OPTIMIZING GEOMETRIC TRIANGULATIONS BY USING INTEGER PROGRAMMING

整数計画法による三角形分割の最適化

by

Akira Tajima

田島 玲

A Dissertation

Submitted to

The Graduate School of
The University of Tokyo
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Science
in Information Science

ABSTRACT

Geometric triangulation of a point configuration (simply "triangulation" in the following) has been studied in many fields from various viewpoints. It appeared that the properties of triangulations in higher dimensions are quite different from those in two dimensions, and most of these studies have focused on triangulations in two dimensions.

As for optimization of triangulations, computational geometry has been a mainstream so far. 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. For example, Delaunay triangulation is optimal in many respects in two dimensions, but in three dimensions, it does not lead to optimality in most respects.

On the other hand, 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. Thus, optimizing triangulations, especially in three dimensions, has significance in both theory and applications, and algorithms in computational geometry can not cover all.

In this thesis, we examine the optimality of triangulations by using Integer Programming (IP), another promising approach for optimization. Although the approach is independent of the number of dimensions, we mainly focus on three dimensional cases, because the gap of difficulties exists between two and higher dimensions and three dimensional cases are the simplest cases above the gap and also have several practical applications. We also give further investigations as an IP problem through computational experiments.

First, we focus on the formulations. There are several independent studies related to the optimization of triangulations. We generalize and examine them by classifying them into two groups, those can be reduced to (1) the stable set problem, and (2) the set partitioning problem.

Based on the investigations into formulations, we select one promising formulation and present some results and observations of computational experiments. One of the practical difficulties is that the size of the solvable instances is small. Another is that we obtain highly fractional solutions from the linear programming relaxation when the objective function is (1) a bottleneck value, which is popular in computational geometry, or (2) not weighted, which is also interesting from a mathematical point of view.

We then cope with the difficulties by investigating them as IP problems and utilizing methods in IP. We introduce column generation methods to solve larger instances. The degeneracy of the problem and the geometrical interpretation of the dual problem

are also discussed. To cope with the bottleneck cases, we introduce a binary search algorithm. For the non-weighted objective function, we devise two novel cutting planes based on geometrical properties of triangulation. Computational results indicate that these new cutting planes are practically effective, although traditional odd-cycle cuts or clique cuts are effective only for very small instances. We also observed that our cutting planes were facet-defining for a small example in three dimensions. The branch and bound procedure is still required because we obtain fractional solutions after adding the cutting planes, and we consider an effective way of branching by introducing lower-dimensional simplices as variables.

We finally consider several applications and derivatives of IP-based optimization of triangulations, some are of theoretical interest, and some are of practical interest, mainly in two or three dimensions. For each of them, we give some novel insights by providing the optimal solution. As straight forward applications of our IP-based approach, we look into the minimum weight triangulation and data dependent triangulations. In order to address the degeneracy and symmetry of the point configuration, we introduce the minimum and maximum cardinality triangulations of regular and quasi-regular polytopes. Dissection, a subdivision slightly different from triangulation and highly related to the degeneracy of the point configuration, is also discussed mainly focusing on the extension of the formulation. We then consider quadrilateral and hexahedral mesh generation, which are very important in industrial applications, by extending the formulation. We also give some computational results.

Our work is not only theoretical; we examine our ideas through computational experiments, and the above topics contain implementation issues and computational results respectively. In particular, we could actually optimize middle-sized problems both in theory and applications, which could not be solved by enumeration or other alternatives so far, under various measures. The optimal solutions that have never been obtained so far give us novel insights into problems and objective functions, such as how far we can *actually* improve the solution, not just a theoretical bound.

論 文 要 旨

点集合の凸包の三角形分割については、様々な見地から多くの研究がなされている。高次元での三角形分割については、二次元の場合とかなり特性が異なることがわかってきており、既存研究の多くは二次元に特化したものである。

三角形分割の最適性に関しては、従来計算幾何学によるアプローチが主流であった。最小角などボトルネックとなる値は三角形分割の品質を保証する上で重要な尺度であり、計算幾何学の分野で研究が重ねられているが、二次元に比べ、三次元ではあまり結果が得られていない。例えば、二次元においては Delaunay 三角形分割が多くの最適性を満たしているが、三次元ではほとんど成り立たない。

