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

Fumihiko Takeuchi and Hiroshi Imai

Department of Information Science, University of Tokyo, Tokyo, Japan 113

Abstract. We propose algorithms to enumerate
(1) classes of regular triangulations in respect of symmetry for products of two simplices and
(2) all triangulations, regular or not, for arbitrary configurations of points.
There are many results for triangulations in two dimension, but little is known for higher dimensions. Both objects we enumerate in this paper are for general dimensions.
Products of two simplices, our first object, are polytopes rather simple, but their triangulations are not yet well understood. Since these polytopes are highly symmetric, counting all triangulations naively is inefficient: we may count the "same" triangulation many times. Our first algorithm enumerates the classes of regular triangulations, a subset of triangulations, with respect to the symmetry. We use reverse search technique, utilizing the symmetric structure of the polytope. This enables time complexity linear to the number of these classes, and space complexity of the size of several triangulations.
Even for the polytope of this product, a nonregular triangulation was found. Though algorithms to enumerate regular triangulations are studied well, no efficient algorithm to enumerate all triangulations, including nonregular ones, has been known. Our second algorithm handles this problem for arbitrary configurations of points. It formulates 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. We last apply this second algorithm to the polytope of the product.

1 Introduction

Gel'fand, Kapranov and Zelevinsky introduced the secondary polytope for point configurations in general dimensional space, and showed that its vertices correspond to regular triangulations [7], [8]. Using this property, regular triangulations can be enumerated by enumerating the vertices of that polytope. Billera, Filliman and Sturmfels studied the structure of this secondary polytope with relation to volume vectors, and analyzed the complexity computing the polytope. They also proposed the universal polytope, in which the vertices correspond to all

triangulations, but it has not resulted in a practical triangulation enumeration method [3].

Regular triangulations of the product of two simplices $\Delta_k \times \Delta_l$, where k and l are their dimensions, have relations with other branches of mathematics, such as Gröbner bases [17], [18]. This polytope is highly symmetric: it 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.

Masada, Imai and Imai proposed an algorithm to enumerate regular triangulations with output-size sensitive time complexity, which is same as de Loera's, using the memory only of the size for two triangulations [13], [14]. It uses a general technique for enumeration which is called reverse search, by Avis and Fukuda [1], [2].

Our first algorithm enumerates efficiently the classes with respect to symmetry of regular triangulations for the products of two simplices. Since this polytope is highly symmetric, as mentioned above, it is important to enumerate the classes. According to reverse search, we imaginary make a tree with the classes the vertices, and enumerate those vertices traversing the tree only using local information. The algorithm runs in output-sensitive time, i.e. in time proportional to the number of classes, and requires memory only of the size linear to a triangulation.

De Loera found a nonregular triangulation in $\Delta_3 \times \Delta_3$. So, it is important also to enumerate all triangulations, regular or not. Though there are some results [6], there is no efficient algorithm to enumerate all triangulations in dimension higher than two. Our second algorithm enumerates them for arbitrary configurations of points. We characterize triangulations as a subclass of maximal independent sets of the intersection graph of the maximal dimensional simplices, and apply a general maximal independent set enumeration algorithm. The time complexity is proportional to the number of maximal independent sets, the objects we really enumerate. When triangulations form a proper subset of the maximal independent sets, the gap between them becomes a loss. 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 dimension this does not happen, and in three dimension, we have Schönhardt's polyhedron (cf. [15, 10.2.1]) for example. However we are thinking that the gap may be small even in higher dimension. The memory required in this second algorithm is only about the size of two triangulations. Finally, we apply this to the case of the product of two simplices. The number of the simplices, the vertices of the intersection graph, increases exponential to the dimension,

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.

We begin by brief explanations of the concepts we use: regular triangulations and secondary polytopes (Section 2) and reverse search (Section 3). Next we derive some properties of products of two simplices (Section 4). Then we present our first result: enumeration of their regular triangulations (Section 5). We prepare some notations for enumerating all triangulations (Section 6). Finally we enumerate them as maximal independent sets (Section 7).

2 Regular triangulations and the secondary polytope

Regular triangulations form a subset of triangulations. They correspond to the vertices of a polytope, secondary polytope, which is determined uniquely by a configuration of points. Thanks to this property we can enumerate all regular triangulations applying a vertex enumeration method to the secondary polytope. Please refer to [3], [7], [8], [12] and [20] for discussions in detail.

