1 Introductory Remark

This version uses vastly improved enumeration methods for

- enumeration of all triangulations,
- enumeration of circuits from points,
- enumeration of cocircuits from points.

Moreover, lock-free multithreading is used on demand for enumeration and/or symmetry processing.

For really large problem instances (co-dimension above 13), the exploration-by-flipping of the regular component of the flip-graph in TOPCOM’s `points2(n)triangs` takes a lot of memory. The exploration of the (in general smaller) subregular component of the flip graph can be done with less memory consumption by using MPTOPCOM by Jordan, Joswig, and Kastner ([https://polymake.org/doku.php/mptopcom](https://polymake.org/doku.php/mptopcom)). The enumeration of all triangulations, however, can be done with similar memory consumption now by the TOPCOM client `points2(n)alltriangs`. This client, for the time being, is the fastest method in TOPCOM to enumerate all triangulations of point configurations.
2 What is TOPCOM?

TOPCOM is a collection of clients to compute Triangulations Of Point Configurations and Oriented Matroids, resp.

The algorithms use only combinatorial data of the point configuration as is given by its oriented matroid. Some basic commands for computing and manipulating oriented matroids can also be accessed by the user.

3 How do I use TOPCOM?

All programs read the input from stdin and write the result to stdout so that you can pipe the results to the next command.

A point configuration is given by a matrix (enclosed in square brackets) whose columns (enclosed in square brackets) are the homogeneous coordinates (seperated by commas) of the points in the configuration. A square could be specified as follows.

\[
\begin{bmatrix}
0,0,1, & 0,1,1, & 1,0,1, & 1,1,1
\end{bmatrix}
\]

You may specify generators of the combinatorial symmetry of a point configuration as permutations of the vertex numbers. The symmetry generators of the square read as follows (observe that the count starts at 0):

\[
\begin{bmatrix}
3,2,1,0, & 2,3,0,1, & 0,2,1,3
\end{bmatrix}
\]

Some algorithms also read in a given seed triangulation given by a set of sets of numbers, where elements of a set are enclosed in curly brackets and separated by commas. For example:

\{0,1,2\},\{1,2,3\}\

4 Commands

The following commands are provided:
points2prettyprint (New.) Displays the point and their symmetry generators in a more readable form.

points2chiro Computes the chirotope of a point configuration.

chiro2dual Computes the dual of a chirotope.

chiro2circuits Computes the circuits of a chirotope.

points2circuits Dto. for point configurations (using a faster method).

chiro2cocircuits Computes the cocircuits of a chirotope.

points2cocircuits Dto. for point configurations (using a faster method).

cocircuits2facets Computes the facets of a set of cocircuits.

points2facets Computes the facets of a point configuration.

points2gale Computes a Gale transform of a point configuration.

chiro2circuits Computes the circuits of a point configuration.

chiro2cocircuits Computes the cocircuits of a point configuration.

points2facets Computes the facets of a point configuration.

points2nflips Computes the number of flips of a point configurations and the seed triangulation.

points2flips Computes all flips of a point configurations and the seed triangulation.

chiro2placingtriang Computes the placing triangulation of a chirotope given by the numbering of the elements.

points2placingtriang Dto. for point configurations.

chiro2finetriang Computes a fine (i.e., using all vertices) triangulation by placing and pushing.

points2finetriang Dto. for point configurations.

chiro2triangs Computes all triangulations of a chirotope that are connected by bistellar flips to the seed, which is a regular triangulation if no seed is given in the input file.

points2triangs Dto. for point configurations.
chiro2ntriangs Computes the number of all triangulations of a chirotope that are connected by bistellar flips to the seed, which is a regular triangulation if no seed is given in the input file.

points2ntriangs Dto. for point configurations.

chiro2finetriangs Computes all fine triangulations (the ones that use all the points sometimes called “full”) of a chirotope that are connected by bistellar flips to a fine seed triangulation.

points2finetriangs Dto. for point configurations.

chiro2nfinetriangs Computes the number of all fine triangulations of a chirotope that are connected by bistellar flips to a fine seed triangulation.

points2nfinetriangs Dto. for point configurations.

