Computing Cut Numbers*

C. Sohler and M. Ziegler, Heinz Nixdorf Institute
Department of Mathematics and Computer Science
University of Paderborn, 33095 Paderborn, Germany
{csohler,ziegler}@uni-paderborn.de

Abstract

What is the minimum number of hyperplanes that slice all edges $\mathcal{E}_d$ of the d-dimensional hypercube? The answers have been known for $d \leq 4$ and d=6. This work settles the problem for $d = 5$. More precisely, a computer search implies that 4 hyperplanes do not suffice for this purpose (but 5 do).

We also develop computational approaches for attacking this extremal problem from combinatorial geometry in higher dimensions. They allow us to determine for example all maximal sliceable subsets of $\mathcal{E}_d$ up to dimension 7.

1 Introduction

Let $\mathcal{G}_d = (\mathcal{V}_d, \mathcal{E}_d)$ denote the d-dimensional hypercube, i.e., the geometric graph on vertex set $\mathcal{V}_d = \{-1,+1\}^d \subseteq \mathbb{R}^d$ with $|\mathcal{E}_d| = d\,2^{d-1}$ undirected edges. It is obvious that the d hyperplanes $H_j = \{v \in \mathbb{R}^d : v_j = 0\}$ for $j = 1 \dots d$ slice all edges of $\mathcal{G}_d$. However M. Paterson [7] observed that for $d = 6$ one can do better: 5 sophisticatedly chosen hyperplanes suffice to cut whole $\mathcal{E}_6$.

Call the minimal number of hyperplanes needed to do so for $\mathcal{E}_d$ its *cut number* $S(\mathcal{E}_d)$. Furthermore write $M(\mathcal{E}_d)$ for the maximal number of edges that one single hyperplane can cut. These functions thus capture a combinatorial aspect of d-dimensional Euclidean geometry. Their values as well as asymptotic growth have been first asked for by O'Neil [6]. Table 1 reports some well known facts which imply $S(\mathcal{E}_d) = d$ for $d \leq 4$ and $S(\mathcal{E}_6) = 5$. $S(\mathcal{E}_5)$ however was only known to be either 4 or 5, and Victor Klee in his invited talk at CCCG99 [4] brought this open problem to the authors' attention.

This work describes the strategies which enabled us to exclude the case "$S(\mathcal{E}_5) \leq 4$", therefore proving $S(\mathcal{E}_5) = 5$. We proceed by first relating sliceability to feasibility of linear programs. The combinatorial complexity is then reduced to a (huge but computer tractable) number using hypercube symmetries and restricting to maximal sliceable sets. Finally, a dynamic programming approach allows to check the remaining cases practically.

We also present algorithms by which we want to attack higher dimensions as well. In particular, the decision between $S(\mathcal{E}_7) = 6$ and $S(\mathcal{E}_7) = 5$ might be achievable with sufficient computation power and patience. As intermediate results, complete lists of all maximal sliceable subsets $\mathcal{L} \subseteq \mathcal{E}_d$ (up to symmetries) have been determined and can be downloaded from our web page [11], see also Table 2.

$M(\mathcal{E}_d)$	$=$	$\lceil d/2 \rceil \cdot \binom{d}{\lceil d/2 \rceil}$	[6]
$S(\mathcal{E}_{d-1})$	$\leq$	$S(\mathcal{E}_d) \quad \leq \quad d$	
$S(\mathcal{E}_{d+d'})$	$\leq$	$S(\mathcal{E}_d) + S(\mathcal{E}_{d'})$	[9]
$S(\mathcal{E}_4)$	$\geq$	4	[2]
$S(\mathcal{E}_6)$	$\leq$	5	[7]
$S(\mathcal{E}_d)$	$\geq$	$|\mathcal{E}_d|/M(\mathcal{E}_d) \;=\; \Omega(\sqrt{d})$	

TABLE 1: PROPERTIES OF $S(\cdot)$ AND $M(\cdot)$.