Let $\mathcal{A} = \{a_1, \ldots, a_n\} \subset \mathbb{R}^{k-1}$ be a configuration of points, their convex hull $Q = \mathrm{conv}(\mathcal{A})$, with $\dim(Q) = k - 1$.

Definition 2.1 (triangulation) A simplicial complex T is a *triangulation* of $(Q, \mathcal{A})$ if its skelton $|T| = \cup T$ equals Q and its points are among $\mathcal{A}$.

Definition 2.2 (regular triangulation) A triangulation T of $(Q, \mathcal{A})$ is *regular* if there exists a vector $\psi : \mathcal{A} \to \mathbb{R}$ having the following property. For $P = \mathrm{conv}\{(a_1, \psi_1), \ldots, (a_n, \psi_n)\}$, and π the projection $\pi : \mathbb{R}^k \to \mathbb{R}^{k-1}$ with $\pi \begin{pmatrix} \boldsymbol{x} \\ x_k \end{pmatrix} = \boldsymbol{x}$, $T = \{\pi(F) : F \text{ is a lower face of } P\}$. Here F being a lower face means, $F = \{\boldsymbol{x} \in P : \boldsymbol{c}\boldsymbol{x} = c_0\}$, $\boldsymbol{c}\boldsymbol{x} \leq c_0$ valid for P, $c_{d+1} < 0$.

Definition 2.3 (volume vector) Let T be a triangulation of $(Q, \mathcal{A})$. The *volume vector* for T is a vector $\varphi_T : \mathcal{A} \to \mathbb{R}$ with $\varphi_T(\omega) = \sum_{\sigma \in T : \omega \in \mathrm{vert}(\sigma)} \mathrm{vol}(\sigma)$, where vol is the $\dim(Q)$-dimensional volume function, and $\mathrm{vert}(\sigma)$ is the set of vertices of σ.

Definition 2.4 (secondary polytope) The *secondary polytope* $\Sigma(\mathcal{A})$ of a point configuration $\mathcal{A}$ is the convex hull of points φ_T in $\mathbb{R}^{\mathcal{A}}$ for all triangulations T of $(Q, \mathcal{A})$.

Now we state that regular triangulations correspond to the vertices of the secondary polytope $\Sigma(\mathcal{A})$. The nonregular ones are mapped to the points other than the vertices, and their injectivity is not necessarily guaranteed. The vertices connected by an edge in the secondary polytope are "similar". Indeed, they can be modified each other by "flips". For the definition of flips, please consult the references above.

Theorem 2.5 ([7, Chapter 7. Theorem 1.7., Theorem 2.10.]) The secondary polytope $\Sigma(\mathcal{A})$ has dimension $n - k$, and its vertices correspond one-to-one

to the points φ_T of regular triangulations of $(Q, \mathcal{A})$. The edges are between vertices whose corresponding regular triangulations can be transformed each other by a flip.

3 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 those we want to enumerate. BFS and DFS needed output-size sensitive memory to memorize all reached vertices. To save memory, in addition to the adjacency relation, which is necessary for BFS and DFS, parent-children relation is needful for reverse search [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* here.

Definition 3.1 $(S, \delta, \mathrm{Adj}, f)$ is a *local search given by an A-oracle* if it suffices the followings. (1) S is a finite set. (2) $\delta \in \mathbb{N}$. (3) $\mathrm{Adj} : S \times \{1, \ldots, \delta\} \to S \cup \{\emptyset\}$. For any $a \in S$ and $i, j \in \{1, \ldots, \delta\}$, (i) $\mathrm{Adj}(a, i) \neq a$ and (ii) if $\mathrm{Adj}(a, i) = \mathrm{Adj}(a, j) \neq \emptyset$ then $i = j$. (4) $f : S \to S$ is the parent function: $f(a) = a$ or $\mathrm{Adj}(a, i)$ for some i. (5) There exists a unique root vertex $r \in S$: a vertex 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 enumerate. The maximum degree of the adjacency graph is 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. An example of this reverse search structure is shown in Fig.1. The information of δ, Adj, f and r is given to the reverse search algorithm, and the algorithm returns S as its output. Actually we do not need r, because we can find it by applying f several times to a vertex. The algorithm is presented in Fig.2.

