Optimality and Integer Programming Formulations of Triangulations in General Dimension

Akira Tajima*

Tokyo Research Laboratory, IBM Research
tajima@trl.ibm.co.jp

Abstract. The properties of triangulations in two and three dimensions are computationally investigated by using integer programming (IP). Three IP formulations of triangulations are introduced, two based on the stable set problem, and the other based on the set partitioning problem. Some properties that are interesting from a theoretical or practical point of view are considered as objective functions for IP. Finally, some computational results are given. This approach allows three-dimensional triangulations to be treated in a flexible and efficient way, and has led to the discovery of some interesting properties of three-dimensional triangulations.

1 Introduction

Triangulation is a major topic in computational geometry, and there have been many studies of various approaches to triangulating a configuration of points, such as Delaunay triangulation, the minimum weight triangulation, and the lexicographic triangulation. Most of these studies have focused on triangulations in two dimensions, and less attention has been paid to triangulations in higher dimensions.

As a result of the recent advances in the performance of computers, however, the number of applications using triangulation in three dimensions is growing. For the finite element method (FEM), a three-dimensional polyhedron has to be divided into meshes. There are many meshing techniques: one approach is to distribute points inside the polyhedron and generate meshes by triangulating them. This approach is gaining popularity because of its simplicity and independence of dimensions [14]. Another major application is volume rendering, which is a method for visualizing the results of FEM, or semi-transparent objects such as clouds and flames. Unlike ordinary computer graphics methods that visualize the surface of materials, volume rendering meshes the three-dimensional space [15].

To contribute to these applications, the properties of triangulations in three dimensions should be further investigated. In three dimensions, some properties that are valid in two dimensions do not hold. Bistellar flips (Figure 1) show that the number of triangles is not constant in three dimensions. Schönhardt's polytope (Figure 2), which can be obtained by twisting a prism in such a way that the side faces become concave, shows that not all polyhedra can be triangulated in three dimensions. Some results on triangulatability are given in [13].

---

* also belongs to Department of Information Science, University of Tokyo

In two dimensions, Delaunay triangulation is optimal in many respects; for example, it maximizes the minimum angle, and minimizes the largest circumsphere and the largest minimum enclosing sphere. Many applications, such as meshing algorithms for FEM, depend on this characteristic. But in three dimensions, it only minimizes the largest minimum enclosing sphere [12], and does not necessarily lead to optimality in other respects. In other words, Delaunay triangulation is not so good in three or more dimensions. Thus, it is necessary to obtain a good triangulation for each individual application, namely, to optimize the triangulation.

In this paper we investigate the optimality of triangulations in two and three dimensions by using integer programming (IP) formulations. We first give three formulations in general dimension. Two of them are based on the stable set problem, and the other is based on the set partitioning problem. We then compare these formulations in size, and in other respects related to the efficiency of the procedure for solving the problem. We consider several objective functions in both two and three dimensions, which are interesting from a theoretical or practical point of view. We also give some computational results. Through the experiments, we found some interesting properties of triangulations in three dimensions.

As a preliminary step, we define some special terms. A *d-simplex* is a d-dimensional polytope that is the convex hull of $d+1$ affinely independent points. For example, a line segment, a triangle, and a tetrahedron correspond to a 1-simplex, a 2-simplex, and a 3-simplex, respectively. An *i-face* of a d-simplex is an i-simplex ($0 \leq i \leq d$) that is the convex hull of a subset of the vertices of the d-simplex. In particular, a $(d-1)$-face is called a *facet*. Two d-simplices *intersect* when the intersection is non-empty and is not a face of at least one of the two simplices. In this paper, especially for IP formulations, we consider the division of the convex hull of a point configuration $\mathcal{A}$ of n points in d-dimensional space using d-simplices, but we use the term *triangulation* for convenience. We assume that $\mathcal{A}$ is a configuration in general position (no $d+1$ points lie on the same $(d-1)$-dimensional hyperplane).

This paper is organized as follows. In Section 2, we give two kinds of IP formulations of triangulations in general dimension, then compare and investigate them. In Section 3, we consider objective functions. In Section 4, we give some computational results. Section 5 summarizes this paper.

2 Formulations

In this section, we give IP formulations by regarding triangulation as the stable set problem (Formulations 0 and 1), or the set partitioning problem (Formulation 2). Especially for the former, we consider two cases: that in which each d-simplex corresponds to an element (Formulation 0), and that in which each i-simplex ($0 \leq i \leq d$) corresponds to an element (Formulation 1).

