Finding the Sink Takes Some Time

An Almost Quadratic Lower Bound for Finding the Sink of Unique Sink Oriented Cubes

Ingo Schurr[⋆] and Tibor Szabó [⋆⋆]

Theoretical Computer Science, ETH Zürich
CH-8092 Zürich, Switzerland
{schurr,szabo}@inf.ethz.ch

Abstract. We give a worst-case $\Omega(\frac{n^2}{\log n})$ lower bound on the number of vertex evaluations a deterministic algorithm needs to perform in order to find the (unique) sink of a unique sink oriented n-dimensional cube. We consider the problem in the vertex-oracle model, introduced in [17]. In this model one can access the orientation implicitly, in each vertex evaluation an oracle discloses the orientation of the edges incident to the queried vertex. An important feature of the model is that the access is indeed arbitrary, the algorithm does *not* have to proceed on a directed path in a simplex-like fashion, but could "jump around". Our result is the first super-linear lower bound on the problem. The strategy we describe works even for acyclic orientations. We also give improved lower bounds for small values of n and fast algorithms in a couple of important special classes of orientations to demonstrate the difficulty of the lower bound problem.

1 Introduction

Notation, Definitions. For our purposes a cube is the power set of a set. More precisely, for $A \subseteq B$ finite sets the cube $\mathfrak{C} = \mathfrak{C}^{[A,B]}$ is the edge labeled graph with vertex set $\mathrm{V}(\mathfrak{C}) := [A, B] := \{X \mid A \subseteq X \subseteq B\}$, edge set

$$\mathrm{E}(\mathfrak{C}) := \{\{v, v \oplus \{l\}\} \mid v \in \mathrm{V}(\mathfrak{C}), l \in B \setminus A\}$$

and edge labeling

$$\lambda(\{v, v \oplus \{l\}\}) := l \ ,$$

where the symbol $\oplus$ denotes the symmetric difference of two sets. We will say that an edge e is *l-labeled* if $\lambda(e) = l$, and for a subset L of the labels we say *L-labeled edges* for the set of all edges which are l-labeled with some $l \in L$.

⋆ Supported by the joint Berlin/Zürich graduate program Combinatorics, Geometry, and Computation (CGC), financed by German Science Foundation (DFG) and ETH Zurich.

⋆⋆ Supported by the joint Berlin/Zürich graduate program Combinatorics, Geometry, and Computation (CGC), financed by German Science Foundation (DFG) and ETH Zurich, and by NSF grant DMS 99-70270.

From a purely combinatorial point of view, the cube $\mathfrak{C}^{[A,B]}$ is sufficiently described by its *label set* $\operatorname{carr}\mathfrak{C}^{[A,B]} := B \setminus A$. Up to graph-isomorphism even $\dim \mathfrak{C}^{[A,B]} := |B \setminus A|$, the *dimension*, determines the cube. For most of the time we shall work with cubes $\mathfrak{C}^B := \mathfrak{C}^{[\emptyset,B]}$, i.e. power sets. In case $\operatorname{carr}\mathfrak{C}$ does not play any role we even abandon the superscript and write $\mathfrak{C}$.

The additional parameter A is helpful in naming the subcubes of a cube. A *subcube* of a cube $\mathfrak{C}^{[A,B]}$ is a cube $\mathfrak{C}^{[X,Y]}$ with $A \subseteq X \subseteq Y \subseteq B$. These subcubes correspond to the *faces* of a geometric realization of $\mathfrak{C}^{[A,B]}$.

Instead of writing down X and Y it is often more convenient to describe a subcube by a vertex and a set of labels generating it: we write $\mathfrak{C}(v, L)$ for the smallest subcube of $\mathfrak{C}$ containing v and $v \oplus L$. In fact for $X = v \cap \bar{L}$ and $Y = v \cup L$ we have $\mathfrak{C}(v, L) = \mathfrak{C}^{[X,Y]}$, where $\bar{L}$ denotes the complement of L with respect to $\operatorname{carr}\mathfrak{C}$.

Given an orientation of a cube (i.e. an orientation of its edges), a vertex is called a *sink* if all its incident edges are incoming. An orientation ψ of $\mathfrak{C}$ is called a *unique sink orientation* or USO, if ψ restricted to any subcube has a unique sink. We usually don't distinguish between a USO and the cube equipped with that USO, and refer to the cube as USO as well. An orientation is called *acyclic* if it contains no directed cycle. Acyclic unique sink orientations are abbreviated by AUSO.

