Title	Enumerating Triangulations for Arbitrary Configurations of Points and for Products of Two Simplices
Author(s)	Takeuchi, Fumihiko; Imai, Hiroshi; Imai, Keiko
Citation	数理解析研究所講究録 (1997), 992: 98-105
Issue Date	1997-05
URL	http://hdl.handle.net/2433/61151
Right	
Type	Departmental Bulletin Paper
Textversion	publisher

数理解析研究所講究録
992巻 1997年 98-105

Enumerating Triangulations for Arbitrary Configurations of Points and for Products of Two Simplices

竹内史比古[1] Fumihiko Takeuchi fumi@is.s.u-tokyo.ac.jp
今井浩[1] Hiroshi Imai imai@is.s.u-tokyo.ac.jp
今井桂子[2] Keiko Imai imai@ise.chuo-u.ac.jp

Abstract

We propose two algorithms to enumerate triangulations. These algorithms enumerate all triangulations, regular or not, for arbitrary configurations of points in any dimensions. Our first algorithm characterizes triangulations as maximal independent sets of the intersection graph. This graph has the maximal dimensional simplices of the given point configuration as vertices, and edges between two simplices whose intersection is not a face for at least one of them. We enumerate the maximal independent sets of this graph, which form a superset of triangulations. The algorithm runs in time proportional to the size of this superset and requires memory only of the size of a triangulation. Our second algorithm enumerates the independent sets of the intersection graph whose interior is connected. These sets also form a superset of triangulations. This algorithm runs in time proportional to the size of this superset and requires memory twice the size of a triangulation.

We are also interested in the triangulations of products of two simplices. Since the polytope of this product is symmetric, it is important to enumerate the classes of triangulations in respect of this symmetry. We modify our second algorithm to enumerate the classes of independent sets with connected interior, which forms a superset of the classes of triangulations. This modification also runs in time proportional to the size of this superset and requires memory twice the size of a triangulation.

1 Introduction

Two efficient triangulation enumeration algorithms are proposed in this paper. They both run in time proportional to the size of a superset of triangulations and require only the memory several times the size of a triangulation.

Triangulations are interesting as a mathematical structure and important in practical application. There are many results for triangulations in two dimensions. Regular triangulations, a subset of triangulations, have been studied recently and algorithms to enumerate them are found [3], [4], [7], [8]. However, no efficient algorithm to enumerate all triangulations, including nonregular ones, for dimensions higher than two has been known. The algorithms in this paper enumerate these objects.

Our first algorithm characterizes triangulations as maximal independent sets of the intersection graph, and applies a general maximal independent set enumeration algorithm. The intersection graph here is the graph with all maximal dimensional simplices the vertices and edges between those intersecting improperly. This algorithm works in time proportional to the number of maximal independent sets. The required memory is only of the size of a triangulation. Reducing the space required is important, because memorizing all triangulations, whose number easily becomes huge, is impractical.

When triangulations form a proper subset of the maximal independent sets, the gap between them results in loss of time complexity. If this gap is small, this algorithm is efficient, the first efficient one, to enumerate all triangulations. The existence of this gap is determined geometrically by the configuration of points. In two dimensions this does not happen, and in three dimensions, we have Schönhardt's polyhedron (cf. [12, 10.2.1]) for example. However we are thinking that the gap may be small even in higher dimensions.

Our second algorithm enumerates the independent sets of the intersection graph whose interiors are connected. Such sets can be generated incrementally by adding an adjacent simplex. These connected independent sets form a superset of triangulations. This algorithm runs in time proportional to the size of this super set and requires memory twice the size of a triangulation.

Products of two simplices are polytopes rather simple. Their triangulations, though relation with Gröbner bases are known, are not yet well understood [14], [15]. So, enumeration is meaningful. These polytopes are highly symmetric. The product of k and l dimensional simplices has the symmetry of the direct product of two symmetric groups $S_{k+1} \times S_{l+1}$. So, it is not smart to count all triangulations naively, because we may count the "same" one $(k+1)!\,(l+1)!$ times. De Loera devised a program to enumerate regular triangulations for given sets of points. The program can take this symmetry into account, and he enumerated the triangulations, all of which are regular, for the case of $\Delta_2 \times \Delta_3$ and $\Delta_2 \times \Delta_4$ [4], [5]. When the dimensions become larger, even the number of classes divided by symmetry becomes huge. De Loera is using breadth first search in his program, so all visited triangulations should be kept in the memory, and the memory constraint becomes serious in larger cases. Our algorithm to enumerate the classes of triangulations for this product is a modification of our second algorithm. It runs in time proportional to the number of classes of connected independent sets, and requires memory only twice the size of a triangulation.

We begin by preparations for triangulations of arbitrary configurations of points (Section 2) and products of two simplices (Section 3). Then we show the algorithm to enumerate triangulations as maximal independent sets (Section 4) and as connected independent sets (Section 5).

[1]Department of Information Science, University of Tokyo, Tokyo, Japan 113
[2]Department of Information and System Engineering, Chuo University, Tokyo, Japan 112

2 Triangulations for arbitrary configurations of points

Let $\mathcal{A} = \{a_1, \ldots, a_n\} \in \mathbb{R}^k$ be a collection of points, and $\mathrm{conv}(\mathcal{A})$ its convex hull. We call a pair $(\mathrm{conv}(\mathcal{A}), \mathcal{A})$ *a configuration of points.* A simplicial complex T is *a triangulation* of $(\mathrm{conv}(\mathcal{A}), \mathcal{A})$ if its skeleton $|T| = \cup T$ equals $\mathrm{conv}(\mathcal{A})$ and its points are among $\mathcal{A}$. The triangulations we consider do not have new points. We often represent a simplex by the set of its vertices.