2.1 Formulation 0: the stable set problem of d-simplices

In the intersection graph $G(V, E)$ of d-simplices, V corresponds to the set of d-simplices, and an arc is defined between two nodes when the corresponding two d-simplices intersect. A triangulation has no pair of simplices that intersect

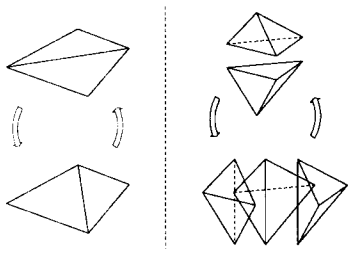

Fig. 1. Bistellar flips in 2 and 3 dimensions

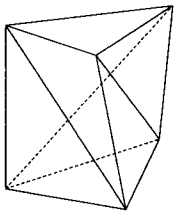

Fig. 2. Schönhardt's polytope

and thus corresponds to a stable set; further, it is maximal, because another d-simplex surely intersects with some d-simplices that constitute the triangulation. On the basis of this observation, Imai [10] gave an algorithm for enumerating triangulations in general dimension.

As mentioned in Section 1, not all polytopes can be triangulated in dimensions higher than two. Thus, there exist cases in which regions like Schönhardt's polytope remain untriangulated inside the convex hull, and the maximal stable set is not a triangulation. Therefore, we have to ensure that the stable set becomes a triangulation by imposing the condition that the sum of the volume v_i of d-simplex i is equal to the volume V of the convex hull.

$$\textbf{minimize} \sum_i c_i x_i \tag{1}$$

s.t.

$$x_i \in \{0,1\}$$

$$x_i + x_j \leq 1 \text{ (for } i,j \text{ s.t. } d\text{-simplex } i \text{ and } d\text{-simplex } j \text{ intersect)}$$

$$\sum_i v_i x_i = V$$

Here, checking the intersections among d-simplices is essentially redundant. For example, consider triangulations of the convex hull of four points on a plane. We have four triangles, and four pairs of triangles that intersect. On the other hand, the four intersections are derived from an intersection of two diagonals. In other words, if we explicitly handle lower-dimensional faces of d-simplices, the formulation can be more efficient.

2.2 Formulation 1: the stable set problem of i-simplices ($i \leq d$)

In the two-dimensional case, Kyoda et al. [11] formulated the minimum weight triangulation (MWT) problem as the stable set problem on the intersection graph of edges (1-simplices). MWT is a famous problem for which it is not known whether a solution can be obtained in polynomial time[9]. In two dimensions, the maximal stable sets of edges are always maximum, and the number of edges M is constant. Thus MWT is obtained by using the following formulation:

$$\textbf{minimize} \sum_i c_i x_i \tag{2}$$

s.t.

$$x_i \in \{0,1\}$$
$$x_i + x_j \leq 1 \text{ (for } i,j \text{ s.t. edge } i \text{ and edge } j \text{ intersect)}$$
$$\sum_i x_i = M$$

Extending to higher dimensions The above formulation (2) is valid only in two dimensions, and the cost and variables are assigned only to edges, so we need to extend it to higher dimensions and assign variables to d-simplices. We first consider the simplest example in three dimensions, a bistellar flip defined on the convex hull of five points. In the lattice-like structure in Figure 3, the i-th layer corresponds to $(i-1)$-simplices, and straight arcs among layers correspond to face relations of simplices in different dimensions. For example, edges (0,1), (0,4), and (1,4) are facets of, and necessary for triangle (0,1,4). Thus, the lattice is interpreted as a poset when we assign a variable to each simplex in such a way that the variable is equal to 1 if the simplex appears in the triangulation, and 0 otherwise. We will call the relations defined by the poset *lattice constraints.*

The lattice itself is independent of the configuration, whereas information on intersections depends on the configuration. In Figure 3, edge (0,4) and triangle (1,2,3) intersect. On the other hand, point (4) and tetrahedron (0,1,2,3) intersect in Figure 4.