chiro2alltriangs Computes all triangulations of a chirotope.

points2alltriangs Dto. for point configurations.

chiro2nalltriangs Computes the number of all triangulations of a chirotope.

points2nalltriangs Dto. for point configurations.

chiro2allfinetriangs Computes all fine triangulations (sometimes called “full”) of a chirotope.

points2allfinetriangs Dto. for point configurations.

chiro2nallfinetriangs Computes the number of all fine triangulations of a chirotope.

points2nallfinetriangs Dto. for point configurations.

chiro2mintriang Computes a triangulation of a chirotope with a minimum number of simplices.

points2mintriang Dto. for point configurations.

B_S n Computes the vertices and symmetry generators of the permutation polytope of the symmetric group of degree $n$, also known as the Birkhoff polytope.

B_A n Computes the vertices and symmetry generators of the permutation polytope of the alternating group of degree $n$, also known as the even Birkhoff polytope.
B_D n Computes the vertices and symmetry generators of the permutation polytope of the dihedral group of degree n.

B_S_center n Computes B_S n with an additional center point.

B_A_center n Computes B_A n with an additional center point.

B_D_center n Computes B_D n with an additional center point.

cube d Computes the vertices and symmetry generators of a d-cube.

cyclic n d Computes the vertices and symmetry generators of the cyclic d-polytope with n vertices.

cross d Computes the vertices and symmetry generators of the d-dimensional cross-polytope.

lattice n m Computes the nm two-dimensional lattice points with non-negative coordinates at most (n − 1, m − 1) and their symmetry generators.

hypersimplex d k [ℓ] Computes the vertices and symmetry generators of the k-th hypersimplex in dimension d. A third parameter makes it the S-hypersimplex with coordinate sums equal to k or ℓ.

santos_triang Computes the point configuration, the symmetry, and the Santos triangulation (without flips).

5 Command Line Options

The following command line options are supported. Note that not all options are sensible for all clients.

Options concerning input/output from files

- i [filename]: read input from [filename] instead of stdin.

Options concerning output of information

- h or --help Print a usage message.
- d Debug.
-v  Verbose.

--heights  Output a height vector for every regular triangulation (implies --regular).

--flips  Output all flips in terms of IDs of adjacent triangulations. (Can be used to generate the flip graph.)

--asy  Write asymptote graphics commands into file (in rank-3 triangulations, points are drawn as well). The graphics contains a view of the point configuration (only in rank 3), the enumeration tree with a classification of enumeration nodes into solutions, non-canonical nodes, deadends, and early detected deadends, as well a statistics file showing a histogram of enumeration node types. The output file has to be processed by the computer graphics compiler asymptote (https://asymptote.sourceforge.io) using the asy-library Combinatorial_Geometry.asy and the \LaTeX-macros in triangbook_macros.sty inside share/asy/.

Options for checking input

--checktriang  Check seed triangulation.

Options for reporting properties of discovered triangulations:

--flipdeficiency  Check triangulations for flip deficiency during flip-graph exploration.

--findregular \([k]\)  (New form.) Check every \(k\)-th triangulation for regularity and stop if a regular one is found during flip-graph exploration.

Options concerning which triangulations are output (no influence on flip-graph exploration)

--noorbitcount  Only count symmetry classes, not the total number.

--cardinality \([k]\)  Count/output only triangulations with exactly \(k\) simplices.

--maxcardinality \([k]\)  Count/output only triangulations with at most \(k\) simplices.

--unimodular  (New.) Output unimodular triangulations only; while this does not reduce the effort of flip graph exploration, since unimodular triangulations are in general not connected by themselves, it does reduce the effort of extension graph exploration line in points2nalltriangs.
--nonregular  Output non-regular triangulations only; note that this does not reduce the effort of flip-graph exploration, since non-regular triangulations are in general not connected by themselves.

**Options concerning which triangulations are explored**

--regular  Search for regular triangulations only (checked liftings are w.r.t. the last homogeneous coordinate, e.g., last coordinates all ones is fine); note that this may reduce the effort of exploration, since regular triangulations are connected by themselves.