Theorem 3.2 ([2, Corollary 2.3.]) The algorithm in Fig.2 works for the structure in Definition 3.1. The time complexity is $O(\delta\,(\mathrm{time}(\mathrm{Adj}) + \mathrm{time}(f))\,\#S)$, 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.

4 $\Delta_k \times \Delta_l$ 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 Fig.3 we show $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$ for example.

Following the notation in section 2, our objects are the triangulations of $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$, where $\mathrm{vert}(\Delta_k \times \Delta_l)$ are the vertices. Examples of triangulations are shown in Fig.4.

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

Lemma 4.1 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 next.

Lemma 4.2 The number of $(k+l)$-simplices of $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ 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 $K_{k+1,l+1}$ [5, Lemma 2.3.], so the time complexity follows.

Lemma 4.3 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 simplex, the shape of the product does not change. We formulate this symmetry.

Definition 4.4 (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 $\Delta_k \times \Delta_l$: $(p,q)\begin{pmatrix} \boldsymbol{e}_i \\ \boldsymbol{f}_j \end{pmatrix} = \begin{pmatrix} \boldsymbol{e}_{p(i)} \\ \boldsymbol{f}_{q(j)} \end{pmatrix}$.
- 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: $(p,q)\,\mathrm{conv}\{v_1, \ldots, v_m\} = \mathrm{conv}\{(p,q)v_1, \ldots, (p,q)v_m\}$.
- 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 the 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 Fig.4 moves to each other by $((1,2),e) \in S_2 \times S_2$. So does T_3 and T_4 by $((1,3),e) \in S_3 \times S_2$. The volume vectors can be regarded as matrices: $(\varphi_T \begin{pmatrix} \boldsymbol{e}_i \\ \boldsymbol{f}_j \end{pmatrix})_{ij} \in \mathbb{R}^{k+1} \times \mathbb{R}^{l+1}$. Those corresponding to the triangulations in Fig.4 are

$$\varphi_{T_1} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix} \quad \varphi_{T_2} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \quad \varphi_{T_3} = \begin{pmatrix} 2 & 2 \\ 3 & 1 \\ 1 & 3 \end{pmatrix} \quad \varphi_{T_4} = \begin{pmatrix} 1 & 3 \\ 3 & 1 \\ 2 & 2 \end{pmatrix}.$$

$S_{k+1} \times S_{l+1}$ acts on a volume vector φ_T as rearrangements of rows and columns of a matrix. Two regular triangulations T and T' are in the same orbit if and only if their volume vectors φ_T and $\varphi_{T'}$ are in the same orbit, since the correspondence between regular triangulations and volume vectors was one-to-one (cf. Theorem 2.5). We introduce an order on volume vectors, and define the representative of the orbits.

Definition 4.5 (lexicographic order on matrices) We take lexicographic order as the ordering on the matrices $\mathbb{R}^{k+1} \times \mathbb{R}^{l+1}$. A matrix (a_{ij}) is smaller than (b_{ij}) if for some (i_0, j_0), $a_{i_0 j_0} < b_{i_0 j_0}$, and for any (i, j) such that $i < i_0$ or such that $i = i_0$ and $j < j_0$, $a_{ij} = b_{ij}$.

Definition 4.6 (order on regular triangulations) We introduce a total order on regular triangulations by comparing their volume vectors as matrices.

Definition 4.7 (representative of orbits of regular triangulations) The *representative* of an orbit of regular triangulations is the maximum one.

In Fig.4, T_2 becomes the representative of the orbit $\{T_1, T_2\}$.

Lemma 4.8 Given a regular triangulation T, the representative element of its orbit can be computed in $O(l!\, k^2 l^2)$ time.

Proof. In order to choose the representative triangulation from the orbit of a given regular triangulation, we look for an element of $S_{k+1} \times S_{l+1}$ whose corresponding rearrangement maximizes the matrix of the volume vector φ_T. We check all of the $(l+1)!$ arrangements of columns. For each of them, the maximum can be obtained by sorting the rows. There are $k+1$ rows of length $l+1$. Comparing two integers in unit time, a comparison between two rows takes $O(l+1)$ time. So we can sort the rows in $O((k+1)^2(l+1))$ time. Hence the whole time complexity is $O((l+1)!\,(k+1)^2(l+1)) = O(l!\, k^2 l^2)$. □

5 Enumerating regular triangulations of $\boldsymbol{\Delta}_k \times \boldsymbol{\Delta}_l$

5.1 Enumerating all regular triangulations

We show the enumeration of regular triangulations of arbitrary configurations of points, and apply this to the case of $\Delta_k \times \Delta_l$. This takes time linear to the cardinality of the whole set of regular triangulations and requires memory linear to the size of a triangulation.