When edge (0,4) and triangle (1,2,3) intersect, only one of them can appear in the triangulation. This can be represented as a constraint that the sum of the corresponding variables is less than or equal to 1. The following condition is necessary to make the formulation efficient:

Proposition 1. *All the vertices of simplices are assumed to be in general position. Then, two d-simplices intersect if and only if they have a k-face and a j-face, respectively, that intersect and satisfy $k + j = d$.*

We obtain a set of d-simplices that do not intersect by imposing the intersection constraints only on the pairs of simplices whose sum of dimensions is equal to d. and the lattice constraints. The volume constraint is again necessary, as in formulation (1), for the set to be a triangulation.

$$\textbf{minimize} \sum_i c_i x_i^d \tag{3}$$

s.t.

$$x_i^k \in \{0,1\}\ (0 \leq k \leq d)$$
$$x_i^k + x_j^{d-k} \leq 1\ (k\text{-simplex } i \text{ and } (d-k)\text{-simplex } j \text{ intersect},\ 0 \leq k \leq d)$$
$$x_i^k - x_j^{k+1} \geq 0\ (k\text{-simplex } i \text{ is a facet of } (k+1)\text{-simplex } j,\ 0 \leq k \leq d-1)$$
$$\sum_i v_i x_i^d = V$$

2.3 Formulation 2: the set partitioning problem

We use the term *chambers* for the minimal cells obtained by dividing the convex hull with all the possible $(d-1)$-simplices. Obtaining a triangulation can be

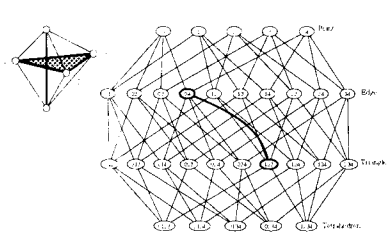

Fig. 3. Face relations among simplices

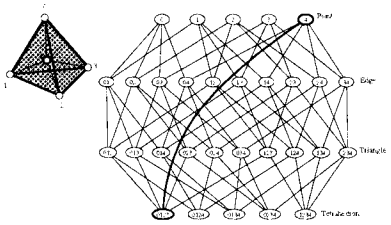

Fig. 4. Intersection of a 0-simplex and a 3-simplex

treated as an instance of the set partitioning problem by regarding each chamber as an element, and each d-simplex as a subset [1].

$$\begin{aligned} &\textbf{minimize } cx^d \\ \text{s.t.}\quad & x_i^d \in \{0.1\} \\ & Ax^d = 1 \quad A_{ij} = \begin{cases} 1 & d\text{-simplex } j \text{ contains chamber } i \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{4}$$

Cocircuit form constraints A large amount of geometric computation is necessary for handling chambers, and therefore we consider another type of constraint called *cocircuit form* constraints instead. Let e_k be a $(d-1)$-simplex. Two half-spaces $\mathcal{H}^+_{e_k}, \mathcal{H}^-_{e_k}$ are defined by a hyperplane $\mathcal{H}_{e_k}$ containing e_k. Let $e_k \cup \{a\}$ be a d-simplex t_i defined by an element a of the point set $\mathcal{A}$ and e_k. The following constraint holds for all $(d-1)$-simplices [7].

$$\sum_{t_i = e_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{e_k}} x_i - \sum_{t_i = e_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{e_k}} x_i = \begin{cases} \pm 1 & e_k \text{ is on the boundary} \\ & \text{of the convex hull} \\ 0 & \text{otherwise} \end{cases}$$

For $(d-1)$-simplices on the boundary, the above equation is the same as the set partitioning constraint for the adjacent chamber. For interior $(d-1)$-simplices, it corresponds to the difference of the constraints for the chambers on both sides.

De Loera et al. [7] showed that the cocircuit form constraints are sufficient to define the affine hull of the characteristic vectors of d-simplices that form a triangulation. From now on, we will call the formulation using the cocircuit form constraints *Formulation 2*.

2.4 Comparison of formulations

In this section, we compare Formulations 0, 1, and 2. Table 1 shows the size of each formulation. Although Formulation 1 uses fewer constraints than Formulation 0. Formulation 2 is still the most compact.

Table 1. Size of each formulation

	#variables	# constraints	# v/c
Formulation 0	$O(n^{d+1})$	$O(n^{2(d+1)})$	$O(1)$
Formulation 1	$O(n^{d+1})$	$O(n^{d+2})$	$O(1)$
Formulation 2	$O(n^{d+1})$	$O(n^{d})$	$O(n)$

#variables: total number of variables
#constraints: total number of constraints
#v/c: number of variables in a constraint

Polytopes We denote the polytope that corresponds to the linear programming relaxation of formulation 0, 1, and 2 by P_0, P_1', and P_2 respectively. The projection of P_1' to the subspace that corresponds to the variables for d-simplices is denoted by P_1. Then, the following relations hold among these polytopes:

Proposition 2. $P_0 = P_1$.