The Problem. Easily stated: find the sink of a USO. Following [17] we assume that the orientation is given implicitly, i.e. we can access an *arbitrary* vertex of the USO through an oracle, which then reveals the orientation of the edges incident to the requested vertex. This basic operation is called *vertex evaluation* (sometimes we refer to it as *step* or *query*), and we are interested in evaluating the (unique) sink of a cube by as few vertex evaluations as possible. Formally, let $\operatorname{eval}(\mathbf{A}, \psi)$ be the number of vertex evaluation it takes for a deterministic algorithm $\mathbf{A}$ to evaluate the sink of a USO (or AUSO) ψ , and define $t(n)$ (or $t_{\text{acyc}}(n)$) to be $\min_{\mathbf{A}} \max_{\psi} \operatorname{eval}(\mathbf{A}, \psi)$. Obviously, $t(n) \geq t_{\text{acyc}}(n)$. Let $\tilde{t}(n)$ and $\tilde{t}_{\text{acyc}}$ be the corresponding functions for randomized algorithms; the expected number of evaluations a fastest randomized algorithm takes until it finds the sink of any USO (or AUSO).

Unique sink orientations provide a common framework for several seemingly different problems. Related abstractions were considered earlier, mainly to deal with geometric problems. LP-type problems, Abstract Objective Functions and Abstract Optimization Problems [3, 8–10, 12, 13] have a rich literature and provide the fastest known algorithms for several geometric problems in the *unit cost model.* As it is always the case with abstractions, it is very well possible that the model of USO is in fact too general for being of any use, but there are a number of results suggesting otherwise.

At first it is not even clear how to find the sink of a USO in $o(2^n)$ evaluations; whether unique sink orientations have enough structure, which distinguishes them from just any orientation of the cube. Indeed, in order to find the sink in an *arbitrary* orientation (with a sink), one needs at least $2^{n-1} + 1$ vertex evaluations [18].

In [17] a deterministic algorithm running in 1.61^n evaluations and randomized algorithm running in 1.44^n evaluations (using that $\tilde{t}(3) = \frac{4074633}{1369468}$ [15]) were given for evaluating the sink of a USO. These algorithms indicate that USOs have some, actually quite rich, structure. Another result, quantitatively pointing to this direction, is due to Matoušek [11]; he showed that the number of USOs is $2^{\Theta(\log n 2^n)}$, which is *significantly less* than $2^{n2^{n-1}}$, the number of all orientations. A bonus for optimists, that within the class of orientations providing the matching lower bound for Matoušek's upper bound the sink can be found in *five*(!) steps.

Motivation. The most obvious and actually quite widely investigated appearance of USOs is the special case of linear programming; i.e. when the polyhedron in question is a slanted geometric cube. Then a linear objective function canonically defines an AUSO on the cube and finding the sink of this orientation obviously corresponds to finding the minimum of the objective function on the polyhedron. The importance of this question is well-demonstrated by the number of papers investigating the running time of specific randomized simplex-like algorithms, like RandomEdge or RandomFacet, on (sometimes even specific) acyclic orientations. The RandomEdge algorithm proceeds on a directed path by choosing uniformly at random among the outgoing edges at each vertex. The RandomFacet algorithm chooses a random facet of the cube where the smaller dimensional algorithm is run and after finding the sink of the facet (and in case that sink is not the global sink) it proceeds to the antipodal facet to find its sink by another smaller dimensional RandomFacet algorithm. Gärtner [4, 5] showed that the expected number of evaluations it takes for the RandomFacet algorithm to evaluate the sink of an AUSO is at most $e^{2\sqrt{n}}$. Gärtner, Henk, and Ziegler [6] analyzed the behavior of RandomEdge and RandomFacet on a specific, particularly interesting orientation, the so-called Klee-Minty cubes.

USOs also appear in less obvious ways, where the correspondence between the problem and the orientation is more abstract. Stickney and Watson [16] defined a combinatorial correspondence between the solution of certain linear complementarity problems [2] (corresponding to so-called P-matrices) and finding the sink in an appropriate USO. In this correspondence the appearance of cycles in the USO is possible.

Similarly, certain quadratic optimization problems, like finding the smallest enclosing ball of $d+1$ affinely independent points in the Euclidean d-space can be reduced to finding the sink in an appropriate USO. For each such point-set, there is a corresponding USO, where the sink corresponds to the smallest enclosing ball of the point set [7]. Here again cycles can arise. The general problem of n points in d-space is known to be reducible to the case when n is small (i.e. around d^2) compared to d. Then one can place the ambient d-space into $\mathbb{R}^{n-1}$ and perturb the points for affine independence.

