Optimization of Geometric Triangulations using Integer Programming

Akira Tajima*†(tajima@trl.ibm.co.jp) Hiroshi Imai† (imai@is.s.u-tokyo.ac.jp)

(Extended Abstract)

Abstract

We investigate the optimization of triangulations of a point configuration mainly in the three-dimensional space using integer programming, based on computational experiments. The formulation using cocircuit form constraints introduced by De Loera et al. is very effective, but there still remain difficulties to solve practical instances of larger size, such that we obtain highly fractional solutions with certain objective functions. We introduce a column generation method to solve larger instances. To cope with fractional solutions, we apply binary search focusing on the cases of optimizing the bottleneck values, which enables us to successfully solve instances as large as ones with summation-style objective functions. We also introduce two kinds of new cutting planes which utilize the geometrical aspect of triangulations. Preliminary computational results show the effectiveness of these cuts.

1 Introduction

Geometric triangulation of a configuration of points has been studied in many fields from various viewpoints such as Delaunay triangulation, and most of these studies have focused on triangulations in two dimensions. Triangulation also has a lot of applications such as the finite element method (FEM) and computer graphics. Combined with recent advances in the performance of computers, they require techniques for and insights into triangulations in higher dimensions, especially in three dimensions.

On the other hand, it appeared that the properties of triangulations in higher dimensions are quite different from those in two dimensions. For example, bistellar flips (Figure 1) show that the number of triangles is not constant in three dimensions. Schönhardt's polytope (see [3], for example), which can be obtained by twisting a prism in a way that the side faces become concave, shows that not all the polyhedron can be triangulated in three dimensions.

As for optimization of triangulations, there are mainly two approaches; one based on computational geometry, and the other based on integer programming as follows.

Computational-Geometric approach 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 [3]. But in three dimensions, it only minimizes the largest minimum enclosing sphere [13], and does not necessarily lead to optimality in other respects. We observed some instances of Delaunay triangulation in three dimensions which have very flat tetrahedra on the boundary [14].

The bottleneck values, such as the minimum angle, are very important to guarantee the quality of the triangulation, and have been investigated intensively in computational geometry. Still, there are very little results in three dimensions, compared with those in two dimensions.

*Tokyo Research Laboratory, IBM Research.

†Department of Information Science, University of Tokyo.

Thus, optimizing triangulations, especially in three dimensions, has significance in both theory and applications, and algorithms in computational geometry can not cover all. The followings indicate the difficulty of problems related to the optimality of triangulations; obtaining the minimum weight triangulation (MWT) in two dimensions is a famous problem that is not known to be solvable in polynomial time or not [9], although there are now known algorithms using *skeletons* which find MWT in two dimensions efficiently in many cases [8]. The existence of a polynomial-time algorithm for triangulating a convex polyhedron with the minimum number of tetrahedra is also an open problem [4].

Integer-Programming approach One of the most promising methods for such kind of problems would be integer programming (IP), with objective functions such as the minimum cardinality, the maximum minimum angle, the maximum minimum aspect ratio, and the minimum weight[1]. We can regard triangulation as two kinds of IP problems; the stable set problem and the set partitioning problem. Masada et al. gave an algorithm for enumerating triangulations in general dimension by regarding a triangulation as a stable set of triangles [11]. Kyoda et al. obtained MWT in two dimensions by formulating a triangulation as a stable set of line segments [10]. We gave an improved formulation of triangulation as a generalized stable set problem by introducing lower dimensional simplices [14].

The set partitioning formulation of triangulation has been discussed in the literature of computational algebra. The smallest cells obtained by introducing all the facets of triangles are called *chambers*. Alekseyevskaya introduced a set partitioning formulation of triangles by regarding chambers as elements, and triangles as subsets [1]. De Loera et al. refined the formulation by introducing cocircuit form constraints [6].

We investigated these two kinds of formulations, and concluded that the formulation using cocircuit form constraints is the most compact to describe the problem, and the most efficient to solve the problem [14]. We, however, encountered several difficulties through computational experiments even with the currently best formulation.