Lemma 3. *For the convex hull of $d+3$ points in d dimensions, there exist two triangulations that share no d-simplex. Further, there exists a d-simplex that lies inside the convex hull and does not belong to either of the two triangulations.*

Proposition 4. $P_2 \subset P_0$ $(n \geq d+3)$.

Proof outline: By focusing on a chamber, we can observe that both the volume and intersection constraints are satisfied if the set partitioning condition is satisfied. Thus $P_2 \subseteq P_0$.

From Lemma 3, we can configure the overlap of two triangulations, both of which are assigned a weight of $\frac{1}{2}$. We then perturb it slightly by using another simplex so that it still satisfies the constraints of Formulation 0. It then violates some chamber constraints, and $P_2 \subset P_0$. □

Spanning triangulations All the formulations above allow triangulations that do not use points inside the convex hull. However, triangulations are often assumed to use all the points: we call them spanning triangulations. In order to avoid non-spanning triangulations, we eliminate non-empty d-simplices in advance. For random point sets, the expected number of empty d-simplices is $O(n^d)$ [3], which is significantly smaller than the upper bound $\binom{n}{d+1}$ achieved when all the points are on the boundary of the convex hull. The lower bound is $\binom{n}{d-1}$ [3].

2.5 Cutting planes

As cutting planes for the problem of formulation (2), Kyoda et al. [11] applied clique cuts and odd-cycle cuts, both of which are well known for the stable set problem, and convex polygon cuts which use geometric information on triangulations. Formulations 0 and 1 are based on the stable set problem, and these cutting planes can cut off fractional solutions of their linear programming relaxations. In this section, we investigate the effectiveness of cutting planes with respect to Formulation 2.

The convex polygon cut is based on the property that the number of edges is constant for triangulations in two dimensions. Thus, this cut is valid only in two dimensions. Although it was given as a constraint on the number of edges, it can also be described as a constraint on the number of triangles, as follows:

Definition 5. Let $\mathcal{V}$ denote a configuration of points, C denote its convex hull, and M denote the cardinality of the spanning triangulations of $\mathcal{V}$. Further let x denote the incident vector of a spanning triangulation of a point configuration $\mathcal{A}$ ($\mathcal{A} \supseteq \mathcal{V}, \mathcal{A} \backslash \mathcal{V} \cap \mathcal{C} = \emptyset$), and let V denote the dimensions of x, which correspond to triangles that only have the elements of $\mathcal{V}$ as their vertices. The following inequality holds:

$$\text{Convex polygon cut} \quad \sum_{i \in V} x_i \leq M$$

Here we focus on an example in which Formulation 2 has a fractional solution [7]. A set of vertices $1, \ldots, 5$ of a regular pentagon and its center 0 constitutes a point set in two dimensions, and has 20 triangles and 16 triangulations. The vector $x \in \mathcal{R}^{20}$ with coordinates $x_{\{123\}} = x_{\{234\}} = x_{\{345\}} = x_{\{145\}} = x_{\{125\}} = x_{\{013\}} = x_{\{024\}} = x_{\{035\}} = x_{\{014\}} = x_{\{025\}} = \frac{1}{2}$, and all other coordinates $= 0$, satisfies the constraints of Formulation 2. The subgraph of the intersection graph of triangles induced by the nodes corresponding to the triangles with value $\frac{1}{2}$ and without vertex 0, constitutes an odd-cycle of size 5. The corresponding constraint

$$x_{\{123\}} + x_{\{234\}} + x_{\{345\}} + x_{\{145\}} + x_{\{125\}} \leq 2$$

cuts off x. Thus, odd-cycle cuts can be effective cutting planes. On the other hand, the convex polygon cut $\sum_{ijk} x_{ijk} \leq M (= 5)$ is satisfied by x and redundant. For convex polygon cuts, we state the following:

Conjecture 1 *Any convex polygon cut is valid for* P_2.