Definition 2.1
Let $(\mathrm{conv}(\mathcal{A}), \mathcal{A})$ be an arbitrary configuration of points.

- $\mathcal{S} = \{\sigma \in 2^{\mathcal{A}} :$ maximal dimensional simplex of $(\mathrm{conv}(\mathcal{A}), \mathcal{A})\}$
- Two simplices in $\mathcal{S}$ *intersect* if their intersection is not a face for at least one of them.
- The *intersection graph* of $\mathcal{S}$ is a graph with $\mathcal{S}$ the vertices and edges between two intersecting simplices.
- $\mathcal{I} = \{I \in 2^{\mathcal{S}} :$ independent set of the intersection graph of $\mathcal{S}\}$
- $\mathcal{M} = \{I \in 2^{\mathcal{S}} :$ maximal independent set of the intersection graph of $\mathcal{S}\}$
- Two simplices in $\mathcal{S}$ are *adjacent* if they share a facet. For any independent set $I \in \mathcal{I}$ its *adjacency graph* is a graph with simplices in I the vertices and edges between two adjacent simplices.
- $\mathcal{I}_{\mathrm{con}} = \{I \in \mathcal{I} :$ the adjacency graph of I is connected$\}$
- $\mathcal{M}_{\mathrm{con}} = \{I \in \mathcal{I}_{\mathrm{con}} :$ maximal in set inclusion in $\mathcal{I}_{\mathrm{con}}\}$
- $\mathcal{T} = \{\{\sigma_1, \ldots, \sigma_r\} \in 2^{\mathcal{S}} :$ a set of maximal dimensional simplices of a triangulation of $(\mathrm{conv}(\mathcal{A}), \mathcal{A})\}$

$\mathcal{I}_{\mathrm{con}}$ is formed by the independent sets which can be made incrementally: starting by an empty set and adding recursively an adjacent simplex, where we promise that any simplex can be added to the empty set. We call the elements of $\mathcal{I}_{\mathrm{con}}$ the *connected independent sets.* We regard $\mathcal{T}$ as the set of triangulations.

The maximal dimensional simplices in a triangulation in $\mathcal{T}$ must not intersect, so $\mathcal{T} \subset \mathcal{I}$. Furthermore, we cannot add anymore maximal dimensional simplex to a triangulation without making an intersection, so any triangulation is maximal in $\mathcal{I}$. Thus $\mathcal{T} \subset \mathcal{M}$. In such manner triangulations can be characterized as maximal independent sets of the intersection graph [9]. In Section 4 we show an algorithm to enumerate $\mathcal{M}$, which leads to the enumeration of $\mathcal{T}$. The difference between $\mathcal{T}$ and $\mathcal{M}$ becomes a loss. This sort of thing happens for Schönhardt's polyhedron (cf. [12, 10.2.1]). Schönhardt's polyhedron is a concave polyhedron made by twisting a triangle of a prism a little bit. No simplex with vertices among the six vertices is included in this polyhedron. Thus the set made by the three tetrahedra fitting the outer concave parts of this polyhedron becomes a maximal independent set of the convex polytope of the six points. But, it is not a triangulation, because the inner part is left. The authors do not know if this kind of gap occurs for the case of the product of two simplices.

A triangulation of a convex set $\mathrm{conv}(\mathcal{A})$ is an independent set with a connected adjacency graph. Thus $\mathcal{T} \subset \mathcal{M}_{\mathrm{con}} \subset \mathcal{I}_{\mathrm{con}}$. In Section 5 we enumerate $\mathcal{I}_{\mathrm{con}}$, which leads to the enumeration of the triangulations $\mathcal{T}$. The difference between $\mathcal{T}$ and $\mathcal{I}_{\mathrm{con}}$ becomes a loss.

3 $\Delta_k \times \Delta_l$ and its symmetry

The standard d-simplex Δ_d is the convex hull $\mathrm{conv}\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{d+1}\}$ in $\mathbb{R}^{d+1}$. We use $\boldsymbol{e}_i$ or $\boldsymbol{f}_j$ for unit vectors whose i-th or j-th element is one and the rest zeros. The product of two standard simplices $\Delta_k \times \Delta_l$ is

$$\Delta_k \times \Delta_l = \mathrm{conv}\{\begin{pmatrix} \boldsymbol{e}_i \\ \boldsymbol{f}_j \end{pmatrix} \in \mathbb{R}^{k+l+2} : i \in \{1, \ldots, k+1\}, j \in \{1, \ldots, l+1\}\}.$$

In Figure 1 we show $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$ for example. Triangulations of the product of k and l dimensional simplices are the triangulations of the point configuration $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$, where $\mathrm{vert}(\Delta_k \times \Delta_l)$ are the vertices. Examples of triangulations are shown in Figure 2.

First we state three lemmas for later use. The computation of the volume of $(k+l)$-simplices in a triangulation of $\Delta_k \times \Delta_l$, which is constant, leads the first.

Lemma 3.1
The number of $(k+l)$-simplices included in a triangulation of $\Delta_k \times \Delta_l$ is $(k+l)!/k!\,l!$.

The $(k+l)$-simplices in $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ correspond to spanning trees of the complete bipartite graph $K_{k+1,l+1}$ [7, 7.3.D.]. This derives the second.

Lemma 3.2
The number of $(k+l)$-simplices in $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ is $(k+1)^l(l+1)^k$.

