $\pm k$ to mean direction $\pm c_{i+1}^k$.

The starting vertex for A_{i+1} is $v_0 = v_0^{i+1} = \{c_0^2, \dots, c_{i+1}^2\}$. Assume that P_i (the path the token takes in A_i) is known to us. Similarly to the previous sections, this can be interpreted as an adversary argument. To construct A_{i+1} we take $2^6 = 64$ copies of A_i and use three different 6-AUSO as connecting frames, utilizing Lemma 1. The crucial frames are given in Figure 3. For vertices that are not on P_i it does not matter which frame we use. For vertices that are on the path P_i we choose the connecting frame according to the following rule:

(1) Vertices that are not D_i^+-saturated (w.r.t. P_i) we connect with F_1.

(2) Vertices that are D_i^+-saturated (w.r.t. P_i) we connect with F_2.

(3) For the sink s_i of A_i we use F_3.

The latter is a 6-AUSO that has the same path $\boxed{1} \rightsquigarrow \boxed{12}$ as F_1, has its sink in $\boxed{12}$ (so the uppermost edge on coordinate c_{i+1}^1 is backward) and all other edges are forward (figure in [25]).

The result of this operation is A'_{i+1}. It remains to perform one reorientation. Namely, the balance-AUSO will be embedded in the face $\boxed{\text{B}}$, shown in Figure 3. The balance-AUSO is a uniform $6(i+1)$-AUSO which has its sink at the vertex v_0^i (all edges are oriented towards v_0^i). Formally, the outmap of vertex $v = v_0 \perp \boxed{\text{B}}$ is such that $s(v) \cap C_i^+ = \emptyset$. This reorientation will not introduce cycles; a formal proof can be found in the full version [25].

In reference to Figure 3, let us present the intuitive idea: The token will walk (in a projected way) along P_i once while walking between $\boxed{1}$ and $\boxed{12}$. Then, it will go back to

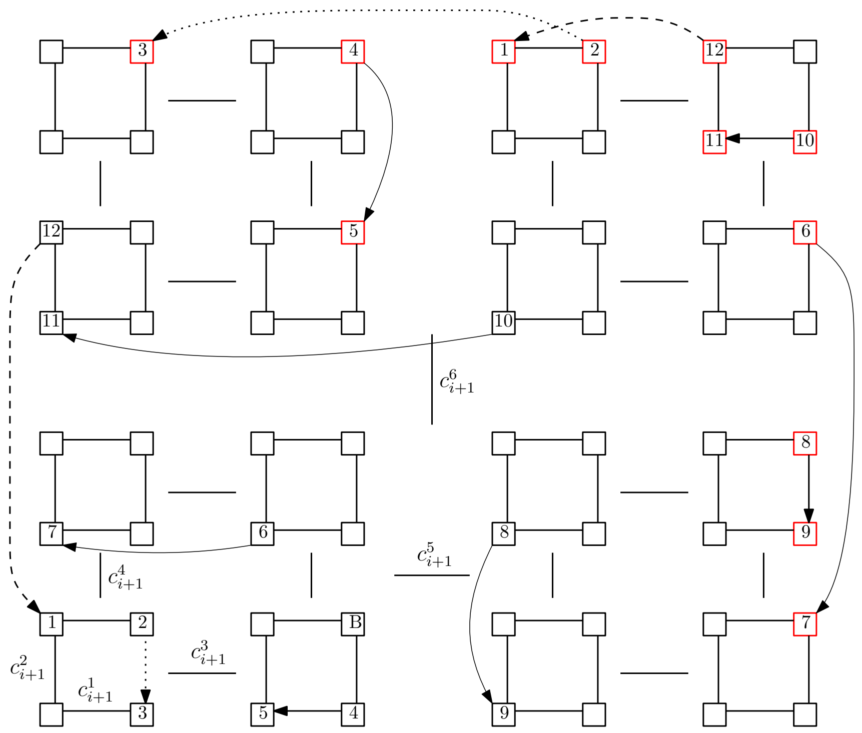

Figure 3 Both orientations F_1 and F_2 are given in this figure. For simplicity, only the orientations of the backward edges are explicitly drawn; every other edge is forward. The frame F_1 includes the *dashed* backward edges but not the dotted ones; F_2 includes the *dotted* backward edges but not the dashed ones. The solid backward edges are included in both frames.

the start of P_i in the balance-AUSO $\boxed{\text{B}}$. Then, it will walk the path P_i once again while walking between $\boxed{1}$ and $\boxed{12}$.