3 Objective Functions

In this section, we introduce some objective functions that are interesting from a theoretical or practical point of view. Objective functions can be categorized into two types. One is simple summation, which only requires us to solve IP on the basis of one of the formulations we introduced in Section 2. The other is bottleneck optimization, such as minimizing the maximum. If we try to solve such problems solely by using IP, the branch-and-bound procedure just goes progressively deeper and the lower bound never improves, and thus we cannot obtain a solution. Binary search is one way to avoid this numerical instability.

Minimize the maximum aspect ratio. In applications such as FEM, we should avoid flatness of triangles for the sake of computational stability and accuracy. The most straightforward index of the flatness is the aspect ratio (AR), which is the ratio of the radii of the circumscribing and inscribing spheres.

Maximize the minimum angle. Delaunay triangulation in two dimensions maximizes the minimum angle. In three dimensions, the angle itself has two varieties; the dihedral angle and the solid angle. There are several meshing algorithms that avoid some of the bad – too large or too small – angles (see [4], for example), but none for triangulating a point configuration. We can obtain a solution by specifying the min-max/max-min angle as the objective function.

Minimize the number of triangles. The number of triangles varies in three dimensions, and is an interesting objective function related to the following open problem:

Open Problem 1 *Is there a polynomial-time algorithm for triangulating an arbitrary convex polyhedron with the minimum number of tetrahedra ? [5]*

Minimize the weight. MWT in two dimensions is an interesting problem, as we mentioned in Section 2.2. In three dimensions, the weight is extended from the edge length to the surface area of the triangle.

4 Computational Experiments

This section gives the results of computational experiments based on Formulation 2. All the experiments were done on an IBM RS/6000 model 990, using the IBM Optimization Subroutine Library to solve mathematical programming problems. We also used the library CGAL [6] for handling geometric objects, and for obtaining Delaunay triangulation, we used the program Triangle [16] for two dimensions, and the program DeWall [8] for three dimensions.

4.1 Problem size and computational time

We measured the size and the CPU time for obtaining the minimum weight (spanning) triangulation of uniformly distributed point sets in two and three dimensions (Tables 2 and 3). The size of solvable problems is quite small in three dimensions. Further, if the point set is in convex position, all the triangles/tetrahedra are non-empty and the number of columns becomes larger, and the size of solvable instances becomes smaller.

We obtained integer solutions by means of the linear programming relaxations in all the cases, whereas in [11], based on the stable set problem, we obtained fractional solutions in many cases and applied cutting planes and the branch-and-bound method.

We experimented with several kinds of objective functions, including the ones in Section 3, and encountered fractional solutions after solving the relaxed linear programming problems only when the objective function was the cardinality, that is, when the problem was unweighted. Formulation 2 is fairly good practically, but still, the cutting planes we mentioned in Section 2.5 would be necessary to solve unweighted cases efficiently.

4.2 Optimality

Figures 5, 6, and 7 show the cardinality, weight and maximum aspect ratio, respectively, of triangulations in three dimensions. We used 10 point sets of cardinality 10, 20, and 30 randomly generated in a unit cube with different seeds. D, W, C, and AR in these figures represent Delaunay triangulation, the minimum weight triangulation, the minimum cardinality triangulation, and the triangulation with the minimum maximum aspect ratio, respectively.

From Figure 5 we can observe that Delaunay triangulation tends to contain more tetrahedra than the other triangulations. Figure 6 shows that Delaunay triangulation is not at all close to the minimum weight triangulation in three dimensions. Figure 7 shows that Delaunay triangulation avoids flat tetrahedra. There is still a gap between Delaunay triangulation and the optimal solution, which is obtained by the triangulation with the minimum maximum aspect ratio.

Table 2. Size and CPU time of the minimum weight triangulation in 2D

No. of points	10	20	30	40	80	160	240	320
No. of rows	45	190	435	780	3160	12720	28680	51040
No. of columns	71	482	1220	2328	11209	47837	109998	197883
IP (sec.)	0.24	0.72	1.78	4.25	45.91	727.99	3956.20	13801.49
Others (sec.)	0.31	1.89	6.21	14.24	131.35	1345.19	5283.15	13770.27

Table 3. Size and CPU time of the minimum weight triangulation in 3D

No. of points	10	20	30	40	50
No. of rows	120	1140	4060	9880	19600
No. of columns	194	3771	19638	57735	139982
IP (sec.)	0.42	9.08	148.27	6374.83	66798.32
Others (sec.)	0.69	10.29	48.67	145.28	372.84