Our contributions In this paper, we further investigate the optimization of triangulations as one of IP problems. First we present difficulties we encountered through computational experiments. One is that the size of the solvable instances is small. We can solve instances of at most 50 points in three dimensions, which is very large ($\binom{n-1}{3}$ rows and $\binom{n}{4}$ columns, as we will mention in section 2) as an IP problem, but still small if we consider applications. Another is that we obtain highly fractional solutions when the objective function is 1) a bottleneck value, which can be optimized by using algorithms in computational geometry if in two dimensions, or 2) non-weighted, which is also interesting from a Mathematical point of view. We also mention the degeneracy of the problem.

We then apply methods in integer programming in order to cope with them. We introduce a column generation method to solve larger instances. The geometrical interpretation of the dual problem is also given. To cope with the bottleneck cases, we introduce a binary search algorithm. Preliminary computational results show the effectiveness of the algorithm. Finally, for the non-weighted objective function, we introduce two kinds of new cutting planes which are based on the geometrical aspects of triangulations, and specific to the optimization of triangulations. Computational results indicate that these new cutting planes are effective, although odd-cycle cuts or clique cuts are effective only for very small instances.

This paper is organized as follows. In section 2, we review the formulations of triangulation; the set partitioning formulation, and the cocircuit form formulation. In section 3, we investigate computational experiences directly using the cocircuit form formulation, and present some difficulties

[1] The weight of a triangulation is defined as the total volume of the constituent $(d-1)$-simplices.

that we encountered. In section 4, we introduce a column generation method in order to solve larger instances. We also give the geometrical interpretation of the dual problem. In section 5, we introduce a binary search method to cope with a difficulty in optimizing the bottleneck value of triangulations. In section 6, we introduce two kinds of new cutting planes. Some computational results are also given. Finally, in section 7, we summarize this paper and give further subjects to be investigated.

2 Formulations

In this section, we review IP formulations based on the set partitioning problem.

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. Throughout this paper, 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).

Let x_i^k correspond to the i-th k-simplex, and c_i denote the cost of the i-th d-simplex.

2.1 The set partitioning problem

We call the minimal cells obtained by dividing the convex hull with all the possible $(d-1)$-simplices as *chambers*. Then obtaining the optimal triangulation can be treated as an instance of the set partitioning problem by regarding each chamber as an element, and each d-simplex as a subset [1].

$$\textbf{minimize } cx^d \tag{1}$$

s.t.

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

$$Ax^d = 1,\ A=(A_{ij}),\ A_{ij} = \begin{cases} 1 & d\text{-simplex } j \text{ contains chamber } i \\ 0 & \text{otherwise} \end{cases}$$

The number of rows of A corresponds to the number of chambers, and the number of columns of A corresponds to the number of d-simplices.

2.2 Cocircuit form constraints

De Loera et al. showed that, if $\mathcal{A}$ is in general position, the following constraints called *cocircuit form constraints* are equivalent to the chamber constraints in (1), and define the affine hull of the characteristic vectors of d-simplices that form a triangulation [6].

Let f_k denote a $(d-1)$-simplex. Two half spaces $\mathcal{H}_{f_k}^+, \mathcal{H}_{f_k}^-$ are defined by a hyperplane $\mathcal{H}_{f_k}$ containing f_k. Let $f_k \cup \{a\}$ denote a d-simplex t_i defined by f_k and an element a of the point set $\mathcal{A}$. The following constraint holds for all the $(d-1)$-simplices, and we can easily select $\binom{n-1}{d}$ constraints that constitute a basis[2] [6].

$$\sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}_{f_k}^+} x_i^d - \sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}_{f_k}^-} x_i^d = \begin{cases} 1 & f_k \text{ is on the boundary and oriented inside} \\ -1 & f_k \text{ is on the boundary and oriented outside} \\ 0 & \text{otherwise} \end{cases}$$

