	Hypercube problems	
	Lecture 19	
	January 3, 2019	
Lecturer: Petr Gregor	Scribe by: Viktor Němeček	Updated: October 2, 2019

1 Unique sink orientations

In this lecture we will have a look at some aspects of unique sink orientations and some time bounds for algorithms for finding the sink.

Definition 1 *A orientation of edges of* Q_n *is a* unique sink orientation (USO)*, if every subcube has a unique sink (a vertex of outdegree* 0*). If it moreover contains no directed cycle, it is an* acyclic unique sink orientation (AUSO)*.*

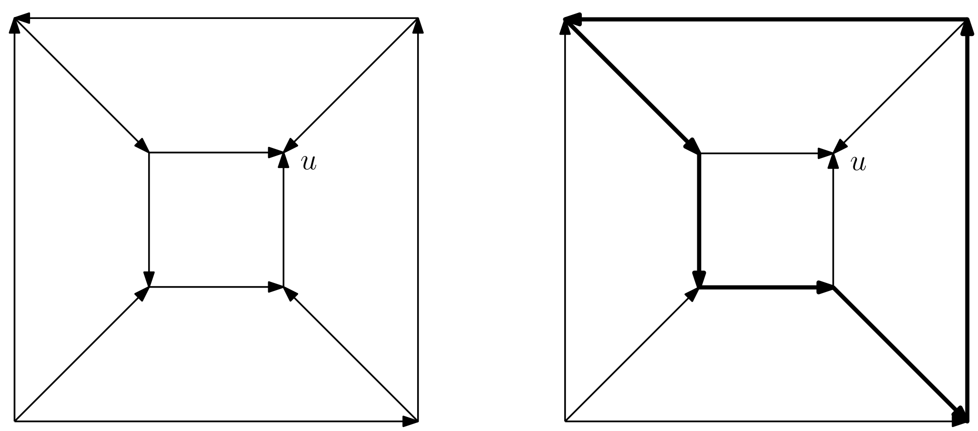

Figure 1: An AUSO (left) and a USO (right) with a 6-cycle in Q_3. In both cases, there is a global sink in the vertex denoted by u.

1.1 Motivation

- Linear programming on a *slanted geometric cube* (a polytope with the combinatorial structure of a cube), linear objective function (in general position) defines an AUSO. AUSOs for general convex polytopes are also known as abstract objective functions, or completely unimodal numberings. Example: Klee-Minty cube.
- Certain *linear complementarity problems* (defined by so-called P-matrices) define a USO.
- Certain quadratic optimization problems define a USO. One such example is finding the smallest enclosing ball of n affinely independent points in $\mathbb{R}^{n-1}$. Here, the vertices

of cube correspond to subsets of points and an edge is oriented from the set $A \cup \{x\}$ to the set A if and only if $x \in \beta(A)$ where $\beta(A)$ is the smallest ball enclosing A.

- Every linear program with n variables and m constraints can be translated into an USO in $Q_{2(n+m)}$ via certain convex program.

1.2 Properties

For an orientation φ of Q_n and $A \subseteq [n]$ let $\varphi^{(A)}$ be the orientation obtained from φ by flipping all edges of directions in A.

Lemma 2 *If φ is a USO in Q_n and $A \subseteq [n]$, then $\varphi^{(A)}$ is a USO as well.*

Proof Let us suppose $|A| = 1$ (if it is larger, we can continue recursively). Let s be the sink of φ and let s' be the sink of the other subcube on directions $\overline{A}$. Then the edge from s' in the direction from A must be outgoing (or both s and s' would be sinks). Thus $\varphi^{(A)}$ has sink s'. Furthermore, if s' was not a unique sink, then at least one of the subcubes on directions $\overline{A}$ had at least two sinks. We can prove that each subcube has unique sink with respect to $\varphi^{(A)}$ in the same way. ■

This lemma does not hold for AUSOs already for Q_3. For an orientation φ of Q_n the *outmap* $S_\varphi : V(Q_n) \to V(Q_n)$ of φ is defined by

$$S_\varphi(v) = \{i \in [n] \mid \text{the edge of direction } i \text{ from } v \text{ is outgoing in } \varphi\}.$$

Lemma 3 *If φ is a USO in Q_n, then S_φ is a bijection.*

Proof Suppose $u \neq v$, but $s_\varphi(u) = s_\varphi(v) = A$ for some $u, v \in V(Q_n)$. Then both u and v are sinks with respect to $\varphi^{(A)}$, contrary to Lemma 2. Thus s_φ is injective, and consequently bijective. ■