The problem in the lemma below can be reduced to judging the existence of a cycle in a subgraph of a directed $K_{k+1,l+1}$ (cf. [5, Lemma 2.3.]), so the time complexity follows.

Lemma 3.3

Given two maximal dimensional simplices in $\Delta_k \times \Delta_l$, judging whether their intersection is a face of both of them or not can be done in $O(k+l)$ time.

The product $\Delta_k \times \Delta_l$ has a symmetric structure: even if we commute the axes of each simplices, the shape of the product does not change. Let us state this symmetry formally.

Definition 3.4 (equivalence on simplices and triangulations)
Let $S_{k+1} \times S_{l+1}$ be the direct product of symmetric groups, and $(p,q) \in S_{k+1} \times S_{l+1}$.

- $S_{k+1} \times S_{l+1}$ acts on the vertices of $\Delta_k \times \Delta_l$:

$$(p,q)\begin{pmatrix} e_i \\ f_j \end{pmatrix} = \begin{pmatrix} e_{p(i)} \\ f_{q(j)} \end{pmatrix}.$$

- The action of $S_{k+1} \times S_{l+1}$ on the simplices of $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ is induced by the action on the vertices:

$$(p,q)\,\mathrm{conv}\{s_1,\ldots,s_d\} = \mathrm{conv}\{(p,q)s_1,\ldots,(p,q)s_d\}.$$

- The action of $S_{k+1} \times S_{l+1}$ on the triangulations of $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ is induced by the action on simplices:

$$(p,q)T = \{(p,q)\sigma : \sigma \in T\}.$$

- The action of $S_{k+1} \times S_{l+1}$ on the vertices, simplices or triangulations defines an equivalence relation on each of them: two elements are equivalent if they can move to each other by an element of $S_{k+1} \times S_{l+1}$. We classify these sets by *orbits*, the equivalence classes.

For example, the triangulations T_1 and T_2 in Figure 2 moves to each other by $((1,2),e) \in S_2 \times S_2$. So does T_3 and T_4 for $((1,3),e) \in S_3 \times S_2$.

Remark 3.5
Simplices in $\mathcal{S}$ can be regarded as matrices in $\mathbb{R}^{k+1} \times \mathbb{R}^{l+1}$: a simplex $\sigma = \left\{ \begin{pmatrix} e_{i_1} \\ f_{j_1} \end{pmatrix}, \ldots, \begin{pmatrix} e_{i_{k+l+1}} \\ f_{j_{k+l+1}} \end{pmatrix} \right\}$ corresponds to a matrix with (i,j) element 1 when $\begin{pmatrix} e_i \\ f_j \end{pmatrix} \in \sigma$, and 0 otherwise. $S_{k+1} \times S_{l+1}$ acts as rearrangement of rows and columns of the matrix.

Definition 3.6 (equivalence on $\mathcal{S}$ and $2^{\mathcal{S}}$)

- We define the action of $S_{k+1} \times S_{l+1}$ on $\mathcal{S}$ by restricting to $\mathcal{S}$ its action on all simplices, which we defined in Definition 3.4. It defines an equivalence relation on $\mathcal{S}$. We notify this relation by $\sim$.
- The action of $S_{k+1} \times S_{l+1}$ on $2^{\mathcal{S}}$ is induced. Its action on $\mathcal{I}$, $\mathcal{M}$, $\mathcal{I}_{\mathrm{con}}$, $\mathcal{M}_{\mathrm{con}}$ and $\mathcal{T}$ is defined by restriction. The actions define equivalence relations on each of the sets. We use $\sim$ also for these relations.

Since $\mathcal{T} \subset \mathcal{M} \subset \mathcal{I} \subset 2^{\mathcal{S}}$ and $\mathcal{T} \subset \mathcal{M}_{\mathrm{con}} \subset \mathcal{I}_{\mathrm{con}} \subset \mathcal{I} \subset 2^{\mathcal{S}}$, $\mathcal{T}/\!\sim \subset \mathcal{M}/\!\sim \subset \mathcal{I}/\!\sim \subset 2^{\mathcal{S}}/\!\sim$ and $\mathcal{T}/\!\sim \subset \mathcal{M}_{\mathrm{con}}/\!\sim \subset \mathcal{I}_{\mathrm{con}}/\!\sim \subset \mathcal{I}/\!\sim \subset 2^{\mathcal{S}}/\!\sim$.

4 Enumerating triangulations—as a subset of maximal independent sets

The intersection graph of maximal dimensional simplices of a point configuration was the graph with these simplices the vertices and edges between two improperly intersecting simplices. Triangulations could be regarded as a subclass of the maximal independent sets of this graph (Section 2). Efficient algorithms to enumerate maximal independent sets are known [11], [16]. Joining the two, we propose a triangulation enumerating algorithm.

4.1 The general maximal independent set enumeration algorithm

First, we cite from [11] the algorithm we use for enumerating maximal independent sets. The algorithm is called the generalized Paull-Unger procedure with improvements by Tsukiyama, Ide, Ariyoshi and Shirakawa [16].

Let the set of vertices be $E = \{1,\ldots,n\}$ and c the independence testing time. We define $\mathcal{M}_j$ the family of independent sets that are maximal within $\{1,\ldots,j\}$. We construct $\mathcal{M}_j$ from $\mathcal{M}_{j-1}$, starting from $\mathcal{M}_0 = \{\emptyset\}$, to obtain $\mathcal{M}_n = \mathcal{M}$. For each I in $\mathcal{M}_{j-1}$, we test the independence of $I \cup \{j\}$. If it is independent, we add it to $\mathcal{M}_j$. If not independent, we add I and other maximal independent sets of $\mathcal{M}_j$ included in $I \cup \{j\}$. A set I' obtained in this way should be maximal in $I \cup \{j\}$. We use this fact reversely: first list up the maximal independent sets in $I \cup \{j\}$, and check if they are in $\mathcal{M}_j$. The algorithm elaborates to produce I' from a single I. We show it in Figure 3.