[2]The rank is proved to be $\binom{n-1}{d}$ by De Loera et al., if $\mathcal{A}$ is in general position

3 Computational Investigations into Simple Use of the Formulation

All the experiments were done on IBM RS/6000 model 990 with 512 MB of memory, using IBM Optimization Subroutine Library for solving mathematical programming problems. The primal-dual predictor-corrector method was used as the interior point method. We also used library CGAL [5] for handling geometrical objects, program Triangle [15] for obtaining Delaunay triangulations in two dimensions, and program DeWall [7] for obtaining Delaunay triangulations in three dimensions. Throughout the experiments, we limited triangulations to be spanning, in other words, to use all the points.

In [14], we investigated and compared the formulations in section 2 with formulations based on the stable set problem, and concluded that the formulation using cocircuit form constraints is the most compact to describe the problem, and the most efficient to solve the problem. Thus, the following discussions in this paper are based on the formulation:

$$\textbf{minimize } cx^d \tag{2}$$

s.t.

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

$$\ldots \text{cocircuit form constraints} \ldots$$

From the geometrical point of view, optimizing the bottleneck such as minimizing the largest aspect ratio, is often required. In such cases, we introduce another variable z as the upper bound[3]:

$$\textbf{minimize } z \tag{3}$$

s.t.

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

$$c_i x_i^d \leq z \quad (\text{for each } d\text{-simplex } i)$$

$$\ldots \text{cocircuit form constraints} \ldots$$

3.1 Observations

Through computational experiments based on the formulations above, we obtained the following observations:

1. Optimizing the bottleneck by using formulation (3) does not work. We obtain too many fractional variables (figure 3), the branch and bound procedure just goes down deeper, and the lower bound never improves [14].

2. For summation-style objective functions such as the minimum weight and the minimum sum of aspect ratio, we always obtain integer solutions just by solving the linear programming relaxation of formulation (2), except for the minimum cardinality triangulation, in other words, the non-weighted case [14]. Thus, practically we do not have fractional solutions with two dimensional instances.

3. The problem is highly primal degenerate. While the rank is $\binom{n-1}{d}$, the number of d-simplices in a triangulation, namely, the number of positive coordinates in a feasible solution is $O(n^{\frac{d+1}{2}})$. Table 1 shows the primal/dual degeneracy and the time consumed to solve the problem. In the weighted cases, the problem is only primal degenerate and the dual simplex method is very effective. On the other hand, in the non-weighted cases, it is also dual degenerate, and it takes time even with the dual simplex method. We can observe that the interior point method is robust against degeneracy.

[3] Maximization problems can be described with slight modifications

Table 1: Primal/dual degeneracy and the effectiveness of LP algorithms

# of points in 3D	20		30	
Cost	Weighted	Non-weighted	Weighted	Non-weighted
Primal degeneracy	814	919	3088	3221
Dual degeneracy	1	357	1	266
CPU-P (sec.)	25.57	40.95	3000.19	3916.51
CPU-D (sec.)	5.36	8.49	139.56	1065.45
CPU-I (sec.)	22.90	19.08	1396.91	1575.72

Primal degeneracy: # of basic variables with 0 value
Dual degeneracy: # of non-basic variables with 0 reduced cost
CPU-(P/D/I): CPU time consumed by the (primal/dual simplex and interior point) method

4. We obtain fractional solutions only in the non-weighted cases. Further, points inside the convex hull are closely related to the solution to be fractional. We randomly generated 50 instances of 20 points on a cylinder (in a convex position), and 20 points uniformly distributed in a cube, respectively. Although one third of the latter resulted in fractional solutions, we had fractional solutions with none of the former.

5. Addition of cutting planes can break the integrality of the solution. Even in the cases with weighted objective functions, a cut specifying the cardinality resulted in a highly fractional solution.

6. The size of the solvable instances is at most 50 points in three dimensions, and 320 points in two dimensions with the current computer performance and the naive use of formulation (2) [14].