IP: CPU time for solving the IP problem
Others: CPU time for the rest

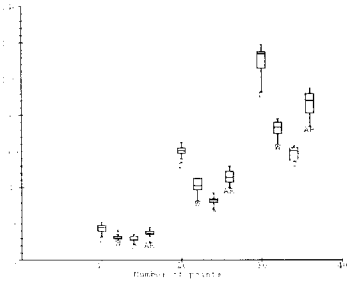

Fig. 5. Cardinality of triangulations

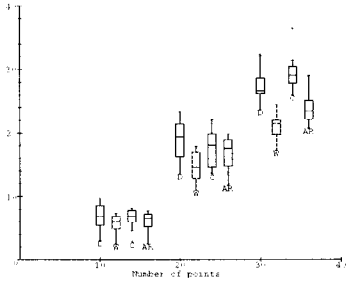

Fig. 6. Weight of triangulations

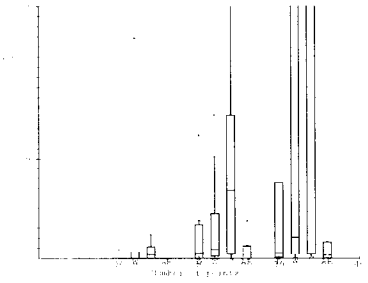

Fig. 7. Minimum maximum aspect ratio of triangulations

5 Conclusions and Remarks

In this paper, we have introduced IP formulations of triangulations of a point configuration in general dimension, and shown through computational experiments that the properties of triangulations differ significantly between two and three dimensions.

Another very important property of triangulations in three dimensions is that not all polyhedra can be triangulated [13]. Through the use of IP and maximization of the volume that can be triangulated, it may be possible to gain some insight into triangulatability. Among the formulations introduced in this paper, the one using cocircuit form constraints was the most efficient for triangulations of a point configuration. But it is still an open question which formulation is the most efficient for triangulations of non-convex polytopes.

Our experiments were based on linear programming, and it would be interesting to compare our approach with graph-based algorithms for the stable set problem ([2] for example).

References

1. T. V. Alekseyevskaya. Combinatorial bases in systems of simplices and chambers. *Discrete Mathematics*, 157:15–37, 1996.
2. E. Balas and J. Xue. Minimum weighted coloring of triangulated graphs, with application to maximum weighted vertex packing and clique finding in arbitrary graphs. *SIAM Journal of Computing*, 20(2):209–221, 1991.
3. I. Bárány and Z. Füredi. Empty simplices in Euclidian spaces. *Canad. Math. Bull.*, 30:436–445, 1987.
4. M. Bern, P. Chew, D. Eppstein, and J. Ruppert. Dihedral bounds for mesh generation in high dimensions. In *6th ACM-SIAM Symp. Discrete Algorithms*, pages 189–196, 1995.
5. M. Bern and D. Eppstein. Mesh generation and optimal triangulation. In *Computing in Euclidean Geometry*, volume 4 of *Lecture Notes Series on Computing*, pages 47–123. World Scientific, 2nd edition, 1995.
6. CGAL. http://www.cs.ruu.nl/CGAL/.
7. J. A. De Loera, S. Hosten, F. Santos, and B. Sturmfels. The polytope of all triangulations of a point configuration. *Documenta Mathematica*, 1:103–119, 1996.
8. DeWall. http://miles.cnuce.cnr.it/cg/swOnTheWeb.html.
9. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
10. H. Imai and K. Imai. Triangulation and convex polytopes. In *RIMS Kokyuroku*, pages 149–166. Research Institute for Mathematical Sciences, Kyoto University, 1996. (in Japanese).
11. Y. Kyoda, K. Imai, F. Takeuchi, and A. Tajima. A branch-and-cut approach for minimum weight triangulation. In *Proceedings of the 8th Annual International Symposium on Algorithms and Computation (ISAAC '97)*, volume 1350 of *Lecture Notes in Computer Science*, pages 384–393. Springer Verlag, 1997.
12. V. T. Rajan. Optimality of the Delaunay triangulation in R^d. In *Proc. 7th ACM Symp. Computational Geometry*, pages 357–363, 1991.
13. J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7:227–253, 1992.
14. K. Shimada. Physically-based automatic mesh generation. *Simulation*, 12(1):11–20, 1993. (in Japanese).
15. P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, 1990.
16. Triangle. http://www.cs.cmu.edu/~quake/triangle.research.html.

This article was processed using the LaTeX macro package with LLNCS style