Note that every linear programm can be translated into a smallest enclosing ball problem. Via this detour unique sink orientations can be seen as an abstraction of linear programming in general (and not only for cube-like polyhedron).

Results. The main finding of our paper is a lower bound of order $\frac{n^2}{\log n}$ for the number of evaluations a deterministic algorithm needs in order to find the sink of an n-dimensional AUSO. Our result is the first non-linear lower bound for the function $t(n)$. On the way we organize our current knowledge about producing USOs. We also prove better lower bounds for small values of n. To motivate our results we look at a couple of simpler classes of orientations.

Lower bounds were found earlier for several related problems or special cases. Aldous [1] considered orientations given canonically by an ordering of the vertices of the cube, which have a unique sink (but not necessarily every subcube has a unique sink). These orientations are acyclic by definition. For them he proves a highly nontrivial exponential lower bound of $\sqrt{2}^{n(1+o(1))}$, using the hitting times of a random walk on the cube. He also provides a simple randomized algorithm which is essentially best possible. Since his model is in some sense weaker and in some sense stronger than our USOs, his results un(?)fortunately do not imply anything for our problem.

Other known lower bounds are related to specific randomized simplex-like algorithms, for example RandomEdge or RandomFacet. On AUSOs Matoušek [12] proved an $e^{\Omega(\sqrt{n})}$ lower bound for the expected running time of RandomFacet, which nicely complements the $e^{O(\sqrt{n})}$ upper bound of Gärtner [4, 5]. Gärtner, Henk and Ziegler [6] showed that the expected running time of RandomEdge on the Klee-Minty cubes is $\Omega(\frac{n^2}{\log n})$. A negative result of Morris [14] could also be interpreted as a lower bound for general USOs. He constructs USOs on which the expected running time of RandomEdge is $((n-1)/2)!$, more than the number of vertices.

Our paper is organized as follows. In Sect. 2 we collect notations, definitions and several known facts about USOs we will make use of. In Sect. 3 we establish our basic building blocks: two different ways of obtaining new USOs from old ones. In Lemma 3 we define a certain product construction of USOs, while in Lemma 4 we describe circumstances under which a local change in an existing USO produces another USO. In order to motivate our lower bound on the general problem, in Sect. 4 we discuss two important smaller classes of USOs for which very fast algorithms exist. Decomposable orientations are built recursively by taking two decomposable orientations of dimension $n-1$ on two disjoint facets of an n-cube and orient all edges between them the same direction. Wiliamson Hoke [19] gave a simplex-like algorithm on them in $O(n^2)$ time. In Prop. 1 we observe that our extra power of being able to "jump around" lets us find the sink in only $n+1$ evaluations. We also observe that this is best possible; i.e. on the class of decomposable orientations we know *a* fastest algorithm. In Prop. 2 we consider the class of the so-called *matching-flip* orientations. This class was introduced by Matoušek and Wagner to give a lower bound for the number of all USOs. Matoušek [11] proved that the number of all USOs is $2^{O(\log n 2^n)}$. The class of matching-flip orientations is in fact so rich that it provides a matching lower bound for the order of magnitude of the logarithm of the number of all USOs. Thus it is somewhat surprising that for this class there is an algorithm finding the sink in five steps.

In Sect. 5 we provide a strategy, which forces every deterministic algorithm to at least $\Omega(\frac{n^2}{\log n})$ evaluations while finding the unique sink of an n-dimensional cube with an acyclic unique sink orientation. We also give a matching lower bound for the 4-dimensional algorithm of [17].

In Sect. 6 we list several open problems.

2 Preliminaries

For an orientation ψ of a cube $\mathfrak{C}^{[A,B]}$ the *outmap* s_ψ *of* ψ is the map assigning to every vertex v the set of labels of outgoing edges, i.e. $s_\psi : \mathfrak{C} \to 2^{\operatorname{carr} \mathfrak{C}}$ with

$$s_\psi(v) = \{l \mid \{v, v \oplus \{l\}\} \text{ is outgoing from } v\} \quad .$$

Obviously, a vertex v_0 is a sink iff $s_\psi(v_0) = \emptyset$.

An important property of outmaps of USOs was proved in [17].

Lemma 1. *[17, Lemma 2.2] Let ψ be a USO. Then s_ψ is a bijection.*

Its proof relies on a simple observation, which is also a tool to produce new USOs from old ones. If ψ is an orientation of $\mathfrak{C}$ and $L \subseteq \operatorname{carr} \mathfrak{C}$, then let $\psi^{(L)}$ be the orientation of $\mathfrak{C}$ which agrees with ψ on $\bar{L}$-labeled edges and differ on L-labeled edges.