2 Sliceable Sets of Segments

Consider a line segment $L = \{\lambda a + (1-\lambda)b : 0 \leq \lambda \leq 1\} =: [a,b]$ in $\mathbb{R}^d$ with endpoints a and b. A hyperplane with normal unit vector u and distance $t \geq 0$ to origin $H = \{w \in \mathbb{R}^d : w \cdot u = t\}$ *slices* L if the intersection $L \cap H$ is a point in the relative interior (a,b) of L. Put differently, a and b have to lie on different open halfspaces H^+ and H^-,

$$\text{i.e., either } \quad a \cdot u > t \quad \wedge \quad b \cdot u < t$$
$$\text{or } \quad a \cdot u < t \quad \wedge \quad b \cdot u > t.$$

We say that a set $\mathcal{L}$ of line segments is *sliceable* if some common hyperplane slices each $L = [a_L, b_L] \in \mathcal{L}$. A covering of $\mathcal{L}$ of k sliceable sets $\mathcal{L}^{(i)}$ is called a *k-slicing* of $\mathcal{L}$. $\mathcal{L}$ is *k-sliceable* if a k-slicing for it exists.

Thus $S(\mathcal{E}_5) \leq 4$ iff there is a 4-slicing of $\mathcal{E}_5$. The following considerations will allow us to reduce the otherwise ridiculously large number of putative coverings to a tractable size. These remaining cases have been fed into computer search and, since none of them was a 4-slicing, proved $S(\mathcal{E}_5) = 5$.

*Partially supported by DFG Grant Me872/7-3

More precisely, we argue that only coverings made up from *maximal* sliceable subsets need to be considered. Furthermore different coverings which can be transformed into one another by means of hypercube symmetries (e.g. permutation of coordinates) may be considered *equivalent* and have to be taken into account only once. An algorithm is presented which enables the determination of all *non*equivalent maximal sliceable subsets by reduction to linear programming. Based on these, we compute as intermediate step all 2-sliceable (non-equivalent maximal) subsets of $\mathcal{E}_5$. From these it is possible to conclude by computer search that 4−slicings of $\mathcal{E}_5$ do not exist.

3 Symmetries and Maximal Sets

First we will make use of the symmetries σ inherent to the hypercube $\mathcal{G}_d$:

- Reflections in direction $1 \le j \le d$, i.e., mappings $u = (u_1, \dots, u_j, \dots, u_d) \mapsto (u_1, \dots, -u_j, \dots, u_d)$
- Permutation $\pi \in \mathcal{S}_d$ of dimensions $u \mapsto (u_{\pi(1)}, \dots, u_{\pi(d)})$.
- Any of the $2^d \cdot d!$ combinations of the above. Clockwise rotation by 90° in the plane for instance is realized by $(x, y) \mapsto (y, -x)$.

Let $\sigma : \mathbb{R}^d \to \mathbb{R}^d$ denote any symmetry mentioned above, that is, an element of the group Σ_d of those linear automorphisms of $\mathbb{R}^d$ which map $\mathcal{G}_d$ to itself. The following observation says that sliceability is invariant unter symmetries:

Lemma 1 *For $\sigma \in \Sigma_d$ and $a, u \in \mathbb{R}^d$, it holds that $\sigma(a) \cdot \sigma(u) = a \cdot u$. In particular if $u \in \mathbb{R}^d$ and $t \ge 0$ are such that $H = \{w \in \mathbb{R}^d : w \cdot u = t\}$ slices segment $L = [a, b]$, then $\sigma(H) := \{w \in \mathbb{R}^d : w \cdot \sigma(u) = t\}$ slices $\sigma(L) := [\sigma(a), \sigma(b)]$. More generally if $\mathcal{L} \subseteq \mathcal{E}_d$ is sliceable by means of H, then so is $\sigma(\mathcal{L}) := \{\sigma(L) : L \in \mathcal{L}\} \subseteq \mathcal{E}_d$ by means of $\sigma(H)$.*

By virtue of this lemma it suffices to consider for each equivalence class $\Sigma_d(\mathcal{L}) := \{\sigma(\mathcal{L}) : \sigma \in \Sigma_d\}$ only *one* unique symmetry representative (USR), for example its lexicographically smallest member.

