Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

JÖRG RAMBAU

TOPCOM: Triangulations of Point Configurations and Oriented Matroids

ZIB-Report 02-17 (April 2002)

TOPCOM: TRIANGULATIONS OF POINT CONFIGURATIONS AND ORIENTED MATROIDS

JÖRG RAMBAU

ABSTRACT. TOPCOM is a package for computing triangulations of point configurations and oriented matroids. For example, for a point configuration one can compute the chirotope, components of the flip graph of triangulations, enumerate all triangulations. The core algorithms implemented in TOPCOM are described, and implentation issues are discussed.

1. INTRODUCTION

TOPCOM[1] uses a combinatorial model of point configurations in order to compute their triangulations in arbitrary dimensions: the oriented matroid.[2, 3, 4] This allows to avoid round-off errors in all calculations once the oriented matroid of the point configuration has been computed. If the point configuration is has rational coordinates then its oriented matroid can be computed with exact arithmetics. The number of operations in exact arithmetics necessary to compute the relevant oriented matroid data for a particular problem is minimal in the sense that any coordinate-based method achieving the same goal needs at least as many operations in exact arithmetics. Naïve coordinate-based approaches like checking intersections of simplices by linear programming need usually a lot more.

2. MATHEMATICAL BACKGROUND

The basic idea utilized in TOPCOM is the following combinatorial characterization of triangulations:[5]

Theorem 2.1. *Let $\mathcal{A}$ be a full-dimensional point configuration in $\mathbb{R}^d$. A set T of full-dimensional $(d+1)$-subsets (simplices) of $\mathcal{A}$ is a triangulation of $\mathcal{A}$ iff*

- (IP) *For every pair S, S' of simplices in T, there exists no circuit (Z^+, Z^-) in $\mathcal{A}$ with $Z^+ \subseteq S$ and $Z^- \subseteq S'$ (Intersection Property).*
- (UP) *For each facet of a simplex S in T there is another simplex $S' \neq S$ having F as a facet, or F is contained in a facet of $\mathcal{A}$ (Union Property).*

With the following characterization, we only need the chirotope representation of the oriented matroid of $\mathcal{A}$ in order to check a triangulation:[6]

Theorem 2.2. *Let $\mathcal{A}$ be a full-dimensional point configuration in $\mathbb{R}^d$. A set T of full-dimensional $(d+1)$-subsets (simplices) of $\mathcal{A}$ is a triangulation fo $\mathcal{A}$ iff*

- (CP) *For every interior facet F, there are exactly two simplices $F \cup s$ and $F \cup s'$ in T such that s and s' have different signs in the cocircuit defined by F (Cocircuit-Property).*
- (EP) *$\mathcal{A}$ contains a simplicial facet or there is an extension in the relative interior of $\mathcal{A}$ in general position that is contained in the interior of exactly one simplex (Extension Property).*

3. Main Algorithms

We present the main algorithms implemented in TOPCOM together with rough estimates for the asymptotic run-time complexity.[1]

We denote by n the number of points in $\mathcal{A}$, by r the rank of $\mathcal{A}$, which equals the dimension (denoted by d) plus one, and by $c := n - r$ the corank of $\mathcal{A}$. In all practically solvable cases you may think of n and r as small numbers compared to the total number of triangulations of $\mathcal{A}$.

3.1. **Compute the Chirotope.** First, we describe how we compute the interface from geometric information in the coordinates of the points to purely combinatorial data. The chirotope χ is a function assigning a sign in $\{0, +, -\}$ to each r-subset of $\mathcal{A}$; this describes the orientation of the corresponding sequence of points.

$$\chi : \begin{cases} \binom{\mathcal{A}}{d+1} & \to & \{+, -, 0\} \\ (i_1, i_2, \ldots, i_{d+1}) & \mapsto & \operatorname{sign}\bigl(\det(a_{i_1}, a_{i_2}, \ldots, a_{i_{d+1}})\bigr) \end{cases} \tag{1}$$

Naïvely, we have to compute $\binom{n}{r}$ determinants of size r, each requiring $O(r^3)$ steps if we proceed by an elimination algorithm (unit cost per arithmetic operation assumed).