一方で、三角形分割には有限要素法やコンピューターグラフィックスなどの現実的な応用があり、近年の計算機能力の向上とあいまって、より高次元、特に三次元での三角形分割に関する技術や知見が求められている。このように、三角形分割の最適化、特に三次元のケースは、理論上、応用上共に重要であると同時に、計算幾何学的アプローチのみではカバーしきれない分野である。

本論文では、最適化の手法として整数計画法をとりあげ、この見地から三角形分割の最適性を考察する。手法自体は次元に依存しないが、三次元に主に着目する。これは、二次元と三次元以上の三角形分割の間には難度に大きな隔たりがあり、その向こう側では三次元が最も扱いやすく、また様々な応用が存在するためである。さらに、計算機実験を通して、整数計画問題としての分析を行う。

第一に定式化を検証する。三角形分割の最適化に関しては幾つか独立して既存研究がある。そこで、我々はこれらを一般化した上で (1) 安定集合問題、(2) 集合分割問題のそれぞれに還元される二つのグループに分類し、比較検討する。

次に、定式化に関する考察に基づき行った予備実験の結果を検証する。実際的な問題点の一つは、解ける問題の規模が小さいことである。もう一つの問題点は、線形緩和解が非常に細かな小数解となる場合があることで、計算幾何でしばしば取り上げられるボトルネック値の最適化や、数学的に興味深い重み無しの目的関数を採用したときに起こる。

そこで、我々はこれらに対処するため整数計画問題として分析した上で、整数計画の手法を応用する。より大規模の問題を解くために列生成法を適用する。数理計画問題としての退化や、双対の幾何的な意味合いについてもあわせて議論する。ボトルネックの最適化には二分探索法を適用する。重み無しの目的関数には、三角形分割の幾何的な特性に着目した二種類の新たな切除平面を紹介する。既存の clique, odd-cycle といった切除平面が現実的には意味を成さないのに対し、新たな二種類の切除平面が効果的であることを計算機実験により実証する。こうした切除平面を導入しても依然として分枝限定法は必要であり、低次元の単体に対応する変数を導入することで分枝限定法を効率化する手法も示す。

最後に、整数計画を用いた三角形分割の応用や拡張を、理論上あるいは実用上興味深い二、三次元において検討する。それぞれについて最適解を与えることにより、新たな知見が得られている。最適化した三角形分割の素直な適用として、辺長和最小三角形分割とデータ依存三角形分割を取り上げる。点集合の退化と対称性についての議論の材

料として(準)正多面体に着目し、最小／最大四面体数を与える。Dissectionは三角形分割を若干拡張した概念であり点配置の退化と大きな関わりがあるが、これについても定式化の拡張を中心に検討する。産業への応用上非常に重要な二次元での四角形分割、および三次元での六面体分割を想定し、定式化の拡張とともに計算機実験を行う。

多くの議論は理論のみでなく計算機実験による検証を伴っており、実装および実験結果に関する記述を含む。特に、理論および応用での中規模の問題に対し、実際に様々な尺度のもと最適解を与えた。これらは既存の方法では求められなかったものであり、理論上の限界ではなく実際にどの程度解を改善できるかといった、問題および目的関数に関する新たな知見を得ることができる。

Acknowledgements

I would like to thank Prof. Hiroshi Imai, Department of Information Science, University of Tokyo, who supervised this thesis. He supported me through advises, discussions and in many other ways.

Fairly large part of my work is based on and inspired by the results by Prof. Jesús A. De Loera in University of California-Davis, and discussions with him. I also thank Mr. Fumihiko Takeuchi for his suggestions, discussions, and information on computational geometry.

Discussions with Prof. Kokichi Sugihara in University of Tokyo on mesh generation technologies were very fruitful. He also invited me to the special interest group on mesh generation, where I obtained a lot of important comments and information.

Prof. Kenji Shimada in Carnegie Mellon University spared his time for discussion with me during the 8th international meshing roundtable that he chaired. I know he was really busy then, and appreciate his valuable comments on meshing applications and data dependent triangulations.

Porf. Akihisa Tamura in Kyoto University introduced me the generalized stable set problem, and showed a great interest in my work. It really encouraged me.

I would like to thank Dr. Kenichi Asai, Dr. Mary Inaba, and other members in Prof. Imai's laboratory.