4 Column Generation for Solving Large Instances

If we consider applications such as FEM, the size of instances currently solvable which we mentioned in section 3.1-6, is not enough. In this section, we investigate a column generation method in order to solve larger instances. Although the number of columns is not exponential ($O(n^{d+1})$), and the number of rows is not so small ($O(n^d)$), we believe there still be an advantage for considering only candidates that can improve the current solution.

In column generation methods, we start with a subset of columns, and iteratively add columns that have negative reduced costs and can improve the current solution until we can find no more candidates[1].

4.1 Geometrical interpretations of column generation

First we consider the dual of the linear relaxation of formulation (2). Each d-simplex has $d+1$ vertices, and we have $d+1$ pairs of a vertex and a facet. Let p_{ij} be 1 if j-th vertex of i-th d-simplex is on the positive side of the counterpart facet, and -1 otherwise. Also let $n(i,j)$ denote the suffix of the j-th facet of i-th d-simplex.

[1]To be precise, we have to switch to the branch and bound procedure if the final solution is not integral, and also generate columns at each node.

$$\textbf{maximize} \sum_i b_i y_i \tag{4}$$

s.t.

$$\sum_{j=1}^{d+1} p_{ij} y_{n(i,j)} \leq c_i \quad \text{(for each } d\text{-simplex } i\text{)}$$

Figure 2 shows a small sample in two dimensions. We have triangles t_1, t_2 as the current solution, t_3 and t_4 are not introduced yet, and $\tilde{c_1} = c_1 - (y_1 + y_2 + y_3) = 0, \tilde{c_2} = c_2 - (-y_3 + y_4 + y_5) = 0$ hold. We can safely set y_6 to zero, and assume $\tilde{c_3} = c_3 - (y_1 + y_4 + y_6) < \tilde{c_4} = c_4 - (y_2 + y_5 - y_6)$ without loss of generality. Then we can consider following four cases:

1. $\tilde{c_3} < 0, \tilde{c_4} < 0$
   Both t_3 and t_4 are added, and improve the cost of the triangulation ($c_1 + c_2 > c_3 + c_4$).

2. $\tilde{c_3} \geq 0, \tilde{c_4} \geq 0$
   Neither t_3 nor t_4 is added.

3. $\tilde{c_3} < 0, \tilde{c_4} \geq 0, c_1 + c_2 > c_3 + c_4$
   t_3 is added, y_6 is updated to be negative. Then $\tilde{c_4} < 0$. t_4 will be added in the next iteration.

4. $\tilde{c_3} < 0, \tilde{c_4} \geq 0, c_1 + c_2 \geq c_3 + c_4$
   t_3 is added, y_6 becomes negative, but still $\tilde{c_4} \geq 0$ holds. t_4 will not be added.

When we newly introduce a facet, the solution will never be improved until at least one d-simplex is introduced on both sides, as in case 1, 3. In formulation (4), adding a d-simplex k with $\tilde{c_k} < 0$ corresponds to adding a cut

$$\sum_{j=1}^{d+1} p_{kj} y_{n(k,j)} \leq c_k$$

which is violated by the current y. When d-simplices are introduced only on one side of $(d-1)$-simplex m which is not on the boundary, the constraints on y_m are one-sided, and the feasible region of y_m is unbounded. Then y_m can take arbitrary value and satisfy the constraints without changing the values of other coordinates, and the objective function never improves. Thus, we can say that 1) we should geometrically check the usage of $(d-1)$-simplices by d-simplices, and 2) applying bistellar flips is in practice an efficient way in the early stage of column generation.

4.2 Preliminary computational experiments

We did some computational experiments using small instances of 30 points in thee dimensions. As we mentioned in section 3.1-3, the problem is highly primal degenerate, and it causes two difficulties when we use simplex methods.

In each iteration, we can start from a primal feasible solution, and using the primal simplex method seems reasonable. But the results in table 1 shows it is often slower than solving with the dual simplex method from scratch.