If we organize the computation in a certain tree we can reuse some elimination operations. The nodes at level k of the computation tree are matrices M_k in column normal form corresponding to a subset A_k of k columns of $\mathcal{A}$. A child A_{k+1} of a node is produced by adding a column to the right of A_k to M_k and performing elimination steps until we arrive at a matrix M_{k+1} in column normal form. The leaves of the tree are all possible $(r \times r)$-submatrices of $\mathcal{A}$ in equivalent triangular form, and their determinant can be computed easily.

Counting nodes and operations in the tree, we get that the number of operations per determinant is essentially $O(\frac{r+c}{1+c} r^2)$. Why? We first count the nodes in the levels of the computation tree as follows: Let $g(k, j) = \binom{j-1}{k-1}$ be the number of submatrices of A with k columns and rightmost column at index j. Then the number of nodes in level k of the tree is equal to $\sum_{j=k}^{n-r+k} g(k, j)$. We estimate the number of operations in each node of level k by rk because of the following considerations. We have to eliminate $k - 1$ values in the kth column. We start the elimination in row 1 and proceed from top to bottom. If the mth row, $1 \leq m < k$, of the mth column is non-zero we can eliminate (i.e., set to zero) the non-zero in the mth row of the newly added kth column by computing $r - m \leq r$ new numbers. If m is the smallest index such that the mth row of column m is zero, we swap column k and column m. The first $k - 1$ columns of the resulting matrix are again in column normal form, and row m of column k is zero now. Therefore, and we can proceed with the next m. The total number of operations is therefore at most rk. The total number of nodes in the tree is given by $\sum_{k=1}^{r} \sum_{j=k}^{n-r+k} g(k, j)$. Hence, the number of operations is at most $G(n, r) := \sum_{k=1}^{r} \sum_{j=k}^{n-r+k} g(k, j)kr$. The number of operations per determinant is then $G(n, r)/\binom{n}{r}$. The analysis of this expression can be done most easily in a computer algebra program to get the result $G(n, r) = \frac{(c+r+1)r^2}{c+2}$, which asymptotically has the claimed growth.

This proves that the computation tree yields savings of order r per determinant whenever the codimension is not much smaller than the dimension.

[1] In particular algorithms, these estimates can be improved by using custom-made data structures, but TOPCOM does not exploit this, since these data structures would perform worse in other algorithms.

The chirotope values are stored in a hash table so that they can be retrieved in amortized constant time. Alternatively, one could compute the lexicographic index of each simplex and store its value in an array, yielding access to each chirotope value constant times computing the lexicographic index.

3.2. **Check a Potential Triangulation.** From now on, we assume that the chirotope of $\mathcal{A}$ has been preprocessed. Given the combinatorial characterizations of triangulations, checking the correctness of a triangulation is, in principle, straight-forward. The only issue is efficiency. TOPCOM uses characterization 2.2 for checking the correctness of triangulations:

- (CP) requires checking the link of all d-facets in T. The computation of all links can be accomplished in $O(r|T|)$ operations by scanning all the simplices in T and collecting their contributions to the links of their facets in a hash map (supporting unique insert in amortized constant time). If the link of some facet F contains only one point, we need to check whether F is in the boundary of $\mathcal{A}$. This can be done by retrieving at most c chirotope values. In case the link has more than two elements, we output 'invalid' immediately. In the case that there are two elements in the link of a facet, the orientations of the two points in the corresponding cocircuit can be checked by retrieving two chirotope values. Summarizing, the number of operations needed to check (CP) is $O(cr|T|)$.
- The existence of a simplicial facet is not checked. Instead, (EP) is always checked by utilizing a lexicographic extension p in the interior of a some simplex in T. The formula for the chirotope values involving p requires the retrieval of at most r other chirotope values (only one if $\mathcal{A}$ is in general position).[4] For all simplices in T we need to check whether p is in its interior. This can be done by retrieving r chirotope values involving p. Thus, for the whole procedure, we need at most $O(r^2|T|)$ steps.