Corollary 4 *For a mapping $S : V(Q_n) \to V(Q_n)$ the following is equivalent:*

1. *S is the outmap of some USO.*
2. *$S \cap B$ is bijective on any subcube on any directions $B \subseteq [n]$.*
3. *$(S(u) \oplus S(v)) \cap (u \oplus v)$[1] is nonempty for any two $u, v \in V(Q_n)$.*

For an orientation φ of Q_n such that every subcube on directions $A \subseteq [n]$ has a unique sink we define the *A-inherited outmap* $S_{\varphi/A} : V(Q_{\overline{A}}) \to V(Q_{\overline{A}})$ *of* φ by

$$S_{\varphi/A}(v) = S_\varphi(u) \cap \overline{A}$$

where u is the sink in the Q_A-fiber of v.

Corollary 5 *If φ is a USO in Q_n and $A \subseteq [n]$, then $S_{\varphi/A}$ is the outmap of some USO of $Q_{\overline{A}}$.*

[1]In this notation, $u \oplus v$ denotes the set of directions in which u and v differ.

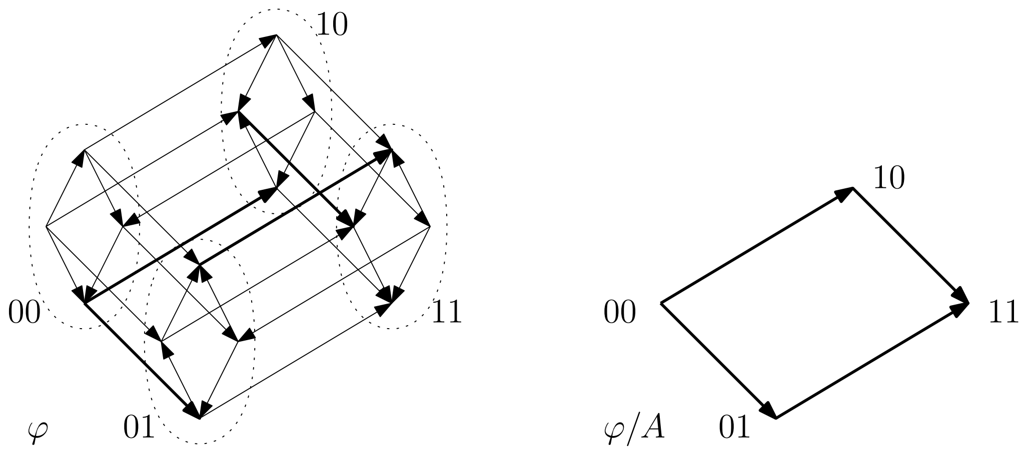

Figure 2: A USO φ in Q_4. If we denote A to be the directions within the little diamonds, then edges outgoing from the local sinks in each of those diamonds (bold in the picture) induce a USO corresponding to the A-inherited outmap $S_{\varphi/A}$.

1.3 Algorithms for finding the sink of a USO

We assume that a USO φ of Q_n is given by an *oracle* that reveals $S_\varphi(v)$ for a query $v \in V(Q_n)$. For an algorithm Alg let eval(Alg, φ) be the number of queries needed by Alg to find the sink of USO φ (including the mandatory query at the sink). We consider

$$t(n) = \min_{\text{det. Alg}} \max_{\text{USO } \varphi \text{ of } Q_n} \text{eval}(\text{Alg}, \varphi)$$

and similarly defined $t_{acyc}(n)$ for AUSOs. Exact values of $t(n)$ are known for $n \leq 4$:

n	0	1	2	3	4
$t(n)$	1	2	3	5	7

A trivial upper bound for $t(n)$ is $2^{n-1} + 1$. This bound is given by an algorithm, which queries each vertex in one bipartite class of Q_n, thus revealing the entire USO, and then queries the sink in case it already had not.

Lemma 6 *For every* $0 \leq k \leq n$, *it holds* $t(n) \leq t(k) \cdot t(n-k)$.

Proof Choose an arbitrary $A \subseteq [n]$ with $|A| = n - k$. Perform a search for the sink of $S_{\varphi/A}$ in $Q_{\overline{A}}$ in at most $t(k)$ queries. Each query to $S_{\varphi/A}$ needs to find (and query) a sink of some Q_A-fiber in at most $t(k-n)$ queries to S_φ. ■

Since $t(n) \leq t(k)^{\lceil n/k \rceil}$ for every $0 \leq k \leq n$ by Lemma 6 and $t(4) = 7$, we obtain the following.