We define two triangulations to be adjacent if they can be modified along a circuit. For example, triangulations T_3 and T_4 in Fig.4 are adjacent, because they can be modified along the circuit consisting of the vertices of the upper two tetrahedra.

Definition 5.1 The structure of reverse search for regular triangulations of an arbitrary point configuration $(Q, \mathcal{A})$ is

- $S = \{\text{regular triangulation}\}$
- $\mathrm{Adj}(T, i) = (\text{the } i\text{-th regular triangulation which can be modified from } T$ along a circuit)
- $f(T) = \begin{cases} \mathrm{Adj}(T, i) & \text{if the largest regular triangulation } \mathrm{Adj}(T, i) \text{ among those} \\ & \text{adjacent to } T \text{ is larger than } T \\ T & \text{otherwise} \end{cases}$

The index i in the definition of $\mathrm{Adj}(T, i)$ is not of importance. Because regular triangulations correspond to the vertices of the secondary polytope $\Sigma(\mathcal{A})$, and lexicographic order is same as the ordering of the vertices by the inner product with a vector $(N^{(k+1)(l+1)}, N^{(k+1)(l+1)-1}, \ldots, N)$ with sufficiently large N, reverse search works. In fact, this is a geometric version of linear programming (cf. [20, Theorem 3.7.]). This algorithm for enumerating all regular triangulations for arbitrary configurations of points is from [14]. We apply this to the case of $\Delta_k \times \Delta_l$.

Theorem 5.2 ([14]) The structure of Definition 5.1 enables reverse search. For the case of $\Delta_k \times \Delta_l$ the time complexity is $O(\binom{k+l}{k}^2 k^3 l^3 \,\mathrm{LP}(kl, \binom{k+l}{k})(k + l + 1))\#\mathcal{R})$, where $\mathrm{LP}(n, m)$ is the time required to solve a linear programming problem with m strict inequalities constraints in n variables, and $\mathcal{R}$ is the set of regular triangulations of $\Delta_k \times \Delta_l$. The memory required is linear to the size of a triangulation.

5.2 Enumerating equivalence classes of a reverse search tree

The reverse search structure in Definition 5.1 enabled us to enumerate the regular triangulations (Theorem 5.2). However the objects we wanted to enumerate were the orbits, or classes, of regular triangulations. In this subsection, we show how to enumerate these classes of a reverse search tree.

Definition 5.3 A reverse search structure, total order and equivalence relation (which we denote by $\sim$) on a set is a *structure for reverse search of classes*, if

- for any element, its parent is the largest one among the elements adjacent and itself, and
- a adjacent to b and $c \sim a$ implies the existence of an element d adjacent to c and $d \sim b$, for any a, b and c.

Theorem 5.4 For the structure for reverse search of classes in Definition 5.3, we can enumerate the classes of elements by the following structure of reverse search. We use the notation in Definition 3.1 for the original reverse search structure of the elements, denote the class including an element a by $[a]$ and its maximum element, which we take as the representative, by $a_{\max}$.

- $S/\!\sim = \{[a] : a \text{ is an element}\}$ is the set we want to enumerate
- $\mathrm{Adj}([a], i) = \begin{cases} [\mathrm{Adj}(a_{\max}, i)] & \text{if } [\mathrm{Adj}(a_{\max}, i)] \neq [a_{\max}] \text{ and if} \\ & [\mathrm{Adj}(a_{\max}, i)] \neq [\mathrm{Adj}(a_{\max}, j)] \text{ for any } j < i \\ \emptyset & \text{otherwise} \end{cases}$

– $f([a]) = [f(a_{\max})]$

The time complexity is $O(\delta(\delta(\text{time(Adj)}+\text{time(representative)})+\text{time}(f))\#(S/\sim))$ where time(representative) is the time to compute the representative element of the class of an given element. The memory required is δ times the size of an element.