Thus, in at most $O(\max\{c, r\}r|T|)$ we can check the validity of a triangulation.

3.3. **Construct a Placing Triangulation.** The construction of a placing triangulation is an incremental process. During the process, we maintain in each step k

- the set of points $\mathcal{A}_k$ that is currently triangulated
- a placing triangulation T_k of $\mathcal{A}_k$
- a set $\mathcal{F}_k$ of all boundary facets of T_k that are interior in $\mathcal{A}$, i.e., those facets
  - that are facets of exactly one simplex in T_k
  - that are not in the boundary of $\mathcal{A}$, i.e., the corresponding cocircuits are neither positive nor negative in $\mathcal{A}$.

The first r points to be added have to form a valid (i.e., full-dimensional) simplex. Finding such a simplex can be integrated in the computation of the chirotope at the cost of storing a simplex S with non-zero chirotope. Consequently, $\mathcal{A}_r$ equals S, T_r consists of the simplex S, and $\mathcal{F}_r$ contains those facets of S that are not in the boundary of $\mathcal{A}$.

Adding a new point a_{k+1} to $\mathcal{A}_k$, yielding $\mathcal{A}_{k+1}$, for some $k \geq r$ will add all simplices $F \cup a_{k+1}$ to $\mathcal{T}_k$ for which $F \in \mathcal{F}_k$ is visible from a_{k+1}, i.e., the sign of a_{k+1} in the cocircuit defined by F is opposite to those signs of the points in $\mathcal{A}_k$ that are non-zero: this yields T_{k+1}. (There are points in $\mathcal{A}$ with non-zero sign since $\mathcal{A}_k$ is full-dimensional for all $k \geq r$.) The visible facets in $\mathcal{F}_k$ are removed and the new non-boundary facets of the new simplices are added to $\mathcal{F}_k$: this yields $\mathcal{F}_{k+1}$.

The process stops as soon as $\mathcal{F}_k$ is empty for some $k \geq r$. This happens at the latest when $k = n$, but maybe earlier, in which case not all points are used for the placing

triangulation. If one wants a triangulation that uses all the points the missing points are added one by one by performing stellar subdivisions inside existing simplices: the points are "flipped-in".

The number of operations on the sets $\mathcal{F}_k$ is bounded by the number of all $(d-1)$-faces of the resulting placing triagulation T, which is at most $O(r|T|)$. The use of universal hashing facilitates unique insert and deletion in amortized constant time. For one visibilty check we need to retrieve at most c chirotope values (if $\mathcal{A}$ is in general position two values suffice). Each facet occuring in some $\mathcal{F}_k$ may have to be checked for visibility in each but one step of the construction.

Thus, in total we need at most $O(rc^2|T|)$ operations (resp. $O(rc|T|)$ operations if $\mathcal{A}$ is in general position) for all visibility checks. This dominates the computation of a placing triangulation.

3.4. **Explore a Flip-Graph Component.** Flips are stored in TOPCOM as pairs of outgoing and incoming simplices (flip-out sets and flip-in sets). TOPCOM uses a standard Breadth-First-Search (BFS) procedure to explore the flip graph.[7] Since the graph is not given explicitely, we need to construct edges dynamically during the exploration. To this end, we maintain for each triangulation a node containing its set of flips, i.e., a set of edges leading out of this node. Given a flip in a triangulation we can compute the resulting triangulation in amortized time $O(r)$ by using a special data structure for triangulations that allows for unique insert and deletion in amortized constant time (e.g., hash tables with universal hashing or something derived thereof).