dim d	3	4	5	6	7
$\lvert\mathcal{E}_d\rvert$	12	32	80	192	448
$M(\mathcal{E}_d)$	6	12	30	60	140
$\lvert\Sigma_d\rvert$	48	384	3 840	46 080	645 120
#SSs	498	784 037	>10G	??	??
#MSSs	51	940	47 285	7 514 066	4 189 035 431
#USRs of MSSs	5	14	62	566	14754

TABLE 2: SIZES OF INTERMEDIATE RESULTS
LISTINGS OF THESE SETS CAN BE FOUND ON OUR WEB PAGE [11].

Second notice that it suffices to consider merely *maximal* sliceable sets in the sense that adding any new edge from $\mathcal{E}_d$ makes them *un*sliceable. Indeed to any k-slicing $(\mathcal{L}^{(i)})_{i=1..k}$, there exists a k-slicing where each member is maximal — a so called *maximal k-slicing*.

Table 2 illustrates the drastical reduction in complexity obtained from considering not *all* sliceable subsets (SSs) but only only maximal ones (MSSs) and their USRs. For example in dimension $d = 4$, there exist only 14 different USRs of MSSs — they are sketched in the appendix of the present work. However, the factor gained by symmetries is not as big as $|\Sigma_d|$ since, e.g., $\mathcal{E}_d$ is USR for only *one* MSS.

4 Linear Program

For finding all MSSs of $\mathcal{E}_d$, already Emamy-Khansary [2] employed the fact that each such MSS is the set of edges exterior to some *cut complex*. A cut complex is the subgraph of $\mathcal{G}_d$ induced by the vertices $\mathcal{V}_d \cap H^+$ for some oriented halfspace H^+.

Since the Vapnik-Chervonenkis-Dimension of d-space with respect to the range of halfspaces is $d+1$, there can be no more than $\sum_{i=0}^{d+1} \binom{|P|}{i} \le |P|^{d+1}$ sets of the form $P \cap H^+$ [3]. Choosing $P = \mathcal{V}_d$, this upper bound of $2^{d(d+1)}$ illustrates the gain achieved by considering MSS instead of the double exponentially many sliceable subsets.

Furthermore, a vertex subset $P \subseteq \mathcal{V}_d$ can easily be tested for being of the form $P = \mathcal{V}_d \cap H^+$. Indeed, the existence of an accordingly oriented hyperplane with parameters t and u is equivalent to feasibility of the following linear program with respect to $(u, t) \in \mathbb{R}^{d+1}$:

$$(1) \quad a \cdot u < t \quad \forall a \in P \qquad \wedge \qquad b \cdot u > t \quad \forall b \in \mathcal{V}_d \setminus P$$

The following lemma from [2] sais that any cut complex of size l emerges from some cut complex of size $l-1$ by adding an exterior vertex. This offers an efficient way of finding all cut complexes by means of dynamic programming with respect to their size. Taking the exterior edges of the cut complexes, a list of all MSSs is then obtained easily.

Lemma 2 *Let $P \subseteq \mathcal{V}_d$ be a cut complex of size $|P| = l+1$. Then there exists a cut complex P' of size l and $p \in \mathcal{V}_d \setminus P'$ adjacent to some $p' \in P'$ such that $P = P' \cup \{p\}$.*

However we point out that only USRs of MSSs are sought. Therefore it suffices to look merely for symmetrically *non*equivalent cut complexes, i.e., USRs of cut complexes. Also, P and $\mathcal{V}_d \setminus P$ have the same exterior edges, hence only cut complexes up to size $|\mathcal{V}_d|/2$ need to be considered. The algorithm thus works as follows:

- Start with the one-element list of all USRs of cut complexes (CCs) having size $l = 1$.
- Suppose $\mathfrak{A}$ is a list of all USRs of CCs of size l.
- Compute a similar list $\mathfrak{B}$ for $l + 1$ as follows:
- For each member $P' \in \mathfrak{A}$ and
- for each each $p \in \mathcal{V}_d \setminus P'$ adjacent to P'
  - Test whether $P := P' \cup \{p\}$ is a cut complex.
  - If so, compute its USR and
  - output that to $\mathfrak{B}$, provided it is new.
- Proceed inductively on l until $l = 2^{d-1}$.