Below, we give two crucial properties that will hold for our construction. The first one is about the base case A_0 (of which a detailed description can be found in the full version [25]).

(i) There is at least one $(\pm[6])$-saturated vertex, other than the starting vertex v_0^0, in A_0. In addition, the sink s_0 is at least two vertices away from the last vertex on the path P_0 that was $(\pm[6])$-saturated.

Property (i) will be utilized in the proof of Lemma 9. The second property holds for every inductive step A_k of the construction $0 \leq k \leq i+1$.

(ii) When the token reaches the sink s_k of A_k there are exactly $4(k+1)$ negative coordinates imbalanced. Let $IM_k = \{-c_0^3, -c_0^4, -c_0^5, -c_0^6, \ldots, -c_k^3, -c_k^4, -c_k^5, -c_k^6\}$. For every $d \in IM_k$ we have $b(d) = 1$ and for every other d we have $b(d) = 0$.

Property (ii) holds for the base case (details in the full version [25]). Then, we will argue in the step-by-step analysis that it also holds for every inductive step of the construction. Also note that if the token takes the directions in IM_k from the sink s_k, it will go to the the starting vertex v_0^k. Such a path is not available in any of the connecting frames; however, a path which spans exactly those directions is available in the balance-AUSO.

We will now analyze the behavior of Zadeh's rule on AUSO A_{i+1}. Firstly, let us define the tie-breaking ordered list T_{i+1}:

$$T_{i+1} = T_i \cdot (+1, -2, +3, -1, +4, -3, +5, -4, +6, -5, +2, -6).$$

As before a number $\pm k$ indicates the direction $\pm c_0^k$. Secondly, we state two lemmas that will be used in the analysis that comes below. We include proofs for those in the full version [25].

▶ **Lemma 9.** *Let token t be at an IN-saturated vertex v, such that $s(v) \cap C_i^+ \neq \emptyset$. Then, $\exists d \in IN$ such that $v \xrightarrow{d} v'$ and v' also has $s(v') \cap C_i^+ \neq \emptyset$. Moreover, v' is not IN-saturated.*

▶ **Lemma 10.** *Let the token t be at a vertex v as in the lemma above. Then there is a vertex v' that comes after v on P_{i+1} and such that v' is IN-saturated.*

Step-by-step analysis. We are ready to give a description for the behavior of the algorithm on A_{i+1}, in as much detail as the space allows; a very careful analysis can be found in the full version [25]. Initially, the token is at the starting vertex v_0. Let us denote with d_{max}^{OUT} the direction that maximizes history over the OUT directions; similarly, we define d_{max}^{IN}. Assume that the token is at an IN-saturated vertex and $b(d_{max}^{OUT}) > 0$. Then, the algorithm will use directions from OUT until it reaches a vertex that is OUT-saturated.

When at $\boxed{1}$, directions from IN will be used, since they have priority in T_{i+1}. After some steps, an IN-saturated vertex v_s will be reached, by Lemma 10. At v_s, we have that $b(d_{max}^{OUT}) > 0$. The connecting frame will be F_2. The directions from OUT will be utilized and the token will take a path $\boxed{1} \rightsquigarrow \boxed{12}$, where it will reach $v_s \perp \boxed{12}$. Then, we have $b(-6) = 1$ and for every other direction $d \in OUT$, $b(d) = 0$; also, $b(d_{max}^{OUT}) = 0$. Because the frame is F_2, the dashed edge is not available: $v_s \perp \boxed{12} \leftarrow v_s \perp \boxed{1}$. Thus, $v_s \perp \boxed{12}$ is OUT-saturated. One direction from IN will be used; by Lemma 9 the next vertex is not IN-saturated. The frame will then be F_1, and direction -6 will be used. The token is back to $\boxed{1}$ and $b(OUT, d) = 0$, for every $d \in OUT$, $b(d_{max}^{OUT}) = 1$ and $b(d_{max}^{IN}) = 0$. The token will keep moving between $\boxed{1}$ and $\boxed{12}$ in the way thus described, until it reaches a vertex v_{s_i}, such that $s(v_{s_i}) \cap C_i^+ = \emptyset$ (v_{s_i} corresponds to the sink of A_i). The latter will be evaluated in $\boxed{1}$, by Lemma 9. Then, we have that $b(d) = 1$, for every $d \in IM_i$, by Property (ii) (which holds inductively).