This computation performs a search on a tree. Nodes at level j are the members of $\mathcal{M}_j$ and the root is $\emptyset$. For each I in $\mathcal{M}_{j-1}$, the corresponding I' (possibly several) in $\mathcal{M}_j$ are its children. We start with the root $\emptyset$. Several searching methods are possible, but we take depth first search here.

Theorem 4.1 ([11])
The algorithm in Figure 3 enumerates all maximal independent sets in $O(nc'K + n^2cKK')$ time and $O(nK')$ memory. Here $K = \#\mathcal{M}$ and we suppose that in *Step 1*, for each $I \in \mathcal{M}_{j-1}$, at most K' sets I' are found in c' time.

4.2 Enumerating triangulations for arbitrary configurations of points

Theorem 4.2
If we have $E = \mathcal{S}$ with an arbitrary fixed order and an oracle that answers the previous or next simplex for a given one in unit time, and apply the algorithm in Figure 3 to enumerate all maximal independent sets of the intersection graph of $\mathcal{S}$, it works in $O(m\,\mathrm{time(intersect)}(\#\mathcal{S})^2\#\mathcal{M})$ time, proportional to the number of maximal independent sets, with the memory for the size of one triangulation. Here $m = \max_{I\in\mathcal{M}} \#I$ and time(intersect) is the time to judge if two simplices intersect properly.

Proof. For the independence test, or the test in *Step 1*, in actual we only have to check the intersection of a newly added simplex with the less than m current ones in I, so c and c' in Theorem 4.1 is computed in $m \cdot$ time(intersect) time. In *Step 1*, for each I in $\mathcal{M}_{j-1}$ the applicants we take are, if $I \cup \{j\} \in \mathcal{M}_j$, $I' = I \cup \{j\}$, and if not I and the set of simplices in $I \cup \{j\}$ except those intersecting with j, so $K' \leq 2$. Since we have the oracle mentioned above, we can traverse the search tree only with the information of our current independent set I and depth j, which is practically same as the size of a triangulation. Furthermore, since backtracking is easy, even with this restricted size of memory, the order of time complexity does not change. □

4.3 Enumerating triangulations for $\Delta_k \times \Delta_l$

The number of the simplices, the vertices of the intersection graph, increases exponential to the dimensions, but we cope with this by using their correspondence with spanning trees of an bipartite graph, and memorizing one simplex, or spanning tree, at once.

Theorem 4.3
For the point configuration $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ the algorithm in Figure 3 enumerates all maximal independent sets of the intersection graph of $\mathcal{S}$ in $O(\binom{k+l}{k}(k+l)k^{2l}l^{2k}\#\mathcal{M})$ time with the memory for the size of a triangulation.

Proof. The simplices in $\Delta_k \times \Delta_l$ correspond to spanning trees of the bipartite graph $K_{k+1,l+1}$ ([7, 7.3.D.]). We can generate such trees using a constant time per tree with small memory ([10], [13]), so the oracle mentioned in the Theorem 4.2 exists, which means that we do not have to memorize all the $(k+1)^l(l+1)^k$ simplices. By Lemma 3.1, $m = \binom{k+l}{k}$, by Lemma 3.2, $\#\mathcal{S} = (k+1)^l(l+1)^k$. Because judging the intersection of simplices for this product of simplices case can be transformed to a graph problem as in Lemma 3.3, this judgement can be done quickly: time(intersect) $= O(k+l)$. □

Applying this method to enumerate the orbits of triangulations would be a future work.

5 Enumerating triangulations—as a subset of connected independent sets

Triangulations $\mathcal{T}$ formed a subset of connected independent sets $\mathcal{I}_{\mathrm{con}}$ (Section 2). The algorithms in this section enumerate these connected independent sets which leads to the enumeration of triangulations. Enumerating maximal connected independent sets $\mathcal{M}_{\mathrm{con}}$, which comes between $\mathcal{T}$ and $\mathcal{I}_{\mathrm{con}}$ and whose gap with $\mathcal{T}$ is smaller, will be more efficient, but we have not succeeded yet.

We begin by explaining reverse search, a general enumeration algorithm which we use (5.1). Next we propose a triangulation enumeration method for arbitrary points (5.2) and apply this to the product of two simplices (5.3). Because of its symmetry, it is important to enumerate the classes of triangulations for the product of two simplices. We show how to enumerate classes of objects using reverse search (5.4), and enumerate the classes of triangulations with respect to symmetry (5.5).

5.1 Reverse search

Reverse search is a general technique for enumeration. It performs at the same output-size sensitive time as breadth first search (BFS) or depth first search (DFS), but requires memory only for twice the size of an object among the whole we want to enumerate. BFS and DFS needs output-size sensitive memory size, because we have to memorize all reached vertices. To enable this, in addition to the adjacency relation, which is necessary for BFS and DFS, parent-children relation is needful to execute reverse search. By this parent-children relation, we construct in implicit a spanning tree in the adjacency graph of the objects of enumeration, and perform a DFS on the tree. The word "implicit" means that we do not really compute the whole tree at once in the algorithm, but we do traverse it using local information [1], [2].