Notice that symmetry reductions are performed on each level l so that $\mathfrak{B}$ contains only nonequivalent entries and stays small for the next stage. In fact, this is how we obtained the last line in Table 2. The USRs of CCs can be downloaded from [11] as well as the MSSs they induce. See in the appendix an illustration of the case $d = 4$. To obtain the full set of MSSs (second last line of Table 2), we explicitely symmetry-expanded these USRs. Analogously, taking all (different) subsets of MSSs yielded the SSs[1].

By storing *only* the MSSs in a kind of cache, sliceability queries can from now on be answered very efficiently without having to resort to expensive floating point operations any more. Indeed, encode subsets of $\mathcal{E}_d$ by binary strings of length $n = |\mathcal{E}_d|$ and store them in the well known [1] *radix tree* data structure $\mathfrak{T}_n$. Such $\mathfrak{T}_n$ represents a subset $S_n \subseteq \{0,1\}^n$ as follows: If $S_n = \emptyset$ then $\mathfrak{T}_n$=`NIL`; otherwise, $\mathfrak{T}_n$ consists of two subtrees $\mathfrak{T}_n = \big(\mathfrak{T}_{n-1}^{(0)}, \mathfrak{T}_{n-1}^{(1)}\big)$ storing the two sets $S_{n-1}^{(0)}, S_{n-1}^{(1)} \subseteq \{0,1\}^{n-1}$ respectively, where

$$S_{n-1}^{(i)} \quad := \quad \big\{(x_1, ..., x_{n-1}) : (x_1, ..., x_{n-1}, i) \in S_n\big\} \ .$$

Radix trees permit a branch-and-bound strategy for subset and superset queries. The latter enables sliceability queries, as

$$\mathcal{L} \subseteq \mathcal{E}_d \text{ sliceable} \quad \Longleftrightarrow \quad \exists \mathcal{L}' \in \mathfrak{T}_n : \ \mathcal{L}' \supseteq \mathcal{L} \ ,$$

if $\mathfrak{T}_n$ contains all (binary strings encoding) MSSs.

5 Dynamic Program w.r.t. k

By virtue of the previous section, we already know all maximal 1-sliceable $\mathcal{L} \subseteq \mathcal{E}_d$. The maximal 2-sliceable subsets (M2SSs) are among the pairwise unions of all MSSs — 'among' because $A \cup B$ is in general not *maximal* 2-sliceable even if $A, B \subseteq \mathcal{E}_d$ are MSSs. For removing these and by additional application of symmetry considerations, we arrive at the following way of finding the collection $\mathfrak{C}$ of USRs for all M2SSs:

- Start with $\mathfrak{A} = \{\text{MSSs}\}$, $\mathfrak{B} = \{\text{USRs for MSSs}\}$.
- Initialize $\mathfrak{C} := \emptyset$.
- For all $A \in \mathfrak{A}$ and all $B \in \mathfrak{B}$
  - Compute the USR C of $A \cup B$.
  - Skip, if $\mathfrak{C}$ already contains a superset of C.
  - Otherwise write C to $\mathfrak{C}$ and
  - remove from $\mathfrak{C}$ all subsets of C.

Subset- and superset-queries are implemented by means of the afore mentioned radix tree. It is important that only *one* of A, B may be chosen an USR:

Lemma 3 *Let C be a M2SS. There exist $A \in \mathfrak{A}$ and $B \in \mathfrak{B}$ and $\sigma \in \Sigma$ such that $C = \sigma(A \cup B)$. In particular, the collection $\mathfrak{C}$ output by the above algorithm contains USRs of* all *M2SSs.*

Proof: By definition, $C = \tilde{A} \cup \tilde{B}$ for MSSs $\tilde{A}$ and $\tilde{B}$.