New flips are found using the chirotope as follows: Assume, $\mathcal{S}$ is the set of flips in T, and we have used one of those flips $f \in \mathcal{S}$ to discover a new triangulation T'. Then the set $\mathcal{S}'$ of flips of T' inherits all those flips in $\mathcal{S}$ whose flip-out sets are disjoint from f's flip-out set. Finding these flips takes time $O(r|\mathcal{S}|)$.

Moreover, there are some new flips. Their flip-out sets have always non-empty intersection with the flip-in set of f. Thus, for finding new flips we can restrict ourselves to potential flips containing simplices of f's flip-in set. To this end, we first build all potential supports (sets of involved vertices) of flips by collecting all $(r+1)$-supersets of simplices in the flip-in set that might support a flip. This can be done in $O(r|T'|)$ by finding adjacent simplices in T'; alternatively one can do this in $O(rn)$ by adding all possible points to an in-flip simplex, which results, however, in more non-flippable supports. Since checking whether or not a support is actually flippable is the expensive operation, TOPCOM uses the first method. The maximal number of possible supports returned by this procedure is bounded from above by the number of boundary facets of the flip-in set of f which is $O(r)$.

To check the flippability of an $(r+1)$-set, we first compute its circuit signature $Z = (Z^+, Z^-)$ by retrieving $r+1$ chirotope values. The two possible triangulations $T^+(Z)$ and $T^-(Z)$ can be computed from this data in time $O(r)$. Then we need to check whether one, say $T^+(Z)$, of the two possible circuit triangulations is a sub-complex of T'. If Z is in general position (no chirotope value is zero) then we need to check containment in T' for all the at most $O(r)$ simplices in $T^+(Z)$. This can be done in amortized time $O(r)$ by using a special data structure for triangulations that allows for membership test in amortized constant time. If Z is not in general position we need to compute the links of all (maximal) simplices in T^+ in T'. This takes at most time $O(r|T'|)$. If all simplices in T^+ have the same non-empty link then Z is indeed flippable, and the flip-out and flips-in sets are the unions of the simplices in $T^+(Z)$ resp. $T^-(Z)$ with the common link; they can therefore be deduced in time $O(r^2)$ (rank of $\mathcal{A}$ times corank of Z).

Summarizing, we need time $O(r^2|T'|)$ for finding new flips in T'; this reduces to $O(r|T'| + r^2)$ when $\mathcal{A}$ is in general position. Thus, generating the new flip set $\mathcal{S}'$ takes time $O(r^2|T'| + r|\mathcal{S}|)$, which is in $O(r^2(|T'| + |T|))$.

Note that computing flips from scratch for each node would yield a time complexity of $\Omega(r|T'|^3)$—assuming the same algorithm is used for checking flippability of a potential support of a flip.

Using a data structure for triangulations that allows to retrieve an adjacent simplex in constant time (the adjacency graph[8]) can reduce the time complexity in the non-general-position case. This, however, would result in a higher complexity when computing an adjacent node, a higher complexity for equality checks for triangulations, and—most important—in a substantially higher memory consumption. Moreover, the sizes of problems for which an asymptotically superior data structure returns the result faster is out of reach memory-wise anyway because for small problems the TOPCOM data structures facilitate very small constants in the computing time bounds. This, however, has to be carefully monitored in the future as the computers are getting more and more powerful.

3.5. **Explore the Partial Order of Partial Triangulations.** Here, TOPCOM employs a standard Depth-First-Search (DFS) in the Hasse-diagram of all partial triangulations, partially ordered by inclusion. Partial triangulations are sets of simplices satisfying (IP). Every node in the DFS-tree contains a partial triangulation T, a set of simplices $A(T)$ (admissibles) of simplices that can be added to T without violating (IP), and a the set $F(T)$ of facets of simplices in the boundary of T that are in the interior of $\mathcal{A}$. The leaves in the DFS-tree are the non-extendable partial triangulations, and some of them are in fact triangulations, namely those T with $F(T) = \emptyset$. This is essentially the method of enumerating maximal independet sets[8] in the intersection graph of all simplices, except that, first, we don't proceed via adjacent simplices only and, second, we use the chirotope for checking proper intersections of simplices. The basic method goes back to 1980.[9]

Edges (i.e., new simplices) are explored in lexicographic order. Consider a node $(T, A(T))$ and add a simplex $S \in A(T)$ to T in order to discover node $(T', A(T')$. Then $A(T)$ is updated to $A(T) \setminus S$, so that the none of the other branches above $(T, A(T)$ can ever reach a partial triangulation with subcomplex $T' = T \cup S$.