I thank Dr. Takayuki Itoh for fruitful discussions, and for kindly supplying us with his experimental data. Mr. Keisuke Inoue also gave me a lot of comments and suggestions.

I finally thank my managers and colleagues in IBM Tokyo Research Laboratory. This work cannot be carried out without their understanding and support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Triangulation and its Applications

Geometric triangulation of a configuration of points has been studied in many fields, and mainly in computational geometry from various viewpoints 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. One reason is that it gets much more difficult when the number of dimensions is larger than two.

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 mesh elements. The mesh elements do not have to be triangles or tetrahedra. Quadrilaterals or hexahedrals are rather required in practice for numerical stability and accuracy [37]. But mesh generation using quadrilaterals or hexahedrals is much more difficult, and many of the practical systems with automatic mesh generation functions use triangular or tetrahedral meshes, especially when the region to be meshed has a complicated form such as mechanical parts.

There are many meshing techniques and they can be classified into two categories, direct and indirect methods [67, 55]. Direct methods generate meshes by dividing the region into small subregions, or paving the mesh elements. On the other hand, indirect methods first distribute points inside the region and then generate meshes by triangulating them. The latter kind of approaches are gaining popularity because of their simplicity and independence of dimensions [67]. Triangulation plays a very important role within these methods.

Another major application of triangulation 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 [69]. Especially for volume rendering, tetrahedral meshes are often preferred because (1) APIs and hardware for computer graphics are specialized for handling triangles or tetrahedra, and (2) the

linear interpolation is uniquely defined inside a triangle or a tetrahedron [43].

These applications require techniques for and insights into triangulations in higher dimensions, especially in three dimensions.

1.2 Triangulations in Higher Dimensions

We first 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 thesis, especially for integer programming formulations, we consider the subdivision of the convex hull of a point configuration $\mathcal{A}$ of n points in d-dimensional space into d-simplices that do not intersect with each other, but we use the term *triangulation* for convenience. Especially when all the elements of $\mathcal{A}$ are used in a triangulation, we say that the triangulation is *spanning*. A triangulation of polytope P corresponds to the triangulation of the vertex set of P. We say $\mathcal{A}$ is a configuration in a general position when no $d+1$ points lie on the same $(d-1)$-dimensional hyperplane.

When $\mathcal{A}$ is in a general position, minimal affinely dependent sets have cardinality $d+2$, and we call them *circuits*. A circuit $\mathcal{Z}$ can be splitted into two subsets $\mathcal{Z}^+$ and $\mathcal{Z}^-$ according to the signs in its affine dependency equation. Then convex hull of $\mathcal{Z}$ ($conv(\mathcal{Z})$) can be triangulated in two ways, using the d-simplices defined by $\{\mathcal{Z} \setminus \{a\}, a \in \mathcal{Z}^+\}$, or the d-simplices defined by $\{\mathcal{Z} \setminus \{a\}, a \in \mathcal{Z}^-\}$ [19]. A bistellar flip in d-dimensions is defined by the interchange between these two triangulations of $conv(\mathcal{Z})$.

As it appeared that the properties of triangulation in higher dimensions are quite different from those in two dimensions, we will review some of the important results in the following sections.

1.2.1 The cardinality of triangulations

Triangulations in two dimensions have an important property that "the number of triangles and the number of edges are invariant", but it does not hold for triangulations in dimensions higher than two. Figure 1.1 shows bistellar flips in two and three dimensions. In two dimensions, a bistellar flip corresponds to the choice of the diagonal to be used out of the two candidates, and the number of triangles is always two. On the other hand, in three dimensions, the convex hull of five points in a convex position can be triangulated in two ways, one into two tetrahedra, and the other into three tetrahedra. Namely, with this most simple example, the cardinality of triangulations in three dimensions can have multiple values.