Lemma 2. *[17, Lemma 2.1] For any $L \subseteq \operatorname{carr} \mathfrak{C}$, if ψ is a USO, then $\psi^{(L)}$ is a USO as well.*

Let us remark here that acyclicity does not necessarily survives this "label-flip"; there are examples of AUSOs ψ and labels $l \in \operatorname{carr} \mathfrak{C}$ (already for $\dim \mathfrak{C} = 3$), such that $\psi^{(\{l\})}$ is *not* an AUSO.

In [17, Lemma 2.3] outmaps of USOs were characterized; $s : \mathrm{V}(\mathfrak{C}) \to 2^{\operatorname{carr} \mathfrak{C}}$ is the outmap of a unique sink orientation, iff

$$(s(v) \oplus s(w)) \cap (v \oplus w) \neq \emptyset \text{ for all } v \neq w \in \mathrm{V}(\mathfrak{C}).$$

We call such outmaps *unique sink outmaps* and – as the acronyms conveniently coincide – we abbreviate them by USO as well. This usually does not cause any confusion since orientations and outmaps determine each other uniquely.

Note that for cubes $\mathfrak{C}^B$ every outmap is a permutation of 2^B. An important example of a unique sink outmap on every $\mathfrak{C}^B$ is the identity. More generally, for every $w \in \mathrm{V}(\mathfrak{C})$ the map ι_w defined by $\iota_w(v) := v \oplus w$ (for any $v \in \mathrm{V}(\mathfrak{C})$) is a USO (note, that $\iota_\emptyset$ is the identity). The corresponding orientation directs every edge towards w. We refer to such orientations as *uniform orientations.* Note that the sink of ι_w is w and the source is $\bar{w} = B \setminus w$.

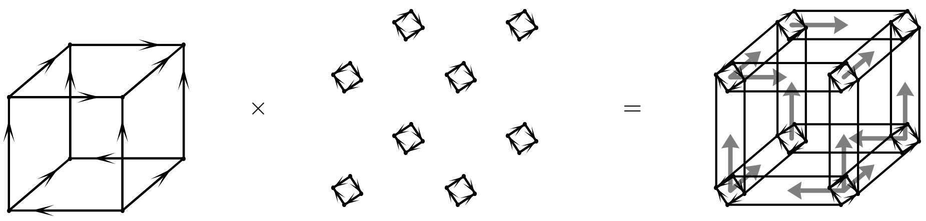

Fig. 1. A product of a three-dimensional USO with two-dimensional ones

3 Basic Constructions

Take a unique sink orientation and replace every vertex by a unique sink oriented cube of a fixed dimension. This construction defines a *product structure* for USOs. Figure 1 tries to illustrate that: In a three-dimensional unique sink orientation the vertices are replaced by two-dimensional unique sink orientations.

The statement of the next lemma provides the formal definition of this product construction, and shows that indeed it produces a unique sink orientation.

Lemma 3. *Let A be a set of labels, $B \subseteq A$ and $\bar{B} = A \setminus B$. For a USO $\tilde{s}$ on $\mathfrak{C}^B$ and for USOs s_u on $\mathfrak{C}^{\bar{B}}$, $u \in \mathrm{V}(\mathfrak{C}^B)$, the map s on $\mathfrak{C}^A$ defined by*

$$s(v) = \tilde{s}(v \cap B) \cup s_{v \cap B}(v \cap \bar{B})$$

is a USO.

Furthermore, if $\tilde{s}$ and all s_u are acyclic, then so is s.

Proof. See journal version.

For $|B| = 1$, Lemma 3 says that two $(n-1)$-dimensional unique sink orientations can be combined to an n-dimensional one by placing them in two disjoint facets and directing all edges in between in the same direction.

The other extreme case $|\bar{B}| = 1$ shows that if a cube contains two opposite facets with the same $(n-1)$-dimensional unique sink orientation, the edges between these facets can be directed arbitrarily.

It is easy to see that we can direct all edges of one label arbitrarily only if the unique sink orientations in the two facets not containing edges with this label are the same. On the other hand, by placing a unique sink orientation in one facet and the orientation which has all edges flipped in the other facet, we are forced to direct all edges in between in the same direction. Therefore Lemma 3 is best possible in some sense, one cannot expect to get a more general product construction without further exploring $\tilde{s}$ and/or some s_u.