This way, we only need the memory to store one complete branch in the DFS-tree. If M is the maximal number of simplices in a triangulation of $\mathcal{A}$, then storing $O(M)$ simplices on a stack for the partial triangulations plus storing $O(M\binom{n}{r})$ simplices in the admissibles fields suffice. Important is the fact that the memory requirements do not depend on the number of triangulations of $\mathcal{A}$.

TOPCOM uses a preprocessed hash map with all possible $\binom{n}{r}$ simplices as keys. The value $A(S)$ of a simplex S is the set of all simplices S' such that S and S' form an admissible pair, i.e., $\{S, S'\}$ satisfies (IP). This preprocessing takes time $O(\binom{n}{r}^2)$. The table allows for the following fast update algorithm: if T' is discovered by adding simplex S to T then $A(T') = A(T) \cap A(S)$. This step can be accomplished in time $O(|A(T)|)$, which is—crudely estimated—in $O(\binom{n}{r})$.

The set of all interior facets of $\mathcal{A}$ is also preprocessed by checking for each $(r-1)$-subset the facet property. This needs time $O(r\binom{n}{r-1})$. The set $F(T)$ can be updated by adding the interior boundary facets of S to $F(T)$ modulo 2. These are can be done in time $O(r|F(T)|)$, which is in $O(r^2M)$.

In total, we need time $O(\binom{n}{r}^2)$ preprocessing time and time $O(r\binom{n}{r})$ per ouput node. The time needed per triangulation is way higher because there are a lot more proper partial triangulations than triangulations.

3.6. **Check Regularity (Requires External LP Solver with Exact Arithmetics).** Checking regularity of a triangulation T requires checking the feasibility of a linear program with a constraint for each interior facet F in T. The constraint induced by F adjacent to the simplices S and S' in T expresses that S and S' can be "folded" at F such that the resulting signed volume spanned by S and S' is positive.

This is a condition that can be expressed as a condition on the determinant of the homogeneous coordinate vectors of the points in $S \cup S'$. This determinant develops into a linear constraint in the height variables for each point. The coefficients turn out to be the same determinants that specify the signs for the chirotope values on all subsets of $S \cup S'$. If we therefore save the determinants rather than the signs in the chirotope then we can form each constraint in time $O(r)$, resulting in a time complexity of $O(r^2|T|)$ for setting up the complete linear program.

This effort is currently dominated by actually solving the linear program in exact arithmetics. TOPCOM employs for this task the cdd package.[10]

3.7. **Symmetry-Handling.** TOPCOM can exploit symmetries that are given by the user in form of generating permutations. The main savings using symmetries can be implemented in the flip graph exploration. In the BFS procedure there are always three kinds of nodes:[7] unknown nodes (white nodes), nodes that have been discovered but may have unknown neighbors (gray nodes), and discovered nodes all of whose neighbors have also been discovered (black nodes). Black nodes can be removed from memory.

The BFS procedure behaves friendly w.r.t. symmetries. That means it is possible (by an equivalent-edge marking procedure) to store only one representative per symmetry class and to remove the black nodes from memory. This works without ever discovering a white node equivalent to a black one.[3] Using the symmetries that leave a triangulation unchanged one can reduce the effort of finding flips.

Symmtries are also observed in checking (CP) in a potential triangulation. In contrast to this, (EP) has to be checked for all simplices—not only for all symmetry classes of simplices.