$$C \quad = \quad \tilde{\sigma}^{-1}\big(\tilde{\sigma}(\tilde{A} \cup \tilde{B})\big) \quad = \quad \tilde{\sigma}^{-1}\big(\tilde{\sigma}(\tilde{A}) \cup \tilde{\sigma}(\tilde{B})\big)$$

for all $\tilde{\sigma} \in \Sigma$. Now choose $\tilde{\sigma}$ such that $\tilde{\sigma}(\tilde{B})$ is USR. Then taking $\sigma := \tilde{\sigma}^{-1} \in \Sigma$, $A := \tilde{\sigma}(\tilde{A}) \in \mathfrak{A}$, and $B := \tilde{\sigma}(\tilde{B}) \in \mathfrak{B}$ proves the first claim. For the second claim notice that, as invariant to the algorithm's loop, all members of $\mathfrak{C}$ are 2-sliceable. Consequent to the above argument, an USR for each M2SS becomes inserted once. On the other hand, it is skipped or removed only if there is some proper superset in $\mathfrak{C}$ — which cannot happen for *maximal* 2-slicable subsets. □

Starting this algorithm once again with input consisting of all M2SSs and their USRs in term gives the M4SSs, and so on. This approach generalizes to finding all USRs for MkSSs for $k \in \mathbb{N}$ by iterating the above algorithm $\ell(k)$ times, where $\ell(k) \leq 2 \log k$ denotes the least length of an addition chain [5, Sect. 4.6.3] for k. In particular $S(\mathcal{E}_d) > k$ iff, among the USRs of MkSSs, there is no $|\mathcal{E}_d|$-element sets.

In fact, this is how we proved $S(\mathcal{E}_5) > 4$: The 47 285 MSSs of $\mathcal{E}_5$ and their 62 USRs were fed into the above algorithm and yielded 175 946 different USRs of 597 175 584 M2SSs. The USRs of M4SSs, we did not compute: it sufficed to look for a M2SS A and an USR B of a M2SSs such that $|A \cup B| \geq |\mathcal{E}_5|$. Having sorted the 175 946 USRs of M2SSs with respect to their cardinality, this allowed for additional *branch-and-bound* optimization. Namely on input of an M2SS A, only USRs B with $|B| \geq 80 - |A|$ have to be considered.

[1] These however were not needed for our purposes; their numbers are listed only for reference.

6 Implementation Details

Our computations have been performed on a workstation with 600MHz AMD K7 processor, 256MB of RAM, 10GB hard disk, Linux operating system and 10MBit/sec Ethernet. They took a total of about two months CPU time, most of it spent in the last step of going through the $597\,175\,584 \times 175\,946$ pairs (A, B). Determining the USRs of MSSs was comparatively fast: less than five seconds for $d \le 5$, one minute for $d = 6$, and two hours for $d = 7$.

The radix tree data structure $\mathfrak{T}$ had to undergo some space optimization: In its original realization, each single bit x_n requires to store a 32-bit pointer to the subtrees $\mathfrak{T}_{n-1}$ at an overhead of factor 32. In our implementation, sub-trees containing only a few elements were therefore replaced by simple arrays.

Concerning the simplex algorithm's implementation, we chose LP_SOLVE v3.0 [10] by Michel Berkelaar. Like many others, it looks for (componentwise) *non-negative* solution vectors – a restriction which can be circumvented in (1) by considering $v := u + 1$ as independent variable[2] and solving

$$a \cdot v - t < a \cdot 1 \quad \forall a \qquad \wedge \qquad b \cdot v - t > b \cdot 1 \quad \forall b$$

instead: Now $v \ge 0$ because u is unit vector. Due to the $(d + 1)$-th variable t, solutions are not unique: if (u, t) is feasible, so is $(\lambda u, \lambda t)$ for arbitrary small λ. This ambiguity was resolved by asking LP_SOLVE for the *maximal* feasible t subject to the constraints $v \le 2$.