Proof. Remind that we supposed the adjacency relation to be symmetric: $\text{Adj}(a,i) = b$ implied $\text{Adj}(b,j) = a$ for some j. Then this also holds for the adjacency relation of the classes: if $\text{Adj}([a_{\max}],i) = [b_{\max}]$, $\text{Adj}(a_{\max},i) = c$ for some $c \sim b_{\max}$, then there exists $d \sim a_{\max}$ adjacent to $b_{\max}$. Two classes are adjacent if and only if there are adjacent elements from each of them. Any element of a class has an adjacent element in all the class-wise adjacent classes. Thus the degree for the reverse search of classes is not larger than the degree of the original reverse search.

The only class $[a]$ with $f([a]) = [a]$ is the class which includes the original root element. For other classes $f([a]) = [b]$ implies $a_{\max} < f(a_{\max}) \leq b_{\max}$. By applying f several times we can move from these classes to the root class. And, if $f([a]) = [b]$ and $[a] \neq [b]$, two classes $[a]$, $[b]$ must be adjacent.

During the enumeration, we have to "jump" to the representative element of the class we visited. The modification of the adjacency function avoids self and multiple adjacency. The time complexity of the adjacency function becomes $\delta(\text{time(Adj)} + \text{time(representative)})$, and the parent function $\text{time}(f) + \text{time(representative)}$. The adjacency function needs an extra object to be memorized. □

5.3 Using symmetry

Now we apply the reverse search for equivalence classes to the regular triangulations, and enumerate all orbits in time proportional to their cardinality.

Theorem 5.5 By applying the reverse search of classes in Theorem 5.4 to the original reverse search structure in Definition 5.1, we can enumerate the orbits of regular triangulations of $(\Delta_k \times \Delta_l, \text{vert}(\Delta_k \times \Delta_l))$. The time complexity is $O(\binom{k+l}{k}^2(k^3l^3\,\text{LP}(kl, \binom{k+l}{k}(k+l+1)) + k^2l^2l!)\#(\mathcal{R}/\sim))$, where $\mathcal{R}/\sim$ is the orbits of regular triangulations. The memory required is linear to the size of a triangulation.

Proof. Since the total order was the lexicographic order of volume vectors in matrix form and the equivalence relation was the rearrangement of rows and columns of the matrices (Definitions 4.4, 4.6, 4.7) the conditions in Definition 5.3 is satisfied. Hence the the reverse search of classes in Theorem 5.4 works. The time to compute the representative element is $O(l!\,k^2l^2)$ by Lemma 4.8. The $l!$ here appears in the whole time complexity, but since we are just finding the maximum arrangement of a small matrix (remind the instances we have to solve are for $k, l = 3$ or 4), practically this is not time consuming compared to solving LP. □

6 Preparations for enumerating all triangulations

Definition 6.1 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 $\mathcal{S}\}$
- $\mathcal{T} = \{\{\sigma_1, \ldots, \sigma_r\} \in 2^{\mathcal{S}} :$ a set of maximal dimensional simplices in a triangulation of $(\mathrm{conv}(\mathcal{A}), \mathcal{A})\}$

The maximal dimensional simplices in a triangulation $\mathcal{T}$ must not intersect, so $\mathcal{T} \subset \mathcal{I}$. We cannot add anymore maximal dimensional simplex to a triangulation in $\mathcal{T}$ without intersecting, so they are maximal. Thus $\mathcal{T} \subset \mathcal{M}$ [9]. In the next section we show an algorithm to enumerate $\mathcal{M}$, which leads to the enumeration of $\mathcal{T}$ the triangulations of $(\mathrm{conv}(\mathcal{A}), \mathcal{A})$. The difference of $\mathcal{T}$ and $\mathcal{M}$ becomes a loss. This sort of thing happens for Schönhardt's polyhedron (cf. [15, 10.2.1]). This is a concave polyhedron made by twisting a little bit a triangle of a prism. We cannot take anymore simplex with vertices among the six vertices from this polyhedron. The set made by the three tetrahedra fitting the outer concave part 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 $\Delta_k \times \Delta_l$.

7 Enumerating all triangulations—as a subset of maximal independent sets

7.1 Triangulations for arbitrary configurations of points

Triangulations can be regarded as a subclass of the maximal independent sets of the intersection graph of maximal dimensional simplices [9]. Efficient algorithms to enumerate maximal independent sets are known [11], [19]. We apply one of these algorithms to our case, and propose a triangulation enumerating algorithm. This algorithm handles arbitrary configurations of points.