Another difficulty is that the iteration does not easily terminate; the optimal solution was always obtained in early iterations, but still there were few candidates with slightly negative reduced costs, and it took hundreds of iterations to prove the optimality. Anbil et al. [2] noted that perturbing the right hand side of the constraints could reduce the difficulty, but we could not observe the improvements in our experiments.

Interior point methods are known to be robust against the degeneracy (e.g. [12]), and we can observe it in table 1. Actually, the iteration terminated soon after the optimal solution was obtained when we used the interior point method for solving the LP problems. Unfortunately, interior point methods requires more memory than simplex methods, and we cannot solve large instances yet.

5 Binary Search for Bottleneck Optimization

As we mentioned in section 3.1-1, when we optimize the bottleneck value of triangulations, introduction of the additional variable as in formulation (3) does not work. The branch and bound procedure just goes deeper and we cannot solve even small instances such as 20 points in a cube.

In this section, we introduce a binary search method. We assume the problem to be the minimization of the largest, such as the minimum maximum aspect ratio of the triangulation. Suppose there are N d-simplices, $t_0, \ldots, t_N$, and they are sorted in ascending order of their objective values. Let U denote the suffix of the d-simplex with the largest cost of a currently available triangulation, and L denote the suffix such that there is no triangulation only using a subset of $\{t_i | 0 \leq i \leq L\}$. The procedure is as follows:

1. *Initialization.* Set U to the suffix of the d-simplex with the largest value of a known triangulation (such as Delaunay triangulation). Set L to 0.
2. If $L + 1 = U$, terminate. Otherwise, set $Z = \frac{L+U}{2}$.
3. Solve the minimum-sum-cost problem with $\{t_i | 0 \leq i \leq Z\}$.
4. If the problem is infeasible, set $L = Z$. Otherwise, set U to the largest suffix of the d-simplex appeared in the solution.
5. Go to step 2.

The procedure terminates in $O(\log n)$ iterations, because $N \leq \binom{n}{d+1}$. In step 3, we solve a weighted summation-style problem. As we mentioned in section 3.1-2, we only have to solve the relaxed LP problem to obtain an integral solution in practice. Further, step 3 results in infeasible in most of the iterations, and we can stop the calculation in the early phase of the simplex method.

Table 2 shows the results of preliminary computational experiments. We used points randomly distributed in a cube as inputs, and applied a preprocess to remove non-empty tetrahedra. To obtain the minimum maximum aspect ratio, we had to solve the corresponding minimum-sum aspect ratio problem to the optimality only once for each instance. Further experiments would be necessary to investigate whether we should obtain the optimal solution or stop when we have found a feasible solution.

Thus, by using the above procedure, we can solve instances of bottleneck optimization as large as the instances with the summation-style objective function we can solve.

6 Cutting Planes

Solution can be fractional if the objective function is non-weighted. In this section, we introduce two cutting planes which are based on the geometrical aspects of triangulations.

Table 2: Binary search for the bottleneck value

# points	# iterations	# feasible	# variables	final U value
10	8	1	192	163
20	12	1	3732	3419
30	14	1	18799	18097
40	14	1	55241	52118
50	15	1	137757	135785

6.1 Small fractional examples

De Loera et al. introduced an fractional example in two dimensions [6]. A set of vertices $1,\ldots,5$ of a regular pentagon and its center 0 is a point set in two dimensions and has 20 triangles and 16 triangulations. $p \in \mathcal{R}^{20}$ with $p_{\{123\}} = p_{\{234\}} = p_{\{345\}} = p_{\{145\}} = p_{\{125\}} = p_{\{013\}} = p_{\{024\}} = p_{\{035\}} = p_{\{014\}} = p_{\{025\}} = \frac{1}{2}$ is a feasible solution of the linear relaxation of (2). The subgraph of the intersection graph of triangles induced by the nodes which correspond to the triangles with value $\frac{1}{2}$ and with(/without) vertex 0, constitute an odd-cycle of size 5, respectively. The corresponding constraints

$$p_{\{123\}} + p_{\{234\}} + p_{\{345\}} + p_{\{145\}} + p_{\{125\}} \leq 2 \tag{5}$$

$$p_{\{013\}} + p_{\{024\}} + p_{\{035\}} + p_{\{014\}} + p_{\{025\}} \leq 2 \tag{6}$$

cut off p. Thus, odd-cycle cuts can be effective cutting planes for formulation (2).