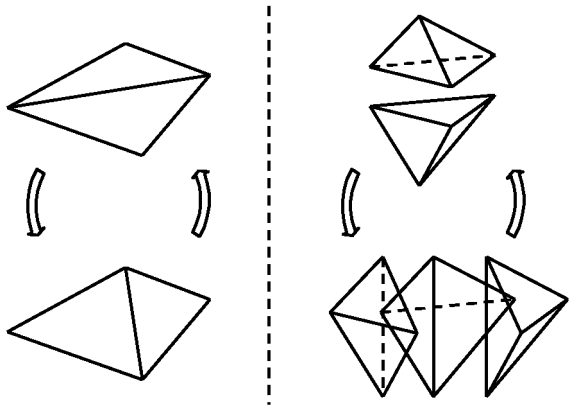

Figure 1.1: Bistellar flips in 2 and 3 dimensions

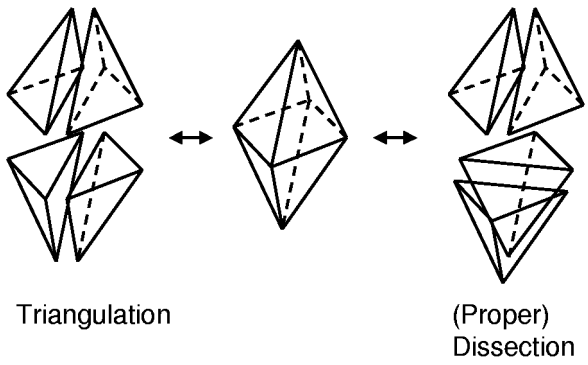

Figure 1.2: An example where a dissection differs from a triangulation

The upper and lower bound of the cardinality are important for purposes such as obtaining the volume of a high dimensional polytope [36]. For example in [18], the cases of three dimensional polytopes are intensively investigated. For general dimensions, lower bound and upper bound theorems on f-vectors are available [10].

Further, we can consider cases called *dissections*, where the constituent d-simplices do not share their faces with their neighbors, namely, it is a subdivision with d-simplices but is not a simplicial complex (Figure 1.2). When $\mathcal{A}$ is not in a general position, there are cases that are dissections but not triangulations. It is theoretically very interesting to investigate the cases where the upper and lower bound of the cardinality differ between triangulations and dissections [21].

1.2.2 Triangulatability of non-convex regions

When we consider applications such as FEM, it is very important to cope with non-convex regions, not only the convex hull of a point configuration. In two dimensions, it is well known that any region can be triangulated. Delaunay triangulation can also

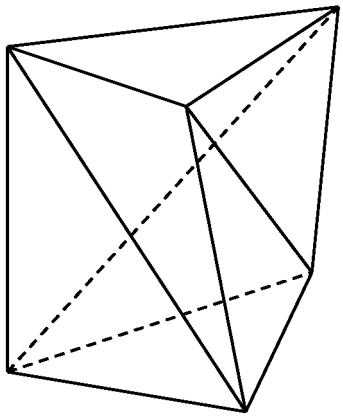

Figure 1.3: Schönhardt's polytope

be extended to *constrained Delaunay triangulation*, which aligns the edges specified to be used in the triangulation. It can be obtained in $O(n \log n)$ time, same as in non-constrained cases [7].

In three dimensions, not all the non-convex regions can be triangulated without additional (Steiner) points. Figure 1.3 shows Schönhardt's polytope, which can be obtained by twisting a prism so that three side faces become concave. Among the six vertices of this polytope, we cannot choose any four points that constitute a tetrahedron inside the polytope. Therefore this polytope cannot be triangulated without Steiner points. It means that greedy heuristics for triangulation in three dimensions can get stuck intermediately, with regions like Schönhardt's polytope left untriangulated.

Further theoretical results on triangulatability of polyhedra are given in [63],

Theorem 1.1 *[63] It is NP-complete to decide whether a given three-dimensional polyhedron can be triangulated without using additional Steiner points.*

Theorem 1.2 *[63] For any fixed integer $k > 0$, it is NP-hard to determine whether a given polyhedron can be triangulated with at most k Steiner points.*

1.2.3 Delaunay triangulation in higher dimensions

Delaunay triangulation in two dimensions is known to satisfy many optimalities such as maximization of the minimum angle, minimization of the maximum circumcircle, minimization the maximum minimum-enclosing circle [7]. Many applications, such as meshing algorithms for FEM, depend on these properties, for example in a way that generate mesh elements while maintaining Delaunay condition [28].

In three dimensions, however, it only proved to minimize the maximum minimum-enclosing sphere [58]. We actually observed that some instances of Delaunay triangulation in three dimensions had very flat tetrahedra on the boundary [71].