--noinsertion  Never flip-in a point that is unused in the seed triangulation.

--reducepoints  Try to greedily minimize the number of vertices used while flipping; keep a global upper bound on the current minimal number of vertices and do not accept triangulations with more vertices.

--keepcard  Never change the cardinality of triangulations by flipping.

**Options concerning symmetries**

--affinesymmetries  (New form.) Assume that the symmetries are affine, in particular, that they conserve regularity.

--isometricsymmetries  (New.) Assume that the symmetries are isometric, in particular, that they preserve volume.

--nosymmetries  Ignore the symmetries.

**Options controlling the internals of the clients**

--memopt  Save memory by using caching techniques.

--usegkz  (New.) Use GKZ vectors as a finger print in symmetry handling (only for points with isometric symmetries).

--usenaivesymmetries  (New.) Use naive full traversal of all symmetries for symmetry handling.

--useswitchtables  (New.) Use Jordan-Joswig-Kastner switch tables for symmetry handling.
--usesymmetrytables (New.) Use tables of classified symmetries for symmetry handling. Obsolete, since slower than the other options.

--symtables [n] (New.) Use [n] symtables for preprocessing symmetries. Obsolete, since slower than the other options.

--preprocesschiro (New.) Preprocess the chirotope (default for points2[n]alltriangs).

--preprocesspoints (New.) Heuristically transform points (only relevant for (co)circuit enumeration).

--simpidxsymmetries (New.) Preprocess a representation of the symmetry group on simplex indices (only relevant for triangulation enumeration).

--userandomorder (New.) Sort simplices in preprocessed index table randomly (only for points with isometric symmetries).

--usevolumeorder (New.) Sort simplices in preprocessed index table by volume (only for points with isometric symmetries).

--usevolumes (New.) Use volumes to check extendability of partial triangulations (only for points with isometric symmetries).

--fullextensioncheck (New.) Put more effort in the check of extendability of a partial triangulation.

--noextensioncheck (New.) Skip the check of extendability of a partial triangulation.

--extensioncheckfirst (New.) Check extendability prior to symmetry.

--preprocesspoints Preprocess the coordinate matrix of the points (slightly useful for (co-)circuit enumeration)

--chirocache [n] Set the chirotope cache to n elements.

--localcache [n] Set the cache for local operations.

--qsopt_ex Use QSopt_ex for regularity checks (not thread-safe).

--soplex Use soplex for regularity checks (requires separate installation of soplex).
Options concerning multi-threading

--parallelenumeration Use multiple threads for enumeration.

--workbuffercontrol Control the interrupt of workers by size of the current work-buffer.

--parallelsymmetries Use multiple threads only locally for symmetry checks.

--threads [n] Use [n] threads (if possible).

--minnodebudget [n] Let each thread process at least [n] nodes (to avoid multi-threading overhead).

--maxnodebudget [n] Let each thread process at most [n] nodes (to avoid thread starving).

--scalenodebudget [n] Scale the default node budget by [n] percent (n integer)

--minworkbuffer [n] (Currently unused.) Try to keep the work buffer above [n] nodes (to balance overhead and thread starving).

--maxworkbuffer [n] (Currently unused.) Try to keep the work buffer below [n] node (to balance overhead and thread starving).

Options for warm starts from previous calculations

These options currently only work for an interrupted flip graph exploration.

--dump Write intermediate results into a file.

--dumpfile [dumpfilename] Write intermediate results into file dumpfilename (default: TOPCOM.dump).

--dumpfrequency [k] Dump the results of each kth BFS round

--dumprotations [k] Dump into k different rotating files.

--read Read intermediate results from a file.