A simple consequence of Lemma 3 is that in an AUSO sink and source could be placed anywhere, independently of each other. This fact is stated in the next corollary and will be utilized in the proof of our main result.

Corollary 1. *For any two distinct vertices $u, v \in \mathrm{V}(\mathfrak{C}^A)$, there is an acyclic unique sink outmap with sink u and source v.*

Proof. See journal version.

Another way to construct new unique sink orientations is by local modification. Given a n-dimensional unique sink orientation one can replace the orientation of a subcube under certain conditions. It is clear that the replacing orientation has to be a USO, but that does not suffice.

Lemma 4. *Let s be a unique sink outmap on a cube $\mathfrak{C}^A$ and $\mathfrak{C}_0$ be a subcube with label set $B \subseteq A$. If $s(v) \cap \bar{B} = \emptyset$ for all $v \in \mathrm{V}(\mathfrak{C}_0)$ and s_0 is a unique sink outmap on $\mathfrak{C}^B$, then the map $s' : 2^A \to 2^A$ defined by $s'(v) = s_0(v \cap B)$ for $v \in \mathrm{V}(\mathfrak{C}_0)$ and $s'(v) = s(v)$ otherwise is a unique sink outmap on $\mathfrak{C}^A$.*

If s and s_0 are acyclic, then s' is acyclic as well.

For example in Fig. 2 the lower three-dimensional subcube can be directed arbitrarily, since all edges incident to it are incoming.

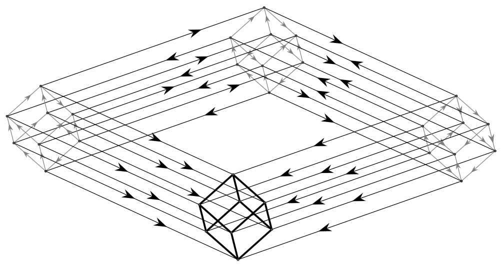

Fig. 2. Three-dimensional flippable subcube

Proof. See journal version.

Corollary 2. *Let s be a unique sink outmap on a cube $\mathfrak{C}^A$ and $\mathfrak{C}_0$ be a subcube with label set $B \subseteq A$. Suppose that $s(v) \cap \bar{B} = s(v') \cap \bar{B}$ for all $v, v' \in \mathrm{V}(\mathfrak{C}_0)$. If s_0 is a unique sink outmap on $\mathfrak{C}^B$, then the map $s' : 2^A \to 2^A$ defined by $s'(v) = s_0(v \cap B)$ for $v \in \mathrm{V}(\mathfrak{C}_0)$ and $s'(v) = s(v)$ otherwise is a unique sink outmap on $\mathfrak{C}^A$.*

Proof. See journal version.

Let us remark that unlike in Lemma 4, here acyclicity does not necessarily carry over to s'.

Again, the special case $|B| = 1$ is of some interest. In this scenario $\mathfrak{C}_0$ is a single edge and by the preceding corollary, we see that an edge can be flipped, if the outmaps of the two adjacent vertices only differ in the label of the edge. Since flipping an edge solely affects the outmap of the adjacent vertices the converse also holds. Therefore an edge $\{v, w\}$ is flippable iff $s(v) \oplus s(w) = v \oplus w$.

In particular, if ι_w is the outmap of a uniform orientation, then one can flip the edges of an arbitrary matching of the cube and the result is still a USO. As we shall see in the next section this construction is particularly interesting.

4 Interlude: Two Simple Classes

Let us recall that in the special case $|B| = 1$ of Lemma 3 we obtained an n-dimensional USO out of two (possibly distinct) $(n-1)$-dimensional USOs. The new orientation is special in the sense, that all edges of one label point in the same direction. This leads us to the notion of combed orientations; an orientation is called *combed*, if there is a label l for which all l-labeled edges are oriented towards the same facet.

Iterating the construction we get the class of *decomposable* unique sink orientations. Let $\mathcal{D}_1$ be the class of all 1-dimensional unique sink orientations and denote by $\mathcal{D}_{n+1}$ the class of all orientations constructed from two orientations from $\mathcal{D}_n$ using Lemma 3. We call $\mathcal{D}_n$ the class of decomposable unique sink orientations in dimension n. Equivalently, an orientation is decomposable iff every nonzero-dimensional subcube is combed.

In general it is not easy to check, whether a unique sink orientation is decomposable or even combed (one has to to evaluate half of the vertices), but in the class of decomposable unique sink orientations it is easy to find the sink.