First we state the adjacency and parent-children relation for reverse search. This structure for reverse search is named "local search structure given by an A-oracle." We call it *a structure for reverse search* in this paper.

Definition 5.1
$(S, \delta, \mathrm{Adj}, f)$ is *a local search given by an A-oracle* if it suffices the followings.

- S is a finite set.

- $\delta \in \mathbb{N}$
- $\mathrm{Adj} : S \times \{1, \ldots, \delta\} \to S \cup \{\emptyset\}$. For any $a \in S$ and $i, j \in \{1, \ldots, \delta\}$ (1) $\mathrm{Adj}(a, i) \neq a$ and (2) if $\mathrm{Adj}(a, i) = \mathrm{Adj}(a, j) \neq \emptyset$ then $i = j$.
- $f : S \to S$ is the parent function: $f(a) = a$ or $\mathrm{Adj}(a, i)$ for some i. Furthermore, there exists a unique root vertex $r \in S$ such that $f(r) = r$. For any other vertex $a \neq r$, there exists $n \in \mathbb{N}$ such that $f^{(n)}(a) = r$.

S is the set to be enumerated. We set the maximum degree of the adjacency graph to δ. For each vertex $a \in S$ the adjacency function Adj returns its indexed adjacent vertex, or sometimes $\emptyset$ if the vertex has degree less than δ. This index is for use in the enumeration algorithm. We always assume that the adjacency relation is symmetric: if $\mathrm{Adj}(a, i) = b$ then $\mathrm{Adj}(b, j) = a$ for some j. The algorithm traverses the *reverse search tree*, a oriented spanning tree of S with an edge between a vertex and its parent. An example of this reverse search structure is shown in Figure 4. The information of δ, Adj, f and r is given to the reverse search algorithm, and it returns S as its output. The algorithm is presented in Figure 5.

Theorem 5.2 ([2, Corollary 2.3.])
The algorithm in Figure 5 works for the structure in Definition 5.1. The time complexity is $O(\delta(\mathrm{time(Adj)} + \mathrm{time}(f))\#S)$, where time(Adj) and time(f) are the time necessary to compute functions Adj and f. The memory required is the size of two objects in S.

5.2 Enumerating triangulations for arbitrary configurations of points

We propose an algorithm to enumerate all independent sets with a connected adjacency graph $\mathcal{I}_{\mathrm{con}}$ for arbitrary configurations of points. The triangulations $\mathcal{T}$ are included in this class. Since we use reverse search, the time complexity is proportional to the number of those independent sets, and memory of only twice the size of a triangulation is required.

Imagine the Hasse diagram of $2^{\mathcal{S}}$. The induced subgraph for the vertices $\mathcal{I}_{\mathrm{con}} \subset 2^{\mathcal{S}}$ is connected. We construct a spanning tree of this induced subgraph and enumerate the vertices $\mathcal{I}_{\mathrm{con}}$. We assume *a total order on* $\mathcal{S}$, and *the induced lexicographic order on* $2^{\mathcal{S}}$. Any order on $\mathcal{S}$ is acceptable. For each $I \in \mathcal{I}$ the *incrementable facets* are the facets of a simplex in I included in $\partial(\bigcup_{\sigma \in I} \mathrm{conv}(\sigma)) \setminus \partial(\mathrm{conv}(\mathcal{A}))$, where ∂P is the union of facets of P. They are the internal facets of I, where we try attach an adjacent simplex. Let m_{facet} be the maximum cardinality of incrementable facets for all $I \in \mathcal{I}_{\mathrm{con}}$. Fix a total order on the incrementable facets of each $I \in \mathcal{I}_{\mathrm{con}}$, and on points $\mathcal{A}$.

Theorem 5.3
For the following structure, reverse search enumerates all sets in $\mathcal{I}_{\mathrm{con}}$, in which all triangulations are included.

- $\delta = m_{\mathrm{facet}}\#\mathcal{A} + 1$
- $f(I) = \max_{\text{order on } 2^{\mathcal{S}}} \{I \setminus \{\sigma\} \in \mathcal{I}_{\mathrm{con}} : \sigma \in I\}$
- $\mathrm{Adj}(I, (i-1)\#\mathcal{A} + j) = \begin{cases} I \cup \{\tau_i \cup \{v_j\}\} & \text{for the } i\text{-th incrementable facet } \tau_i \text{ of } I \text{ and } j\text{-th point } v_j \in \mathcal{A}, \text{ if} \\ & \tau_i \cup \{v_j\} \text{ is a maximal dimensional simplex, and does not intersect} \\ & \text{with or is not equal to the simplices in } I. \\ f(I) & \text{if } (i-1)\#\mathcal{A} + j = \delta \\ \emptyset & \text{otherwise} \end{cases}$
- $\mathrm{Adj}(\emptyset, i) = \{\text{the } i\text{-th simplex in } \mathcal{S}\}$
- $r = \emptyset \in \mathcal{I}_{\mathrm{con}}$

The time complexity is $O((m_{\mathrm{facet}}\#\mathcal{A}+1)m\,\mathrm{time(intersect)}\#\mathcal{I}_{\mathrm{con}})$, where $m = \max_{I \in \mathcal{M}} \#I$ is the maximum cardinality of maximal dimensional simplices in $\mathcal{M}$ and time(intersect) is the time to judge if two simplices intersect properly. The memory required is the size of two triangulations.