The frame for v_{s_i} is F_3. The token will go over $+1$ and $+3$ to $\boxed{B}$. Therein, it will take a path $v_{s_i} \perp \boxed{B} \rightsquigarrow v_0 \perp \boxed{B}$ for which it will use exactly the directions from IM_i; afterwards, IN is balanced. The token will take a path $v_0 \perp \boxed{B} \rightsquigarrow v_0 \perp \boxed{8} \rightsquigarrow v_0 \perp \boxed{12}$. All the vertices on this path are IN-saturated because IN is balanced. The frame will be F_2 and, so, the dashed edge is not available: $v_0 \perp \boxed{12} \leftarrow v_0 \perp \boxed{1}$. At this point $b(-3) = b(-4) = b(-5) = b(-6) = 1$ and for every other $d \in OUT$, $b(d) = 0$. From now on, all the steps that the token will take using directions from IN will be consistent with the path P_i. This is because IN is balanced and the token is at a vertex that corresponds to v_0^i (the starting vertex of A_i).

The token will keep moving between $\boxed{1}$ and $\boxed{12}$, in a similar way as described above, until it reaches a vertex v_{s_i} such that $s(v_{s_i}) \cap C_i^+ = \emptyset$. The latter will be evaluated in $\boxed{1}$, by Lemma 9. The main difference to the situation of the previous paragraphs is in the balance vector. After the token reaches an IN-saturated vertex in $\boxed{1}$, the balance vector will be $b(OUT, -4) = b(OUT, -5) = b(OUT, -6) = 1$ and $b(d_{max}^{OUT}) = 1$. This is also the case when v_{s_i} is reached. But then, the connecting frame is F_3 and a path $v_{s_i} \perp \boxed{1} \rightsquigarrow v_{s_i} \perp \boxed{12}$ will be taken. The global sink will be exactly at $v_{s_i} \perp \boxed{12}$. After the sink has been reached, we have $b(d_{max}^{OUT}) = 0$ and, thus, $b(-3) = b(-4) = b(-5) = b(-6) = 1$. Thus, Property (ii) will also be satisfied for the new inductive step.

With the above analysis, we have proved that the path P_{i+1} will have length that is larger than twice the length of path P_i. Therefore, we obtain the recursion $T(n+6) > 2T(n)$ which leads to the proof of Theorem 8.

6 Conclusions

In this paper, we have constructed AUSO on which the three pivot rules of interest can take exponentially long paths. Several interesting problems remain open: First and foremost is settling if Zadeh's and Johnson's rules admit exponential lower bounds even on linear programs. Moreover, it remains open to decide if Zadeh's rule admits Hamiltonian paths on AUSO, a direction suggested by the authors of [1]. Finally, we are interested in exponential lower bounds for all the history-based rules that are discussed in [1]. We believe that our methods can be used to prove exponential lower bounds on AUSO for all of those rules.

References

1 Yoshikazu Aoshima, David Avis, Theresa Deering, Yoshitake Matsumoto, and Sonoko Moriyama. On the existence of Hamiltonian paths for history based pivot rules on acyclic unique sink orientations of hypercubes. *Discrete Applied Mathematics*, 160(15):2104–2115, 2012. doi:10.1016/j.dam.2012.05.023.

2 David Avis and Oliver Friedmann. An exponential lower bound for cunningham's rule. *Mathematical Programming*, pages 1–35, 2016. doi:10.1007/s10107-016-1008-4.

3 József Balogh and Robin Pemantle. The Klee-Minty random edge chain moves with linear speed. *Random Structures & Algorithms*, 30(4):464–483, 2007. doi:10.1002/rsa.20127.