Proposition 1. *For a decomposable USO of dimension n one needs at most $n+1$ vertex evaluations to evaluate the sink. Moreover, $n+1$ is best possible; i.e. for every deterministic algorithm there is a decomposable USO on which the algorithm needs at least $n+1$ evaluations.*

Proof. See journal version.

Our second example in this section arises from Lemma 2. Consider a uniform orientation and a matching of the cube. According to Lemma 2 every edge in our matching is flippable. Since flipping one of them does not affect the flippability of other edges of the matching, we can flip any number of them simultaneously.

In consequence, for every matching and every uniform orientation we obtain different unique sink orientations. These "matching-flip" USOs were first constructed by Matoušek and Wagner. By counting perfect matchings of the hypercube one obtains a very good lower bound for the number of unique sink orientations, which has the same order of magnitude in the logarithm as the number of all USOs.

Thus it is somewhat surprising (and encouraging for the general problem) that it is very easy to find the sink of an orientation from this very large class.

Proposition 2. *For a unique sink orientation coming from a uniform orientation by flipping a matching one needs at most 5 steps to find the sink. The value 5 here is best possible.*

Proof. See journal version.

5 The Strategy

In this section we prove the main result of the paper. In order to show lower bounds on $t(n)$ we consider ourselves the oracle playing against a deterministic algorithm, we call Al. We try to make sure that our answers to Al's evaluation requests (i) force Al to evaluate many vertices before the sink, and (ii) could be extended to an AUSO of the whole cube.

Given a sink-finding algorithm Al, we construct an acyclic unique sink orientation for which Al needs an almost-quadratic number of queries. For the first $n - \lceil \log_2 n \rceil$ inquiries we maintain a partial outmap $s : W \to \operatorname{carr} \mathfrak{C}^{[n]}$ on the set $W \subseteq \mathrm{V}(\mathfrak{C}^{[n]})$ of queried vertices, containing the answers we gave Al so far. We also maintain a set L of labels and an acyclic unique sink outmap $\tilde{s}$ on $\mathfrak{C}^L$. This smaller dimensional outmap $\tilde{s}$ is our "building block" which enables us to extend our answers to a global USO at any time.

Before the first inquiry we set $L = W = \emptyset$. After each inquiry we answer to Al by revealing the value of the outmap s at the requested vertex. Then we update L and $\tilde{s}$ such, that the following conditions hold.

(a) $|L| \leq |W|$,
(b) $w' \cap L \neq w'' \cap L$ for every $w' \neq w'' \in W$ and
(c) $s(w) = \tilde{s}(w \cap L) \cup \bar{L}$ for every $w \in W$.

Informally, condition (b) means that the projections of the queried vertices to $\mathfrak{C}^L$ are all distinct. This we shall achieve by occasionally adding a label to L, if the condition would be violated. Condition (c) is hiding two properties of our answers to the inquiries of Al. First that our answers are consistent with $\tilde{s}$ on L-labeled edges, and then that all $\bar{L}$-labeled edges, i.e. the edges leaving $\mathfrak{C}(w, L)$, are outgoing.

Suppose now that Al requests the evaluation of the next vertex u. We can assume that $u \notin W$. Depending on whether there is a $w \in W$ with $u \cap L = w \cap L$ or not we have to distinguish two cases.

Case 1. For every $w \in W$ we have $w \cap L \neq u \cap L$.

Then we answer $s(u) = \tilde{s}(u \cap L) \cup \bar{L}$ and leave L and $\tilde{s}$ unchanged. $(a) - (c)$ all hold trivially by the definition of the updates and the assumption of Case 1.

Case 2. There is a $v \in W$, such that $u \cap L = v \cap L$.

Let us immediately note that by condition (b), there is exactly one such $v \in W$. The assumption of Case 2 and $u \neq v$ implies that we can fix a label $l \in \bar{L}$ such that $l \in u \oplus v$.

We answer $s(u) = s(v) \setminus \{l\}$. Note that as $l \notin L$, $l \in s(v)$.