Proof. Slight modifications are required for the case of the empty set. We define all simplices in $\mathcal{S}$ to be adjacent to $\emptyset$. The degree of $\emptyset$ can be larger than δ, but we can check that this does not change the whole time complexity. Clearly, reverse search for this structure works. We can keep the elements of I sorted, so time(f) is constant. To compute an adjacent vertex, we have to check if $\tau_i \cup \{v_j\}$ is a simplex, and if so, if it intersects properly with the existing simplices in I. The second test takes m time(intersect), and generally the first one can be done in time(intersect) time. Thus time(Adj) is $O(m\,\mathrm{time(intersect)})$. □

We do not know a non-trivial bound for the gap between $\#\mathcal{M}_{\mathrm{con}}$ and $\#\mathcal{I}_{\mathrm{con}}$. This gap becomes a loss for the time complexity, compared to the cardinality of triangulations, so the evaluation of this gap is important to decide the efficiency of this algorithm.

5.3 Enumerating triangulations for $\Delta_k \times \Delta_l$

Theorem 5.4
The algorithm in Theorem 5.3 enumerates all connected independent sets, in which all triangulations are included, in $O(\binom{k+l}{k}^2 (k+l)^2 kl \#\mathcal{I}_{\mathrm{con}})$ time with the memory of the size of two triangulations.

Proof. Recall Lemma 3.1. Less than $\binom{k+l}{k}$ simplices appear in an independent set, and each $(k+l)$-simplex has $k+l+1$ facets, so $m_{\text{facet}} \leq \binom{k+l}{k}(k+l+1)$. The cardinality of points $\mathcal{A}$ is $(k+1)(l+1)$. Judging whether two simplices intersect or not can be done in $O(k+l)$ time (Lemma 3.3). □

The figure $\binom{k+l}{k}$ here seems to be large, but this was the cardinality of maximal simplices appearing in a triangulation.

5.4 Reverse search of representatives

When a set of objects with a reverse search structure is classified such that its representatives satisfy a certain condition, we can perform a reverse search on the subset consisting of the representatives of the classes.

Definition 5.5
A reverse search structure and a classification of a set is *a structure for reverse search of representatives* if the parent of any representative element is a representative.

Theorem 5.6
For the structure for reverse search of representatives in Definition 5.5, we can enumerate the representatives by the following structure of reverse search. We use the notation in Definition 5.1 for the original reverse search structure of the elements.

- $R = \{a \in S : a \text{ is a representative}\}$ is the set we want to enumerate
- $\text{Adj}_R(a,i) = \begin{cases} \text{Adj}(a,i) & \text{if } \text{Adj}(a,i) \in R \\ \emptyset & \text{otherwise} \end{cases}$
- $f_R(a) = f(a)$

The time complexity is $O(\delta(\text{time(Adj)} + \text{time}(f) + \text{time(representative?)})\#R)$ where time(representative?) is the time to check if an element is a representative.

Proof. The reverse search tree of R becomes a subtree of the reverse search tree of S. The root of the original reverse search tree becomes the only element $a \in R$ with $f_R(a) = a$ i.e. the root of R. If $f_R(a) = b$ and $a \neq b$, a and b are adjacent in R. For any element in R we can reach the root by operating f_R several times, moving within R.

Checking if an applicant of an adjacent element $\text{Adj}(a,i)$ really is adjacent in R requires testing if it is a representative. So $\text{time}(\text{Adj}_R) = \text{time(Adj)} + \text{time(representative?)}$. Since the parent function is the same, $\text{time}(f_R) = \text{time}(f)$. Because the adjacency relation of the reverse search for representatives is a subset of the original adjacency, its degree is not larger than the original one. □

5.5 Enumerating classes of triangulations for $\Delta_k \times \Delta_l$ using symmetry

We apply Theorem 5.6 to the reverse search structure in Theorem 5.3 and enumerate the representatives of the classes of triangulations of $(\Delta_k \times \Delta_l, \text{vert}(\Delta_k \times \Delta_l))$ with respect to the symmetry of $S_{k+1} \times S_{l+1}$.

As stated above Theorem 5.3, we can introduce any order on S, and this is true also for the rest of this paper. But, from now on, we take the lexicographic order of its corresponding matrix representation. The matrix representation of simplices was dealt in Remark 3.5. We define a matrix (a_{ij}) to be smaller than (b_{ij}) in lexicographic order, if there exists (i',j') with (1) $a_{ij} = b_{ij}$ for any (i,j) with $i < i'$ or $i = i'$ and $j < j'$ and (2) $a_{i'j'} < b_{i'j'}$. It is easy to reformulate the following to the case of arbitrary ordering.

We define a label for each element in $\mathcal{I}_{\text{con}}$. By the lexicographic order on the labels, we introduce a total order on $\mathcal{I}_{\text{con}}$. The maximal element in this order will be taken as the representative for each class in $\mathcal{I}_{\text{con}}/\sim$.

Definition 5.7 (labeling on $\mathcal{I}_{\text{con}}$)
The *labeling* $l : \mathcal{I}_{\text{con}} \to (\mathbb{R}^{k+1} \times \mathbb{R}^{l+1})^*$, where Σ^* is the set of words on an alphabet Σ, is an alignment of simplices in a connected independent set $I \in \mathcal{I}_{\text{con}}$. We align the simplices by the order we visit in performing a breadth first search on the adjacency graph of I. We start from the maximum simplex in I, and when several simplices are adjacent to our current simplex, we visit from the larger ones.

For the connected independent set I in Figure 6 which is an example from $\Delta_2 \times \Delta_2$, the labeling is

$$l(I) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Clearly, l is injective.