Corollary 7 $t(n) = \mathcal{O}(\sqrt[4]{7}^n) = \mathcal{O}(1.63^n)$.

We will note here without proof that the best currently known upper bound is $t(n) = \mathcal{O}(1.61^n)$.

1.3.1 Constructions of (A)USO

A canonical AUSO of Q_n is a AUSO with the outmap $S(u) = u \oplus A$ for some fixed $A \subseteq [n]$.

Lemma 8 *Let S be (the outmap of) a USO in Q_A, $A \subseteq [n]$, and for $u \in V(Q_A)$ let S_u be (the outmap of) a USO in $Q_{\overline{A}}$ where $\overline{A} = [n] \setminus A$. Then $S' : V(Q_n) \to V(Q_n)$ defined as*

$$S'(v) = S(v \cap A) \cup S_{v \cap A}(v \cap \overline{A})$$

is (the outmap of) a USO in Q_n. Furthermore, if S and all S_u's are acyclic, then so is S'.

Proof Any subcube on directions $B \subseteq A$ or $B \subseteq \overline{A}$ has a unique sink since it is entirely in a Q_A-fiber, in which $S' = S$, respectively in a $Q_{\overline{A}}$-fiber of some u, in which $S' = S_u$. Otherwise, any subcube on directions B has a sink v such that $S'(v) \cap B = \emptyset$, in other words $v \cap (A \cap B)$ is a sink with respect to S and $v \cap (\overline{A} \cap B)$ is a sink with respect to $S_{v \cap \overline{A}}$, both determined uniquely.

Any cycle with respect to S' projects into a closed walk in Q_A. If it is nontrivial, S' has to be cyclic. If the projection is a single vertex u, then it is entirely within one $Q_{\overline{A}}$-fiber, and S_u in this fiber is thus cyclic. ■

Now let us point out two special cases. When $|A| = 1$, we have two different $(n-1)$-dimensional USOs with all edges in between in the same orientation. This gives us a class of decomposable USOs (by recursive application). On the other hand, when $|\overline{A}| = 1$ we have two copies of the same $(n-1)$-dimensional USO with the edges in between oriented arbitrarily.

Corollary 9 *For any vertices $u \neq v$ there is an AUSO with the sink u and the source v.*

Proof Take two canonical AUSOs in Q_{n-1}, one of them with a sink in u and the other with a source in v, with all the edges in between oriented from the latter to the former. ■

Lemma 10 *Let S be a USO in Q_n such that $S(v) \cap \overline{A} = \emptyset$ (e.g., v is the sink of some subcube on directions $\overline{A}$) for all vertices v in a subcube C on directions $A \subseteq [n]$. Then replacing S on C with another USO S_C of C gives a USO S'. Furthermore, if S and S_C are both acyclic, then so is S'.*

Proof Only subcubes intersecting with C are affected. Since all edges to C are incoming, the sink must be in the intersection where it is determined by S_C. No cycle can leave C, so S' is acyclic if S and S_C are. ■

Corollary 11 *Let S be a USO in Q_n such that $S(v) \cap \overline{A} = S(u) \cap \overline{A}$ for all vertices u, v in some subcube C on directions $A \subseteq [n]$. Then replacing S with another USO S_C of C produces a USO S'.*

Proof This follows from Lemmas 10 and 2. ■

Let us note that this corollary does not hold for AUSOs, as Lemma 2 does not hold for AUSOs. Once again, we will mention one notable special case, which is a matching flipping. Let M be a (not necessarily perfect) matching in Q_n and S_M be obtained from some canonical USO by flipping edges in M. Then S_M is a USO. This also sets a lower bound on the number of all possible (labeled) USOs as follows

$$n^{\Omega(2^n)} = \left(\frac{n}{e}\right)^{2^n} \leq \text{p. m. in } Q_n \leq \#\text{USOs (with multiplicity due to isomorphism)} = n^{\mathcal{O}(2^n)}.$$

1.3.2 Algorithm for decomposable USOs

The following algorithm takes at most $n+1$ queries, and this is optimal.

```
Choose arbitrary v₀; i ← 0;
while S(vᵢ)≠ ∅
    vᵢ₊₁ ← vᵢ ⊕ S(vᵢ); i++
return vᵢ
```

Claim 12 *For every i, the sink u and v_i are in the same $(n-i)$-subcube.*

Proof We always cross the decomposing direction towards the sink (and a decomposing direction is never crossed away from the sink). ■