4 William H Cunningham. Theoretical properties of the network simplex method. *Mathematics of Operations Research*, 4(2):196–208, 1979. doi:10.1287/moor.4.2.196.

5 Jan Foniok, Bernd Gärtner, Lorenz Klaus, and Markus Sprecher. Counting unique-sink orientations. *Discrete Applied Mathematics*, 163, Part 2:155–164, 2014. doi:10.1016/j.dam.2013.07.017.

6 Oliver Friedmann. A subexponential lower bound for Zadeh's pivoting rule for solving linear programs and games. In *IPCO 2011*, pages 192–206, 2011. doi:10.1007/978-3-642-20807-2_16.

7 Oliver Friedmann, Thomas Dueholm Hansen, and Uri Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *STOC 2011*, pages 283–292, 2011. doi:10.1145/1993636.1993675.

8 Bernd Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM J. Comput.*, 24(5):1018–1035, 1995. doi:10.1137/S0097539793250287.

9 Bernd Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures & Algorithms*, 20(3):353–381, 2002. doi:10.1002/rsa.10034.

10 Bernd Gärtner, Martin Henk, and Günter M. Ziegler. Randomized simplex algorithms on Klee-Minty cubes. *Combinatorica*, 18(3):349–372, 1998. doi:10.1007/PL00009827.

11 Bernd Gärtner and Antonis Thomas. The complexity of recognizing unique sink orientations. In *STACS 2015*, pages 341–353, 2015. doi:10.4230/LIPIcs.STACS.2015.341.

12 Bernd Gärtner and Antonis Thomas. The niceness of unique sink orientations. In *APPROX/RANDOM 2016*, pages 30:1–30:14, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.30.

13 Thomas Dueholm Hansen, Mike Paterson, and Uri Zwick. Improved upper bounds for random-edge and random-jump on abstract cubes. In *SODA 2014*, pages 874–881, 2014. doi:10.1137/1.9781611973402.65.

14 Thomas Dueholm Hansen and Uri Zwick. Random-edge is slower than random-facet on abstract cubes. In *ICALP 2016*, pages 51:1–51:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.51.

15 Gil Kalai. A subexponential randomized simplex algorithm (extended abstract). In *STOC 1992*, pages 475–482, 1992. doi:10.1145/129712.129759.

16 Victor Klee and George J. Minty. How good is the simplex algorithm? *Inequalities III*, pages 159–175, 1972.

17 Jiří Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures & Algorithms*, 5(4):591–607, 1994. doi:10.1002/rsa.3240050408.

18 Jiří Matoušek. The number of unique-sink orientations of the hypercube*. *Combinatorica*, 26(1):91–99, 2006. doi:10.1007/s00493-006-0007-0.

19 Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16(4/5):498–516, 1996. doi:10.1007/BF01940877.

20 Jiří Matoušek and Tibor Szabó. RANDOM EDGE can be exponential on abstract cubes. *Advances in Mathematics*, 204(1):262–277, 2006. doi:10.1016/j.aim.2005.05.021.

21 Ingo Schurr and Tibor Szabó. Finding the sink takes some time: An almost quadratic lower bound for finding the sink of unique sink oriented cubes. *Discrete & Computational Geometry*, 31(4):627–642, 2004. doi:10.1007/s00454-003-0813-8.

22 Ingo Schurr and Tibor Szabó. Jumping doesn't help in abstract cubes. In *IPCO 2005*, pages 225–235, 2005. doi:10.1007/11496915_17.

23 Alan Stickney and Layne Watson. Digraph models of Bard-type algorithms for the linear complementarity problem. *Math. Oper. Res.*, 3(4):322–333, 1978. doi:10.1287/moor.3.4.322.

24 Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *FOCS 2001*, pages 547–555, 2001. doi:10.1109/SFCS.2001.959931.

25 Antonis Thomas. Exponential lower bounds for history-based simplex pivot rules on abstract cubes. *CoRR*, abs/1706.09380, 2017. URL: https://arxiv.org/abs/1706.09380.

26 Norman Zadeh. What is the worst case behavior of the simplex algorithm. *Polyhedral computation*, 48:131–143, 2009.