From a geometrical point of view, the number of triangles around a point inside the convex hull must be larger than 3, namely,

$$p_{\{013\}} + p_{\{024\}} + p_{\{035\}} + p_{\{014\}} + p_{\{025\}} + p_{\{012\}} + p_{\{023\}} + p_{\{034\}} + p_{\{045\}} + p_{\{015\}} \geq 3$$

This is also an effective cutting plane for the above example.

In three dimensions, we can easily consider the same situation as above; A set of vertices $1,\ldots,5$ of a regular pentagon on $x-y$ plane, and two points $6,7$ on z axis with a positive(/negative) coordinate, respectively. $q \in \mathcal{R}^{35}$ with $q_{\{1367\}} = q_{\{2467\}} = q_{\{3567\}} = q_{\{1467\}} = q_{\{2567\}} = q_{\{1236\}} = q_{\{2346\}} = q_{\{3156\}} = q_{\{1456\}} = q_{\{1256\}} = q_{\{1237\}} = q_{\{2347\}} = q_{\{3457\}} = q_{\{1457\}} = q_{\{1257\}} = \frac{1}{2}$ is a feasible solution, and can be cut off by odd-cycle cuts. Different from the case in two dimensions, we focus on the number of tetrahedra around the interior edge 6-7. The sum must be larger than 3 again, and q does not satisfy it.

6.2 $(d-2)$-face cuts

Although we observed that the cutting planes in the previous section can be effective, we can seldom find ones that cut off the current fractional solution, especially when the instance is large. In other words, odd-cycle cuts are not effective in practice. Figure 4 shows that, even in the case of 30 points, the intersection graph is very large, and the weight of each node is very small. The largest positive value is 0.56 (no 1s in the solution), most of the values are less than 0.1, and we can not find a cutting plane of the above-mentioned type that cut off the fractional solution.

The lower dimensional simplices, however, have fractional values too, and the situation changes if we consider the relative values. We introduce variable x_i^{d-2} corresponding to i-th $(d-2)$-simplex e_i as we did for the generalized stable set formulation of triangulations in [14]. Let S_i denote the set of all the d-simplices which contain e_i as a $(d-2)$-face. Let a_{ij} denote the angle[5] of j-th d-simplex around e_i, which are normalized so that the perigon is equal to 1.

Proposition 1 *The following equation holds for each $(d-2)$-face which is not on the convex hull:*

$$x_i^{d-2} = \sum_{j \in S_i} a_{ij} x_j^d \tag{7}$$

We consider the intersection graph of a set $N_i (\subseteq S_i)$ of d-simplices around e_i. Let C_i denote the size of the maximum independent set of N_i.

[5] The dihedral angle in three dimensions. In general dimension, the d simplex has two $(d-1)$-faces neighboring the $(d-2)$-face, and the angle corresponds to the angle between the normal vectors of the two $(d-1)$-faces.

Proposition 2 *The following inequality is valid for any triangulation*

$$\textbf{Upper bound cut} \quad \sum_{j \in N_i} x_j^d \leq C_i x_i^{d-2} \tag{8}$$

We can see that, in the two dimensional example above, the variable corresponding to the point inside the convex hull is equal to 1 ($= \frac{2}{5} \times \frac{1}{2} \times 5$), and (6) is a special case of this inequality.

The constraint around a point (or an edge in three dimensions) can be generalized as follows:

Proposition 3 *The following inequality is valid for any triangulation*

$$\textbf{Lower bound cut} \quad \sum_{j \in S_i} x_j^d \geq 3 x_i^{d-2} \tag{9}$$

Not only that the $(d-2)$-face cuts can cope with quite small fractional values, but also the number of candidates are small (at most $\binom{n}{d-1}$) and the calculation to look for violated constraints is not expensive.