From the order on the matrices $\mathbb{R}^{k+1} \times \mathbb{R}^{l+1}$, a lexicographic order on $(\mathbb{R}^{k+1} \times \mathbb{R}^{l+1})^*$ is induced. The order between words having different length does not matter in our case.

Definition 5.8 (order on $\mathcal{I}_{\text{con}}$)
We define a total order on $\mathcal{I}_{\text{con}}$ by the order of their labels in $(\mathbb{R}^{k+1} \times \mathbb{R}^{l+1})^*$.

Definition 5.9 (representative of $\mathcal{I}_{\text{con}}/\sim$)

For each class in $\mathcal{I}_{\rm con}/\sim$, we take the maximum element in this order as the representative.

Now we define the parent function f for reverse search of representatives.

Definition 5.10
The parent function f on the connected independent sets $\mathcal{I}_{\rm con}$ is, for $I \in \mathcal{I}_{\rm con}$ with label $l(I) = \sigma_1 \cdots \sigma_k$, $f(I) = \{\sigma_1, \ldots, \sigma_{k-1}\}$. We promise $f(\emptyset) = \emptyset$.

By executing breadth first search in parallel, we can prove the following lemma.

Lemma 5.11
If a connected independent set $I \in \mathcal{I}_{\rm con}$ is a representative for its class $[I] \in \mathcal{I}_{\rm con}/\sim$, $f(I)$ is the representative for its class $[f(I)] \in \mathcal{I}_{\rm con}/\sim$.

Using this property, we can enumerate all classes in $\mathcal{I}_{\rm con}/\sim$ by tracking their representatives.

Theorem 5.12
If we take δ, r and Adj as in the $\Delta_k \times \Delta_l$ case of Theorem 5.3, f as in Definition 5.10 and apply Theorem 5.6 to enumerate the representatives of the classes, we can enumerate $\mathcal{I}_{\rm con}/\sim$, which includes all representatives of the classes of triangulations $\mathcal{T}/\sim$, of $\Delta_k \times \Delta_l$. The time complexity is $O(\binom{k+l}{k}^3 (k+l) k^3 l^3 k!\, l!\, \#(\mathcal{I}_{\rm con}/\sim))$, and the required memory is of the size of two triangulations.

Proof. We can check that the reverse search structure in Theorem 5.3 with the modification of the parent function as in Definition 5.10 works. Since the new f satisfies the requirements in Definition 5.5, we can enumerate the representatives of $\mathcal{I}_{\rm con}/\sim$ by Theorem 5.6.

The time to check if an element of $\mathcal{I}_{\rm con}$ is a representative was time(representative?). There are $(k+1)!(l+1)!$ cases corresponding to $S_{k+1} \times S_{l+1}$, and for each case, we have to sort the vertices and execute breadth first search to calculate its label. This takes $O(\binom{k+l}{k}^2 (k+1)(l+1))$ time with comparison of two numbers as unit. Here, $\binom{k+l}{k}$ was the maximum cardinality of maximal dimensional simplices i.e. the maximum number of vertices in the adjacency graph. So time(representative?) $= O(\binom{k+l}{k}^2 (k+1)(l+1)(k+1)!(l+1)!)$. □

References

[1] David Avis & Komei Fukuda: *A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra, Discrete Comput. Geom.* **8** (1992), 295–313

[2] David Avis & Komei Fukuda: *Reverse Search for Enumeration, Discrete Appl. Math.* **65** (1996), 21–46

[3] Louis J. Billera, Paul Filliman & Bernd Sturmfels: *Constructions and Complexity of Secondary Polytopes, Advances in Math.* **83** (1990), 155–179

[4] Jesús A. de Loera: *Computing Regular Triangulations of Point Configurations,* 1994 `ftp://cam.cornell.edu/pub/puntos/help.ps`

[5] Jesús A. de Loera: *Nonregular Triangulations of Products of Simplices, Discrete Comput. Geom.* **15** (1996), 253–264

[6] Jesús A. de Loera, Serkan Hoşten, Francisco Santos & Bernd Sturmfels: *The Polytope of All Triangulations of a Point Configuration, Doc. Math.* **1** (1996), 103–119

[7] Israel M. Gelfand, Mikhail M. Kapranov & Andrei V. Zelevinsky: *Discriminants, Resultants and Multidimensional Determinants*, Birkhäuser, Boston 1994

[8] Israel M. Gel'fand, Andrei V. Zelevinskiĭ & Mikhail M. Kapranov: *Newton Polyhedra of Principal A-determinants, Soviet Math. Dokl.* **40** (1990), 278–281

[9] Hiroshi Imai & Keiko Imai: *Triangulation and Convex Polytopes,* in: "Geometry of Toric Varieties and Convex Polytopes", *RIMS Kokyuroku* **934** (1996), Research Institute for Mathematical Sciences, Kyoto University, 149–166 (in Japanese)

[10] H. N. Kapoor & H. Ramesh: *Algorithms for Generating All Spanning Trees of Undirected, Directed and Weighted Graphs, Lecture notes in Computer Science*, Springer-Verlag, 1992, 461–472

[11] E. L. Lawler, J. K. Lenstra & A. H. G. Rinnooy Kan: *Generating All Maximal Independent Sets: NP-Hardness and Polynomial-Time Algorithms, SIAM J. Comput.* **9** (1980), 558–565

[12] Joseph O'Rourke: *Art Gallery Theorems and Algorithms, International Series of Monographs on Computer Science* **3**, Oxford University Press, New York 1987