--readfile [readfilename] Read intermediate results from file dumpfilename (default: TOPCOM.dump.[rotationnumber]).
6 Examples

In the subdirectory examples you find some example inputs for TOPCOM routines. If you want to see the input in a prettier way, then you can type the following:

```
points2prettyprint < cube_3.dat
```

It prints the matrix of point coordinates and the symmetry generators in tabular form. Or:

```
points2chiro < lattice_3_3.dat
```

outputs the sign string of the chirotope of the sub-lattice of integer points \((i, j)\) with \(i, j = 0, 1, 2\).

```
points2chiro < lattice_3_3.dat | chiro2ntriangs
```

or

```
points2ntriangs < lattice_3_3.dat
```

yields the number of triangulations that are connected to the regular ones by bistellar flips.

```
points2ntriangs --regular --affinesymmetries < moae.dat
```

counts all regular triangulations of the “mother of all examples”, two nested triangles in the plane.

```
points2nalltriangs < lattice_3_3.dat
```

yields the number of all triangulations via a new algorithm symmetric lex-subset reverse-search on partial triangulations. It is mostly faster than flip-graph exploration and much faster than in older versions of TOPCOM; moreover, it does not take a lot of memory.

The example r12.chiro is the chirotope of the oriented matroid \(R12\) with disconnected realization space, constructed by Jürgen Richter-Gebert. If you want to compute, e.g., a placing triangulation of \(R12\) then type
chiro2placingtriang < r12.chiro

The facets of a 4-cube can be computed by

\[ \text{cube 4 | points2facets} \]

but be aware of the fact that this is not an efficient way of computing facets of a point configuration. It is, however, numerically stable because rational arithmetics is used. Most notably, you can check the Santos triangulation by

\[ \text{santos_triang | points2nflips -v --memopt --checktriang} \]

Recall that the options mean:

- `-v` verbose;
- `--memopt` save memory;
- `--checktriang` check seed triangulation.

A new result as of version 1.0.10 is that TOPCOM can now enumerate all triangulations of larger point configurations like the 4-cube in reasonable time. The speed depends on the computational environment and the options:

\[ \text{cube 4 | points2nalltriangs -v --parallelenumeration --workbuffercontrol} \]

uses verbose output and multi-threaded symmetry processing with load-balancing based on work buffer control, i.e., the workers stop and push the open nodes into the job list whenever the master’s job list contains fewer elements than there are workers.

Since not every point configuration can be represented in rational coordinates, it is useful to enumerate triangulations from the chirotope data. This way you can compute, e.g., all triangulations of the regular icosahedron:

\[ \text{chiro2nalltriangs -i icosahedron.chiro -v} \]

Note the newly supported alternative syntax `-i [filename]` for the specification of the input file.

The enumeration of circuits and cocircuits up to symmetry is now a non-trivial part of TOPCOM. For example:
cube 4 | points2ncircuits -v --parallelenumeration --workbuffercontrol

enumerates the symmetry classed of all the circuits of the 4-cube.

Moreover, the clients to explore the flip graph for points can now use the GKZ vector as a fingerprint to detect known triangulation classes in case all symmetries are isometric. This is faster, but needs some more memory for storing all the known GKZ vectors. Moreover, a basic multi-threaded symmetry check can be used. You can activate all of this by:

cube 4 | points2ntriangs --parallelsymmetries --usegkz --isometricsymmetries

As an example for the graphics output the 2-by-3 lattice in dimension two is presented. Graphics output is supported by the enumeration of all triangulations, circuits and co-circuits. The files Combinatorial_Geometry.asy and triangbook_macroses.sty (to be found under share/asy in the TOPCOM home directory) must be present in the working directory where TOPCOM_graphics.asy is compiled via asy:

lattice 2 3 | points2nalltriangs -v --parallelenumeration --asy
asy TOPCOM_graphics.asy

The point configuration can be found in TOPCOM_points.pdf (see Figure 1). The enumeration tree can be found in TOPCOM_tree_graph.pdf (see Figure 2). Here, a clover icon means a triangulation in a new symmetry class, a recycling icon means a node in

![Figure 1: The points](image)
Figure 2: The enumeration tree with classified nodes – here with three threads and workbuffercontrol

an old symmetry class, a "stop-sign icon" means a deadend with no possible admissible lex-larger extension, a "bulb icon" means an early detected deadend with an uncoverable interior facet, and a "lightning-sign icon" means that no further extension option can prevent an uncoverable interior facet. Statistics can be found in TOPCOM_statistics.pdf (see Figure 3).
Figure 3: The statistics of classified nodes