6.3 Computational results

We investigated the effectiveness of the two kinds of $(d-2)$-face cuts with the instances of 20 and 30 points randomly generated in a unit cube, which gave fractional solutions under the non-weighted objective function. As the objective function is non-weighted, we can also add a cutting plane which cut off the fractional part of the objective value;

$$\sum_i x_i^d \geq \lceil C_f \rceil \quad C_f : \text{the current fractional solution.}$$

We first repeated solving the relaxed LP and adding cuts, until we could find no additional cuts, then transferred to the branch and bound procedure. We used the mixed integer programming module of the mathematical programming library for the branch and bound. Table 3 shows the results. Each row corresponds to the number of $(d-2)$-face upper bound cuts, the number of $(d-2)$-face lower bound cuts, the number of iterations of adding cuts, the lower bound of the objective function obtained after adding cuts, the number of fractional variables after adding cuts, and the number of nodes investigated during the branch and bound, respectively. Each column corresponds to the result with the pure branch and bound, $(d-2)$-face upper bound cuts, $(d-2)$-face lower bound cuts, both of the two kinds of cuts, and both of them plus the fractional cut, respectively.

We can observe that the lower bound cuts are especially effective, whereas the upper bound cuts do not work well. With the case of 20 points, combined with the fractional cut, the $(d-2)$-face cuts brought integer solutions. On the other hand, with the case of 30 points, the addition of the fractional cut resulted in the increase of $(d-2)$-face cuts, and also the increase of the branch and bound nodes. This may be related to the fact that the rows corresponding to $(d-2)$-faces are dense. Further investigations are necessary on it.

7 Conclusions and Remarks

In this paper, based on the computational experiments using the cocircuit form constraints, we further investigated the optimization of triangulations by combining the geometrical observations and IP approaches. A column generation method was introduced to solve large instances, and the geometrical interpretation of the dual problem was also presented. Binary search was applied to the optimization of the bottleneck value of triangulations. Based on the observation of small fractional examples, we introduced two kinds of cutting planes which utilized the geometrical aspect of triangulations.

Table 3: The effectiveness of the cutting planes

	# points	NONE	UB	LB	UB&LB	UB,LB&FRAC
# Cuts(UB)		-	7	-	5	3
# Cuts(LB)		-	-	20	20	8
# iterations	20	-	5	3	2	1
Lower Bound	(Optimal:38)	37.286	37.361	37.688	37.688	38.000
# fractional		131	187	80	85	0
# Nodes in B&B		12	15	4	4	1
# Cuts(UB)		-	0	-	1	17
# Cuts(LB)		-	-	61	62	102
# iterations	30	-	0	3	4	14
Lower Bound	(Optimal:61)	58.801	58.801	59.310	59.311	60.000
# fractional		1172	1172	1271	1256	1407
# Nodes in B&B		326	326	155	159	289

The effectiveness of our $(d-2)$-face cuts shows that the fractional solutions are related to the existence of lower-dimensional simplices inside the convex hull. As the formulation using cocircuit form constraints allow the weighted sum of triangulations, setting upper bounds with cutting planes such as odd-cycle cuts often result in just another fractional weights. Cuts giving lower bounds seem more effective. For the branch and bound procedure, we used a commercial IP package as it was, but controlling the branching, such as branching on lower-dimensional simplices, would reduce the number of nodes during the branch and bound.

We can currently solve instances of at most 50 points in three dimensions. Further investigations are necessary to enlarge the limit, say, to 100 points. Roughly speaking, as the number of simplices is $\binom{n}{4}$ in three dimensions, triangulation of 100 points might correspond to TSP of 10000 sites, and be a good milestone.

References

[1] T. V. Alekseyevskaya. Combinatorial bases in systems of simplices and chambers. *Discrete Mathematics*, 157:15–37, 1996.