Another difficulty arose from ensuring that hyperplanes cut through the *interior* of edges. Mathematically, this reflects the *strictness* of the inequalities in (1). But distinction between $\le$ and $<$ is far too volatile to numerical jitters. For this reason, we introduced a small parameter ϵ and required

$$a_L \cdot v \le (1 - \epsilon)\, t \qquad \wedge \qquad b_L \cdot v \ge (1 + \epsilon)\, t\ .$$

7 Robustness and Reliability

The value of the afore mentioned ϵ is crucial since some k-slicing might contain an edge $[a, b]$ which is cut by a hyperplane at $a(\epsilon/2) + b(1 - \epsilon/2)$, that is definitely in its interior but closer than ϵ to its boundary. For that reason, we performed several runs of our algorithms for different values. These revealed robustness in the sense that the reported MSSs were independent of ϵ in the range from 10^{-2} to 10^{-4} — smaller choices induced warnings in LP_SOLVE announcing "numerical instability".

Such warnings lead us to another issue, a problem common to all computer-based 'proofs': their soundness! Indeed, any programming mistake on behalf of ours could possibly void all claims and so could things we relied on such as errors in the compilers, libraries, and even the processer itself (e.g., Pentium bug). Such reproaches had been discussed intensively in connection with computer-proving the famous Four-Colour Theorem, and we followed [8] in establishing confidence by

- performing extensive testing of each module,
- disclosing all intermediate results [11],
- encouraging others to verify them and
- to repeat computations in own implementations.
- Furthermore, we reproduced the complete lists of USRs for MSSs of $\mathcal{E}_3$ and of $\mathcal{E}_4$ from [2]
- and verified $|\mathcal{L}| \le M(\mathcal{E}_d)$ for all $\mathcal{L} \in \text{MSSs}(\mathcal{E}_d)$.

But the probably most striking argument for the validity of our software is that we could in fact reproduce the result $S(\mathcal{E}_6) \le 5$. Looking through the computed $7\,514\,066$ MSSs of $\mathcal{E}_6$ [11], we found all five slicings used by Paterson [7], and our algorithms agreed that these do cover $\mathcal{E}_6$.

8 Higher Dimensions

Since $S(\mathcal{E}_5) = 5$, Table 1 implies $S(\mathcal{E}_6) = 5$. The next unknown cut number is $S(\mathcal{E}_7) \in \{5, 6\}$. Comparision of the numbers in Table 2 for $d = 5$ with $d = 7$ indicate a factor $100\,000$ increase in complexity. Although this is out of reach by the above method, we believe it to be tractable using the following observation:

If $\mathcal{G}_{d+1}$ can be sliced by k hyperplanes, so can each of its (isomorphic copies of) subcubes $\mathcal{G}_d$. Suppose all k-slicings of $\mathcal{E}_d$ in d-space are given in a list $\mathfrak{A}$. Then the restrictions of any k-slicing of $\mathcal{E}_{d+1}$ to its subcubes must be members from $\mathfrak{A}$.

More formally, let $\{-1\} \times \mathcal{G}_d$, $\{+1\} \times \mathcal{G}_d$ $\mathcal{G}_d \times \{-1\}$, and $\mathcal{G}_d \times \{+1\}$ denote the 'left', 'right', 'bottom', and 'top' subcubes of $\mathcal{G}_{d+1}$; correspondingly $\{-1\} \times \mathfrak{A}$ for the k-slicings of $\{-1\} \times \mathcal{G}_d$ induced from $\mathfrak{A}$ by the canonical embedding $\mathbb{R}^d \ni x \mapsto (-1, x) \in \mathbb{R}^{d+1}$. The following algorithm computes a list $\mathfrak{B}$ of all k-slicings of $\big(\{-1\} \times \mathcal{E}_d\big) \cup \big(\{+1\} \times \mathcal{E}_d\big)$:

- For each $\big(\mathcal{L}_-^{(1)}, ..., \mathcal{L}_-^{(k)}\big) \in \{-1\} \times \mathfrak{A}$
- and each $\big(\mathcal{L}_+^{(1)}, ..., \mathcal{L}_+^{(k)}\big) \in \{+1\} \times \mathfrak{A}$:
- Compute the sets $\mathcal{L}_-^{(i)} \cup \mathcal{L}_+^{(i)}$, $i = 1...k$;
- Use $\mathfrak{T}$ to check whether each of them is sliceable.
- If yes, write this k-tuple to $\mathfrak{B}$.