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 [19].

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 independency 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 $I \cup \{j\}$. If I' is such set, it 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 5.

This computation performs a search on a tree. Nodes at level j correspond to members of $\mathcal{M}_j$ with the tree rooted by $\emptyset$. 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 7.1 ([11]) The algorithm in Fig.5 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.

Theorem 7.2 If we have $E = \mathcal{S}$ with an arbitrary fixed order and an oracle that answers in unit time the previous or next simplex for a given one, and apply the algorithm in Fig.5 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 with the memory for the size of one triangulation. Here $m = \max_{I\in\mathcal{M}} \#I$ is the maximum cardinality of maximal dimensional simplices in $\mathcal{M}$ and time(intersect) is the time to judge if two simplices intersect properly.

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 7.1 is computed in $m \cdot \mathrm{time(intersect)}$ 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, the order of time complexity does not change. □

7.2 Triangulations for $\Delta_k \times \Delta_l$

Theorem 7.3 For the point configuration $(\Delta_k \times \Delta_l, \mathrm{vert}(\Delta_k \times \Delta_l))$ the algorithm in Fig.5 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], [16], so the oracle mentioned in the Theorem 7.2 exists, which means that we do not have to memorize all the $(k+1)^l(l+1)^k$ simplices. By Lemma 4.1, $m = \binom{k+l}{k}$, by Lemma 4.2, $\#\mathcal{S} = (k+1)^l(l+1)^k$. Because judging the intersection of simplices for this product of simplices case can be transformed to a graph problem as in Lemma 4.3, this judgement can be done quickly: $\mathrm{time(intersect)} = O(k+l)$. □

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

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] Carl W. Lee: *Regular Triangulations of Convex Polytopes*, in: "Applied Geometry and Discrete Mathematics—The Victor Klee Festschrift" (Peter Gritzmann and Bernd Sturmfels, eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science **4**, Amer. Math. Soc. 1991, 443–456

[13] Tomonari Masada: *An Algorithm for the Enumeration of Regular Triangulations*, Master's Thesis, Department of Information Science, University of Tokyo, March 1995

[14] Tomonari Masada, Hiroshi Imai & Keiko Imai: *Enumeration of Regular Triangulations*, in: "Proceedings of the Twelfth Annual Symposium on Computational Geometry" Association for Computing Machinery (ACM), New York 1996, 224–233

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

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

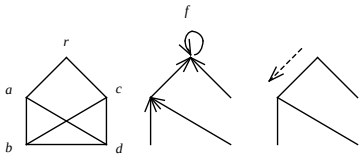
[17] Bernd Sturmfels: *Gröbner Bases of Toric Varieties*, *Tôhoku Math. J.* **43** (1991), 249–261

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

[19] 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

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

Fumihiko Takeuchi:
fumi@is.s.u-tokyo.ac.jp
Hiroshi Imai:
imai@is.s.u-tokyo.ac.jp



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

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

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

Fig. 1. 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 $j = \delta$

Fig. 2. The algorithm of reverse search

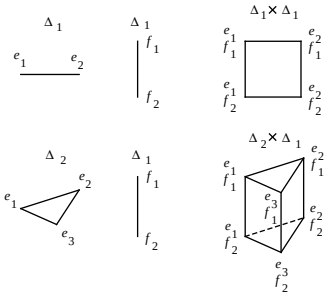


Fig. 3. Product of simplices: $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$

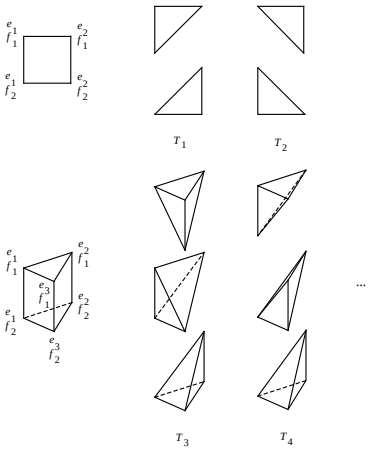


Fig. 4. Triangulations for $\Delta_1 \times \Delta_1$ and $\Delta_2 \times \Delta_1$

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 $I \in \mathcal{M}_{j-1}$.)

Fig. 5. The algorithm for enumerating maximal independent sets