4. Data Structures and Implementation Details

4.1. **Simplicial Complexes.** Simplicial complexes are regarded as the sets of their maximal simplices. TOPCOM aims at using set data structures that perform fast on the following tasks: equality check, membership test, unique insert, deletion, intersection, union.

Simplices are implemented as dynamic bitsets. The largest number of points ever handled in TOPCOM was 324 in dimension six (the Santos triangulation). This still requires only 41 Bytes per simplex, as opposed to $7 \cdot 4 = 28$ Bytes for an array representation. For all potentially accessible enumeration problems the number of points must be much smaller: in most applications 32 bits suffice, in which case also the memory consumption is superior.

Equality check needs $\lceil n/32 \rceil$ integer equality checks. Membership test, unique insert, and deletion just need a single (very fast) bit operation each. Intersections and unions are linear in n; however, on a normal PC, 32 bits can be processed at a time, thus this is practically very fast for reasonable values of n.

Simplicial complexes are based on dynamic bitsets where each bit represents a simplex. The assignment of bit positions to simpleces remains fixed during one complete run of the program. When a simplex occurs for the first time it gets the next free bit assigned; this assignment is stored in a dictionary. Retrieving the simplex corresponding to a bit is implemented via an array indexed by integers; getting the bit position of a simplex is accomplished by a hash table look-up. Both can be done (theoretically and practically) in amortized constant time.

This way, operations on simplicial complexes can work fast via bitset operations. Another advantage is the excellent behaviour in terms of memory cache misses (a main bottleneck on nowaday's computers): retrieving even a long bitstring from memory is very fast whenever the bits are stored consecutively. If we enumerate a flip graph component then for each possible simplex there is at least one triangulation (usually many) that contains that simplex. Thus, we need to store each possible simplex at least once, whence the use of the TOPCOM data structure does not harm memory-wise.

The main draw-back of this structure is that equality checks take time $O(\binom{n}{r})$, with small constants, though. The plausible observation is that for dense simplicial complexes (number of simplices is of order $\binom{n}{r}$), the structure is performing very well; for very sparse complexes, one should use an ordinary representation.

When the sizes of accessible problems grow, TOPCOM will probably switch to an asymptotically faster date structure. For (most) instances solvable on nowaday's computers, the bitstring technique works fastest.

4.2. **Chirotopes.** The chirotope values for simplices are stored in a hash map, allowing for amortized constant time retrieval. If the problem is very large, the table can be assigned a fixed size and works as a cache with random evict.

4.3. **Node in the Flip-Graph.** TOPCOM stores the triangulation T as a simplicial complex and the flips of T as a hashmap assigning marked or unmarked to the representation of the flip.

A flip's internal representation is simply a dependent set with $r+1$ points, which is stored as a simplex. By deleting $r+1$ bits and retrieving $r+1$ chirotope values, one can obtain from this a flip represented by a pair of simplicial complexes: the flip-in and the flip-out simplices. Thus, deleting the flip-out set from and adding the flip-in set to a triangulation works on bitset level. Marking of a particular flip would work in amortized constant time in hash maps. Because the set of flips assigned to one triangulation is not that large, TOPCOM now rather uses sorted maps for flips (logarithmic time for marking, but less overhead).

4.4. **Node in the Tree of Partial Triangulations.** The triangulation T, the set $A(T)$ of admissible simplices, and the set of uncovered interior facets $F(T)$ are all stored as simplicial complexes. This way the computing the intersection of admissible sets is a bitset operation. Moreover, adding interior facets modulo two means an xor-operation on bitset level, which is also very fast in practice.

4.5. **Symmetries.** Symmetries are stored as arrays σ of integers, where $\sigma[i]$ is the image of vertex i under the symmetry σ. The images of simplices and simplicial complexes are computed vertex by vertex. Here is certainly potential for improvement by using more sophisticated structures.