Remark 1 If is *not* necessary to consider, within each two k-slicings, the permuted combinations $\mathcal{L}_-^{(i)} \cup \mathcal{L}_+^{(\pi(i))}$ for $\pi \in \mathcal{S}_k$: If $\big(\mathcal{L}^{(1)}, ..., \mathcal{L}^{(k)}\big) \in \mathfrak{A}$ is a k-slicing of $\mathcal{G}_d$, then so is $\big(\mathcal{L}^{(\pi(1))}, ..., \mathcal{L}^{(\pi(k))}\big) \in \mathfrak{A}$.

[2] By abuse of notation, identify $1 \in \mathbb{R}$ with $(1, 1, \ldots, 1) \in \mathbb{R}^d$.

Remark 2 After having, from the two lists of all k-slicings of $\{-1\} \times \mathcal{E}_d$ and $\{+1\} \times \mathcal{E}_d$ respectively, computed the list $\mathfrak{B}$ of all k-slicings of $\big(\{-1\}\times\mathcal{E}_d\big)\cup\big(\{+1\}\times\mathcal{E}_d\big)$, these are easily transformed to list $\mathfrak{C}$ containing all k-slicings of $\big(\mathcal{E}_d \times \{-1\}\big) \cup \big(\mathcal{E}_d \times \{+1\}\big)$. Based on these two, one similarly computes all k-slicings of

$$\big(\{-1\}\times\mathcal{E}_d\big)\cup\big(\{+1\}\times\mathcal{E}_d\big)\cup\big(\mathcal{E}_d\times\{-1\}\big)\cup\big(\mathcal{E}_d\times\{+1\}\big).$$

But for any dimension, those four sets cover whole $\mathcal{E}_{d+1}$, so that the latter list is in fact one of all k-slicings of $\mathcal{G}_{d+1}$. In particular, it is empty iff $S(\mathcal{E}_{d+1}) > k$.

Remark 3 This approach is practical only for $k = S(\mathcal{E}_d)$, i.e. to decide whether $S(\mathcal{E}_{d+1}) = S(\mathcal{E}_d)$ or $= S(\mathcal{E}_d) + 1$. Indeed, for $k > S(\mathcal{E}_d)$, the 'surplus' hyperplanes may be placed arbitrarily. This degree of freedom makes the list $\mathfrak{A}$ of k-slicings of $\mathcal{E}_d$ explode.

Remark 4 It suffices to consider merely k-slicings with pairwise disjoint members and only symmetrically non-equivalent ones. In this context, two k-tuples $\big(\mathcal{L}^{(1)}, ..., \mathcal{L}^{(k)}\big)$ and $\big(\tilde{\mathcal{L}}^{(1)}, ..., \tilde{\mathcal{L}}^{(k)}\big)$ of subsets from $\mathcal{E}_d$ are equivalent if there is some *common* symmetry $\sigma \in \Sigma_d$ such that $\tilde{\mathcal{L}}^{(i)} = \sigma\big(\mathcal{L}^{(i)}\big)$ for all $i = 1...k$. Like in Section 5, either $\big(\mathcal{L}_-^{(1)}, ..., \mathcal{L}_-^{(k)}\big)$ or $\big(\mathcal{L}_+^{(1)}, ..., \mathcal{L}_+^{(k)}\big)$ — but not both — may be chosen USR in this sense. Similarly, reduction with respect to equivalence under permutations $\big(\mathcal{L}_-^{(\pi(1))}, ..., \mathcal{L}_-^{(\pi(k))}\big)$ is possible for one of them.