[13] A. Shioura, A. Tamura & T. Uno: *An Optimal Algorithm for Scanning All Spanning Trees of Undirected Graphs SIAM J. Comp.*, to appear

[14] Bernd Sturmfels: *Gröbner Bases of Toric Varieties, Tôhoku Math. J.* **43** (1991), 249–261

[15] Bernd Sturmfels: *Grobner bases and convex polytopes, University Lecture Series* **8**, American Mathematical Society, 1996

[16] Shuji Tsukiyama, Mikio Ide, Hiromu Ariyoshi & Isao Shirakawa: *A New Algorithm for Generating All the Maximal Independent Sets SIAM J. Comput.* **6** (1977), 505–517

[17] Günter M. Ziegler: *Lectures on Polytopes, Graduate Texts in Mathematics* **152**, Springer-Verlag, New York 1995

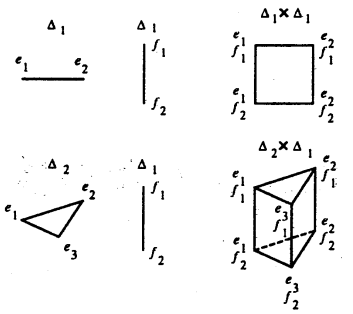


Figure 1: Products of simplices: $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$

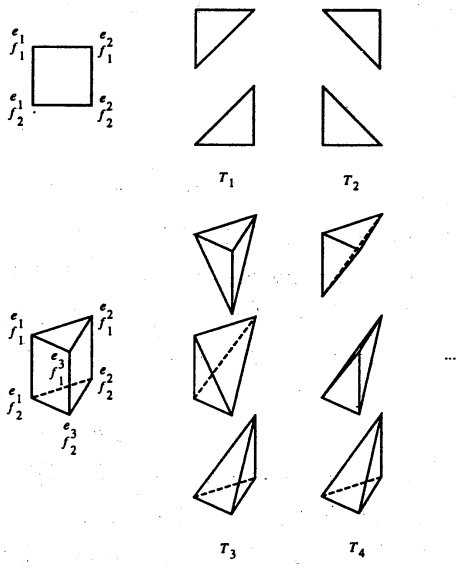


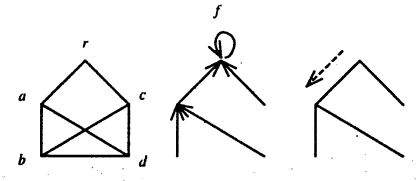
Figure 2: Triangulations for $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$

Step 1. For each $I \in \mathcal{M}_{j-1}$, find all independent sets I' that are maximal within $I \cup \{j\}$.

Step 2. For each such I', test I' for maximality within $\{1, \ldots, j\}$. Each set I' that is maximal within $\{1, \ldots, j\}$ is a member of $\mathcal{M}_j$, and each member of $\mathcal{M}_j$ can be found in this way. However a given $I' \in \mathcal{M}_j$ may be obtained from more than one $I \in \mathcal{M}_{j-1}$. In order to eliminate duplications we need one further step.

Step 3. For each I' obtained from $I \in \mathcal{M}_{j-1}$ that is maximal within $\{1, \ldots, j\}$, test for each $i < j$, $i \notin I$, the set $(I' \setminus \{j\}) \cup (I \cap \{1, \ldots, i-1\}) \cup \{i\}$ for independence. Reject I' if any of these tests yields an affirmative answer. (This step retains I' only if it is obtained from the lexicographically smallest $I \in \mathcal{M}_{j-1}$.)

Figure 3: The algorithm for enumerating maximal independent sets



$S = \{r, a, b, c, d\}$
$\delta = 3$

$\mathrm{Adj}(r,1) = a$
$\mathrm{Adj}(r,2) = c$
$\mathrm{Adj}(r,3) = \emptyset$

$\mathrm{Adj}(a,1) = r$
$\mathrm{Adj}(a,2) = b$
$\mathrm{Adj}(a,3) = d$

$\mathrm{Adj}(b,1) = c$
$\mathrm{Adj}(b,2) = d$
$\mathrm{Adj}(b,3) = a$

$\mathrm{Adj}(c,1) = r$
$\mathrm{Adj}(c,2) = b$
$\mathrm{Adj}(c,3) = d$

$\mathrm{Adj}(d,1) = c$
$\mathrm{Adj}(d,2) = a$
$\mathrm{Adj}(d,3) = b$

$f(r) = r$
$f(a) = r$
$f(b) = a$
$f(c) = r$
$f(d) = a$

Figure 4: An example of a set and its adjacency (top left), parent-children relation (top center), the reverse search tree (top right) and the structure in formulas (above)

ReverseSearch$(\delta, \mathrm{Adj}, f, r)$
$v := r \quad j := 0$
repeat
  while $j < \delta$ **do**
    $j := j + 1 \quad next = \mathrm{Adj}(v, j)$
    if $next \neq \emptyset$ **then**
      if $f(next) = v$ **then**
        $\{v := next \quad j := 0\}$
  if $v \neq r$ **then**
    $u := v \quad v := f(v)$
    $j := 0$
    repeat $j := j + 1$
    until $\mathrm{Adj}(v, j) = u$
until $v = r$ and $j = \delta$

Figure 5: The algorithm of reverse search

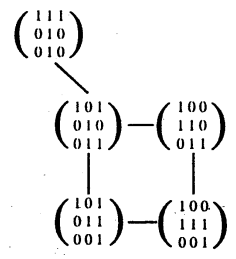


Figure 6: An adjacency graph of a connected independent set from the case for $\Delta_2 \times \Delta_2$