Now we have to update L and $\tilde{s}$. Our new L we get by adding l. To define the new orientation $\tilde{s}$ we take two copies of the old $\tilde{s}$ on the two facets determined by l. Then by Lemma 3, we can define the orientation of the edges going across arbitrarily, so we make them such that the condition (c) is satisfied. More formally, let

$$\tilde{s}(z) = \begin{cases} \tilde{s}(v \cap L) & \text{if } z = u \cap (L \cup \{l\}) \\ \tilde{s}(w \cap L) \cup \{l\} & \text{if } z = w \cap (L \cup \{l\}) \\ & \text{for some } w \in W,\ w \neq u \\ \tilde{s}(z) \cup (z \cap \{l\}) & \text{otherwise.} \end{cases}$$

Condition (a) still holds, because we added one element to each of W and L. Since L just got larger, we only need to check condition (b) for the pairs of vertices containing u. By the uniqueness of v, it is actually enough to check (b) for the pair u, v. Since l was chosen from $u \oplus v$ and now is included in L, $u \cap L \neq v \cap L$. Condition (c) is straightforward from the definitions.

We proceed until $|W| = n - \lceil \log_2 n \rceil$, and then change the strategy. By condition (a), $|L| \leq n - \lceil \log_2 n \rceil$. We choose an arbitrary superset $L' \supseteq L$ of L such that $|L'| = n - \lceil \log_2 n \rceil$. The set of labels $\bar{L}'$ determine at least $2^{|\bar{L}'|} \geq n$ disjoint subcubes generated by L'. As $|W| < n$, we can select one of them, say $\mathfrak{C}_0$, which does not contain any point evaluated so far.

Our plan is to apply Lemma 4 with $\mathfrak{C}_0$ and an orientation s, which is consistent with the outmaps of the vertices evaluated so far. The lemma then will enable us to reveal s on $\mathrm{V}(\mathfrak{C}^A) \setminus \mathrm{V}(\mathfrak{C}_0)$ to Al and still be able to start a completely "new game" on a cube of relatively large dimension. To construct s satisfying the conditions of Lemma 4 we use Lemma 3 twice.

First we define an orientation $\bar{s}$ of $\mathfrak{C}^{L'}$ using Lemma 3 with $A = L', B = L, \tilde{s}$ as defined above, and outmaps s_u on $\mathfrak{C}^{L' \setminus L}$ with the property that for every $w \in W$ the map $s_{w \cap L}$ has its source at $w \cap (L' \setminus L)$. This last requirement can be fulfilled because of condition (b).

Thus the resulting outmap $\bar{s}$ is consistent with the evaluated vertices in the sense that $s(w) \cap L' = \bar{s}(w \cap L')$ for each $w \in W$.

Next we apply Lemma 3 again with L', so we have to construct USOs s_u of $\mathfrak{C}^{\bar{L}'}$ for every $u \in \mathrm{V}(\mathfrak{C}^{L'})$. In doing so, we only take care that the sinks of all these orientations are in $\mathfrak{C}_0$, and if $u = w \cap L'$ for some $w \in W$ then $w \cap \bar{L}'$ is the source of s_u. (By condition (b) there can be at most one such vertex w for each u.) The appropriate s_u is constructed in Corollary 1. Now Lemma 3, applied with L', $\bar{s}$ and these s_u's, provides us with an orientation s which agrees with our answers given for the evaluated vertices, and all $\bar{L}'$-labeled edges are incoming into $\mathfrak{C}_0$.

By Lemma 4 we can reveal s on $\mathrm{V}(\mathfrak{C}) \setminus \mathrm{V}(\mathfrak{C}_0)$ to Al, and still be able to place *any* orientation on $\mathfrak{C}_0$, a subcube of dimension $n - \lceil \log_2 n \rceil$. Therefore we just proved

$$t_{acyc}(n) \geq n - \lceil \log_2 n \rceil + t_{acyc}(n - \lceil \log_2 n \rceil) \ .$$

Theorem 1. *A deterministic algorithm needs $\Omega(\frac{n^2}{\log n})$ many steps to find the sink of an acyclic unique sink orientation on a n-dimensional cube.*

Proof. See journal version.

For small dimensions $t(n)$ is easy to check; $t(0) = 1$, $t(1) = 2$, $t(2) = 3$ and $t(3) = 5$. In the following proposition we give a matching lower bound for the SevenStepsToHeaven algorithm of [17] in dimension 4. We note that with similar methods a lower bound of $2n-1$ could be given in any dimension, which is better for small values than our asymptotic lower bound from Theorem 1.

Proposition 3. $t(4) = 7$.

Proof. See journal version.

6 Comments and Open Problems

Of course in an ideal world one could hope to determine the functions $t(n)$, $\tilde{t}(n)$, $t_{acyc}(n)$ and $\tilde{t}_{acyc}(n)$ exactly. There are more realistic goals, though, for the nearer future. A natural question concerning the strategy in our paper is whether the AUSOs we use (i.e. the orientations arising from Lemma 3 and Lemma 4) can be realized geometrically, i.e. as a linear program. A positive answer would be of great interest.