Claim 13 *For every deterministic algorithm, there is a decomposable USO that requires at least $n+1$ queries.*

Proof Strategy for an oracle: Answer $S(u_0) = [n]$ (e.g. v_0 is the source) for the first query u_0. For the second query u_1, pick a decomposing direction $\ell \in u_0 \oplus u_1$ and return $[n] \setminus \{\ell\}$. Continue similarly. ■

1.3.3 Algorithm for matching flipped USOs

The following algorithm needs always at most 5 steps. We will note here without proof that the algorithm is also optimal. In the algorithm, we will of course stop if we find the sink before the end.

```
Choose an arbitrary v₁
v₂ ← v₁ ⊕ S(v₁)
v₃ ← v₂ ⊕ S(v₂)
if |S(v₂)|=1
    v₄ ← v₃ ⊕ S(v₃)
    evaluate v₄
else
    v₄ ← v₃ ⊕ (S(v₃) ∩ S(v₂))
    v₅ ← v₄ ⊕ S(v₄)
    evaluate v₅
```

Claim 14 *This strategy succeeds for any USO obtained from a canonical USO with a sink u by flipping some matching.*

Proof The vertex v_2 has distance at most one from u, so $|S(v_2)| \leq 2$. If v_2 was u (and not the sink), v_3 is the sink. Now v_2 is not u. If $|S(v_2)| = 1$, v_3 is u, and v_3 or v_4 is the sink. If $|S(v_2)| = 2$, v_3 is a neighbor of u, and either v_3 is the sink, or v_4 is u and v_5 the sink. ■

1.3.4 Lower bound

Now we propose a lower bound for deterministic algorithms that finds sinks in AUSOs (stronger than a lower bound for USO).

Lemma 15 *The following inequality holds for every $n \geq 2$,*

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

Proof We play as an oracle against a deterministic algorithm. Our strategy for the first $n - \lceil \log_2 n \rceil$ queries: maintain $A \subseteq [n]$ with $|A| \leq$ #queries so far and an AUSO S on Q_A that can be used in Lemma 8 to produce AUSO S' consistent with our answers. Furthermore, we make sure that all queries so far project to distinct vertices in Q_A.

For a query u we return $S'(u) = (u \cap A) \cup \overline{A}$ (e.g. a source with respect to $\overline{A}$). If a query would project to a vertex, to which another query already projected, we increase A by a separating direction and update S applying Lemma 8.

After $n - \lceil \log_2 n \rceil$ steps we can find a Q_A-fiber where we have a freedom to play the same strategy (any AUSO can be taken here by Lemma 10; a sink lies here because for all queries we responded with as many outgoing edges as possible).

Indeed, since $|\overline{A}| \geq \lceil \log_2 n \rceil$, there is some Q_A-fiber of u with no query.

Why we have the freedom? Each $Q_{\overline{A}}$-fiber contains at most 1 query, u can be sink in S_v of $Q_{\overline{A}}$-fiber at v by Corollary 9, S and S_v can be used to produce consistent S' by Lemma 8, and by Lemma 10 we can take any AUSO in the chosen Q_A-fiber. ■

Theorem 16

$$t_{acyc}(n) = \Omega\left(\frac{n^2}{\log_2 n}\right)$$

Proof We prove by induction for $n \geq 2$ that

$$t_{acyc}(n) \geq \frac{n^2}{2\lceil \log_2 n \rceil} - \frac{n}{2}.$$

This holds for $n = 2$ and $n = 3$. For $n \geq 4$, applying Lemma 15 and the induction assumption,

$$t_{acyc}(n) \geq n - \lceil \log_2 n \rceil + \frac{(n - \log_2 n)^2}{2\lceil \log_2(n - \lceil \log_2 n \rceil) \rceil} - \frac{1}{2}(n - \lceil \log_2 n \rceil) \geq \frac{n^2}{2\lceil \log_2 n \rceil} - \frac{n}{2}.$$

■

References

[1] J. Matoušek, *The number of unique-sink orientations of the hypercube*, Combinatorica 26 (2006), 91–99.

[2] I. Schurr, T. Szabó, *Finding the Sink Takes Some Time: An Almost Quadratic Lower Bound for Finding the Sink of Unique Sink Oriented Cubes*, Discrete Comput Geom 31 (2004), 627–642.

[3] T. Szabó, E. Welzl, *Unique sink orientations of cubes*, In Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science, 547–555, 2001.