5. Computational Experiments

We present some results of experiments with the data structures used in TOPCOM-0.10.0 in Table 1. In Table 2, we reproduce a table from [3] containing some large enumeration problems solved by TOPCOM for the first time (to the best of our knowledge).

	data structure representing a simplicial complex		
operation	STL set(vec(int))	STL set(bitstr.)	TOPCOM
insert 99 simplices	100%	51%	62%
assignment	100%	48%	0.2%
delete 89 simplices	100%	79%	75%
intersect (~ 50 simplices)	100%	52%	4.5%
lex. compare (~ 50 simplices)	100%	53%	108%
equality false (~ 50 simplices)	100%	100%	700%
equality true (w.r.t. line above)	84700%	44950%	933%

TABLE 1. Operations in simplicial complexes (1.8GHz Pentium IV, 1GByte RAM, simplices chosen randomly) using three different data structures for simplicial complexes: STL (=C++ Standard Template Library) set(vec(int))=STL set, based on red-black trees, of arrays of integers; STL set(bitstr.)=STL set of TOPCOM's dynamic bitstrings; TOPCOM= TOPCOM's bitstrings specifying simplex indices w.r.t. a table of essentially all possible simplices; parameters comparable to a triangulations of the 4-cube; note that equality checks in STL sets are fast when the cardinalities of the operands differ, as was the case in the equality-false test instances above; the genuin equality checks for TOPCOM's simplicial complexes are much faster but cannot take advantage of the cardinality function

what	*configuration*	*description*	*result*
# triangulations	$C(12,5)$	cyclic polytope	`5,049,932`
# triangulations	$C(13,7)$	cyclic polytope	`6,429,428`
# fine triangulations	(4×5)-lattice	two-dim. lattice (20 points)	`2,822,648`
# flip graph component	$\Delta_3 \times \Delta_3$	product of tetrahedra	`4,533,408`
# flip graph component	C^4	four-dimensional cube	`92,487,256`
check (IP') & (UP')	Santos triang.	six-dimensional construction	`okay`
# flips	Santos triangulation	six-dimensional construction	`0`

TABLE 2. Some figures computed by TOPCOM for the first time; cyclic polytopes have connected flip graphs;[5] so have two-dimensional point sets; fine triangulations use all the vertices

References

[1] J. Rambau. TOPCOM—Triangulations of Point Configurations and Oriented Matroids. Software under the Gnu Public Licence, available under `http://www.zib.de/rambau/TOPCOM.html`, 1999.

[2] J. A. de Loera. *Triangulations of Polytopes and Computational Algebra*. PhD thesis, Cornell University, 1995.

[3] J. Pfeifle and J. Rambau. Computing triangulations using oriented matroids. ZIB-Report 02-02, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2002.

[4] A. Björner, M, Las Vergnas, B. Sturmfels, N. White, and G. M. Ziegler. *Oriented matroids*, volume 46 of *Encyclopedia of Mathematics*. Cambridge University Press, Cambridge, 1993.
[5] J. Rambau. Triangulations of cyclic polytopes and higher Bruhat orders. *Mathematika*, 44:162–194, 1997.
[6] J. A. de Loera, S. Hoşten, F. Santos, and B. Sturmfels. The polytope of all triangulations of a point configuration. *Documenta Mathematika*, 1:103–119, 1996.
[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
[8] F. Takeuchi and H. Imai. Enumerating triangulations for products of two simplices and for arbitrary configurations of points. In *Computing and Combinatorics*, pages 470–481, 1997.
[9] E. Lawler, J. Lenstra, and A. H. G. Rinnooy Kan. Generating all maximal independent sets: Np-hardness and polynomial-time algorithms. *SIAM Journal on Computing*, 9:558–565, 1980.
[10] K. Fukuda. cdd—an implementation of the double description method. `http://www.ifor.math.ethz.ch/~fukuda/cdd_home/cdd.html`.

ZUSE-INSTITUT BERLIN, TAKUSTR. 7, 14195 BERLIN, GERMANY