The lone lower bound we are able to give for $\tilde{t}(n)$ is a mere $n/2$, which works even on the set of decomposable orientations. Any nonlinear lower bound would be interesting. An extension of our method for the deterministic lower bound seems plausible.

There is ample space for improvement on the upper bounds as well. The fastest known deterministic algorithm does work for general USOs; a first goal would be to exploit acyclicity in order to separate $t(n)$ and $t_{acyc}(n)$ from each other.

The *only* known nontrivial randomized algorithms for the general USO problem work in a product-like fashion. An improvement of $\tilde{t}(n)$ for a small value of n implies better algorithms for large n. Unfortunately the determination of $\tilde{t}(n)$ becomes unmanageable relatively soon; already the determination of $\tilde{t}(3)$ by Rote [15] presented a significant computational difficulty, and $\tilde{t}(4)$ is still not known. It would be really desirable to create nontrivial randomized algorithms working by an idea *different* from the Product Algorithm of [17]. A possible approach would be to find a way to randomize the deterministic FibonacciSeesaw algorithm [17].

RandomEdge and RandomFacet are very natural randomized algorithms, their analysis is extremely inviting but might be hard. The behavior of RandomFacet is well-understood for AUSO, for general USOs nothing is known. By the construction of Morris [14] RandomEdge is not a good choice to find the sink of a general USO, but for AUSOs not much is known about its behavior.

Acknowledgment: We are grateful to Alan Frieze, Jirka Matoušek, Uli Wagner and Emo Welzl for useful comments and conversations. We also would like to thank Emo Welzl for suggesting the algorithm for decomposable orientations in Prop. 1.

References

1. David Aldous. Minimization algorithms and random walk on the d-cube. *The Annals of Probability*, 11:403–413, 1983.
2. Richard W. Cottle, Jong-Shi Pang, and Richrad E. Stone. *The Linear Complementary Problem.* Academic Press, 1992.
3. Bernd Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM J. Comput.*, 24:1018–1035, 1995.
4. Bernd Gärtner. Combinatorial linear programming: Geometry can help. In *Proc. 2nd Int. Workshop on Randomization and Approximation Techniques in Computer Science*, volume 1518, pages 82–96, 1998. Lecture Notes in Computer Science.
5. Bernd Gärtner. The random-facet simplex algorithm on combinatorial cubes. submitted, 2001.
6. Bernd Gärtner, Martin Henk, and Günter M. Ziegler. Randomized simplex algorithms on Klee-Minty cubes. *Combinatorica*, 18(3):349–372, 1998.
7. Bernd Gärtner, Hiroyuki Miyazawa, Lutz Kettner, and Emo Welzl. From geometric optimization problems to unique sink orientations of cubes. manuscript in preparation, 2001.
8. Bernd Gärtner and Emo Welzl. Linear programming – randomization and abstract frameworks. In *Proc. 13th Ann. ACM Symp. Theoretical Aspects of Computer Science*, volume 1046 of *Lecture Notes Comput. Sci.*, pages 669–687. Springer-Verlag, 1996.
9. Bernd Gärtner and Emo Welzl. Explicit and implicit enforcing – randomized optimization. In *Lectures of the Graduate Program Computational Discrete Mathematics*, volume 2122 of *Lecture Notes in Computer Science*, pages 26–49. Springer-Verlag, 2001.
10. Gil Kalai. Linear programming, the simplex algorithm and simple polytopes. *Math. Programming*, 79:217–233, 1997.
11. Jiří Matoušek. manuscript.
12. Jiří Matoušek. Lower bounds for a subexponential optimization algorithm. *RSA: Random Structures & Algorithms*, 5(4):591–607, 1994.
13. Jiří Matoušek, Micha Sharir, and Emo Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.
14. Walter D. Morris. Randomized principal pivot algorithms for P-matrix linear complementarity problems. to appear, 2001.
15. Günter Rote. personal communication.
16. Alan Stickney and Layne Watson. Digraph models of bard-type algorithms for the linear complementary problem. *Mathematics of Operations Research*, 3:322–333, 1978.
17. Tibor Szabó and Emo Welzl. Unique sink orientations of cubes. In *Proc.* 42^{nd} *IEEE Symp. on Foundations of Comput. Sci.*, pages 547–555, 2000.
18. Emo Welzl. personal communication.
19. Kathy Williamson Hoke. Completely unimodal numberings of a simple polytope. *Discrete Appl. Math.*, 20:69–81, 1988.