[2] R. Anbil, R. Tanga, and E. L. Johnson. A global approach to crew-pairing optimization. *IBM Systems Journal*. 31(1):71–78, 1992.

[3] M. Bern. Triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 22. CRC Press, 1997.

[4] 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.

[5] CGAL. http://www.cs.ruu.nl/CGAL/.

[6] 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.

[7] DeWall. http://miles.cnuce.cnr.it/cg/swOnTheWeb.html.

[8] M. T. Dickerson and M. H. Montague. A (usually?) connected subgraph of the minimum weight triangulation. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages 204–213, 1996.

[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] 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.

[11] T. Masada, H. Imai, and K. Imai. Enumeration of regular triangulations. In *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, pages 224–233, 1996.

[12] J. E. Mitchell. Interior point methods for combinatorial optimization. In T. Terlaky, editor, *Interior Point Methods of Mathematical Programming*, chapter 11, pages 417–466. Kluwer Academic Publishers, 1996.

[13] V. T. Rajan. Optimality of the Delaunay triangulation in R^d. *Discrete & Computational Geometry*, 12(2):189–202, 1994.

[14] A. Tajima. Optimality and integer programming formulations of triangulations in general dimension. In *Proceedings of the 9th Annual International Symposium on Algorithms and Computation (ISAAC '98)*, Lecture Notes in Computer Science. Springer Verlag, 1998. to appear.

[15] Triangle. http://www.cs.cmu.edu/~quake/triangle.research.html.

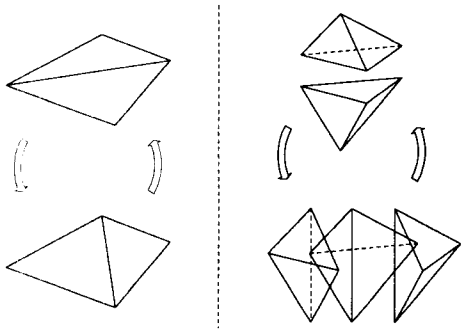

Figure 1: Bistellar flips in 2 and 3 dimensions

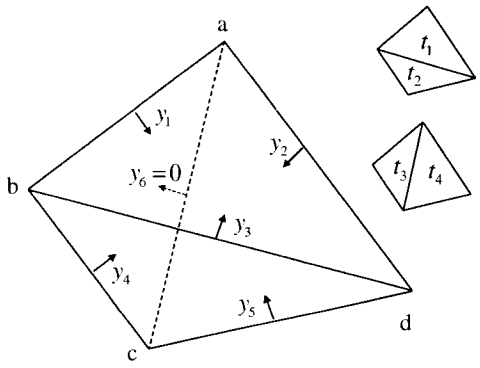

Figure 2: A small sample of column generation

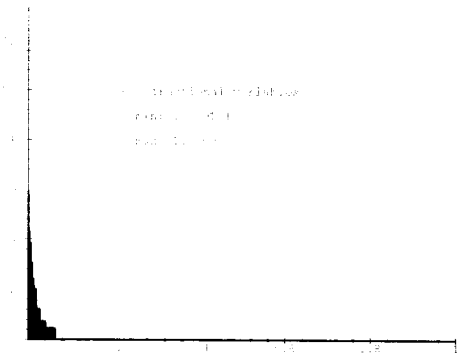

Figure 3: Distribution of positive values: 30 points in 3D, bottleneck

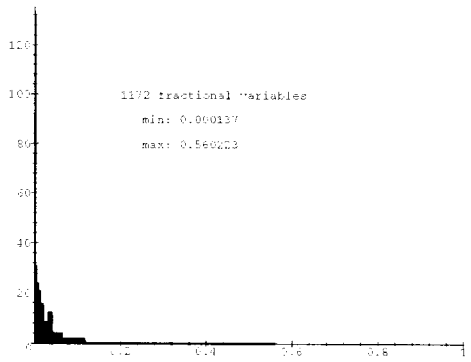

Figure 4: Distribution of positive values: 30 points in 3D, non-weighted