Now the remaining key point is, how to obtain the list $\mathfrak{A}$ of all k-slicings of $\mathcal{G}_d$ for input to the above algorithm. The following strategy provides a means to that:

This time suppose we have already determined a list $\mathfrak{A}$ of all disjoint k-slicings of all k-sliceable subsets of $\mathcal{E}_d$ up to size $l \in \mathbb{N}$. For any member $\big(\mathcal{L}^{(1)}, \ldots, \mathcal{L}^{(k)}\big)$ of $\mathfrak{A}$, try to enlarge some $\mathcal{L}^{(i)}$ by a new edge $L \in \mathcal{E}_d \setminus \bigcup_{i=1}^{k} \mathcal{L}^{(i)}$. If $\mathcal{L}^{(i)} \cup \{L\}$ is still sliceable, then write the enlarged k-tuple to $\mathfrak{B}$.

This gives a list of all disjoint k-slicings of k-sliceable subsets up to size $l+1$. Again, symmetries offer a way of optimization: Only USRs of k-sliceable subsets need to be considered. Consequently, their k-slicings can be chosen to be USRs (in the sense of Remark 4), too; similarly for reduction w.r.t equivalence under permutations.

9 Conclusions and Future Work

We determined the value of $S(\mathcal{E}_5)$ to be 5, thereby solving a so far open problem restated by Klee. The immediate consequence is $S(\mathcal{E}_6) = 5$ and $S(\mathcal{E}_7) \in \{5, 6\}$.

Our methods are based on extensive computer calculations of both kinds floating point numerics and combinatorial search. The simplex method allowed to decide for $P \subseteq \mathcal{V}_d$ whether it is a "cut complex" or not. From the cut complexes we obtained the maximal sliceable sets. Symmetry considerations and restriction to *maximal* sets permitted to drastically reduce the search space.

All intermediate results, as well as exhaustive lists of the maximal sliceable subsets of $\mathcal{E}_5$, $\mathcal{E}_6$, and $\mathcal{E}_7$, are available for download on out web page [11]. They may serve as a basis for computing cut numbers in higher dimensions. These we believe, together with the above robustness and reliability considerations, to establish strong confidence in the soundness of our investigations.

We also presented algorithms for higher dimensions. It is our believe that, in an appropriate environment of distributed computing and with sufficient patience, the value $S(\mathcal{E}_7)$ can be determined as well. Notice that one of the most time expensive steps — symmetry reduction, i.e., finding USRs for CCs and MSSs — can be easily partitioned into independent sub-tasks and thus deserves parallelized treatment.

At the current state of implementation, the bottleneck might arise from main memory limitations. In particular, removal of non-maximal members from a large list seems practicle only if it fits into RAM entirely.

References

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest: "Introduction to Algorithms", McGraw-Hill.

[2] M.R. Emamy-Khansary: "On the cuts and cut number of the 4-cube", pp.211-227 in *Journal of Combinatorial Theory Series A* **41** (1986).

[3] D. Haussler and E. Welzl: "Epsilon-nets and simplex range queries", pp.127-151 in *Discrete Comput. Geom.* **2** (1987).

[4] Victor Klee: "Shapes of the Future – Some Unresolved Problems in High-Dimensional Intuitive Geometry", in *Proceedings of the 11th Canadian Conference on Computational Geometry* (1999), p.17

[5] D.E. Knuth: "The Art of Computer Programming, volume 2: semi-numerical algorithms", Addison-Wesley.

[6] P. O'Neil: "Hyperplane cuts of an n-cube", pp.193-195 in *Discrete Mathematics* **1** (1971).

[7] M. Paterson, unpublished; see Example 3.72 in [9].

[8] N. Robertson, D.P. Sanders, P. Seymour, R. Thomas: "A New Proof of the Four-Colour Theorem", pp.17-25 in *ERA of the AMS* **2** (1996).

[9] Michael E. Saks: "Slicing the hypercube", pp.211-255 in *Surveys in Combinatorics*, Cambridge University Press (1993), Keith Walker (editor).

[10] H. Schwab: "Documentation for LP_SOLVE", `ftp://ftp.es.ele.tue.nl/pub/lp_solve` (1996).

[11] `http://www.uni-paderborn.de/cs/cubecuts`

Appendix: The 14 USRs of CCs and MSSs in Four Dimensions

Cut Complex induced MSS USR of MSS