Further in two dimensions, we can obtain Delaunay triangulation from arbitrary triangulation by applying $O(n^2)$ Delaunay bistellar flips, but we can fall into a local minimum in three dimensions [59]. This means that local search methods cannot arrive at the optimal solution when we see triangulation from the view point of optimization. Recently Santos introduced an example in six dimensions with an isolated triangulation to which we can apply no bistellar flip [64].

1.3 Optimalities of Triangulations

Delaunay triangulation in two dimensions satisfies many optimalities and is almighty in a sense, and many applications depends on it. But we have seen in the previous section that Delaunay triangulation is not so good in three or higher dimensions. 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. Further, as not all the triangulations are connected by bistellar flips in higher dimensions, it is not so promising to design methods that start from Delaunay triangulation and apply flips to it.

Thus, optimizing triangulations, especially in three dimensions, has significance in both theory and applications, and algorithms in computational geometry can not cover all.

The following indicates the difficulty of problems related to the optimalities 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 [33], although there exist algorithms that can obtain the MWT in two dimensions efficiently in many cases, by identifying the subset of the MWT called *skeletons* [23]. See Section 5.1 for more details on the MWT. The existence of a polynomial-time algorithm for triangulating a convex polyhedron with the minimum number of tetrahedra is also an open problem [9].

1.3.1 Related studies

Integer programming approaches

One of the most promising methods to obtain the optimal triangulation 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] .

There are several studies on triangulation that are related to integer programming: Takeuchi et al. gave an algorithm for enumerating triangulations in general dimensions by regarding a triangulation as a stable set of d-simplices [72]. Kyoda et al.

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

obtained the MWT in two dimensions by regarding a triangulation as a stable set of line segments [44].

The set partitioning formulation of triangulation has been discussed in the literature of computational algebra. The term *chambers* is used to denote the smallest cells obtained by introducing all the $(d-1)$-simplices for dividing the convex hull. Alekseyevskaya introduced a set partitioning formulation of triangulation by regarding chambers as elements, and d-simplices as subsets [1]. De Loera et al. further investigated and refined the formulation algebraically, based on the oriented matroid theory [19].

Unfortunately, these studies were done independently, from different points of view.

Enumerating triangulations

Another way, and probably only the other way, to optimize triangulations in general dimensions is to enumerate all the triangulations, although optimization is not the only aim for enumeration.

Masada et al. focused on the enumeration of regular triangulations [47]. A triangulation is *regular* when it is obtained by lifting the vertices with some coordinate values in another dimension and taking the facets of the lower hull of the lifted vertices. Given a point configuration, there exists a polytope called *the secondary polytope* whose vertices correspond to regular triangulations [34]. Masada et al. applied a vertex enumeration algorithm to the secondary polytope in order to enumerate regular triangulations. There is an implementation of enumerating regular triangulations, *PUNTOS* by De Loera [57], which is also based on the secondary polytope.

However, there are non-regular triangulations even in two dimensions, and we need to enumerate *all* the triangulations for optimization purpose. Takeuchi et al.'s study [72] that we referred above also belongs to this category. *TOPCOM* is an implementation by Rambau that can enumerate all the triangulations [73]. To enumerate all the triangulations, it applies a depth first search by putting d-simplices together one by one. It can also enumerate the triangulations that are connected with each other by bistellar flips [60]. This class of triangulations includes regular triangulations, and is a subset of all the triangulations. It is interesting how further we have to do to cope with non-regular triangulations, and this implementation would be a good measure.

The largest drawback with enumeration is that it takes a lot of time – and sometimes space, too –, and is too inefficient to use for optimization. Actually the currently available implementations can only handle very small instances around 10-20 points in three dimensions.

1.4 Our Contributions

In this thesis, we examine the optimization of triangulations through the use of Integer Programming (IP). The approach is independent of the number of dimensions, but we

mainly focus on three dimensional cases, because the gap of difficulties of triangulation exists between two and higher dimensions as we have seen in Section 1.2, and three dimensional cases are the simplest cases above the gap and also have several practical applications.

We cover multiple aspects of IP-based optimization of triangulations. Starting from discussion on formulations, we investigate into our computational experiments and cope with the difficult cases by introducing techniques in IP. Finally we consider applications and derivatives of IP-based optimization of triangulations. Our work is not only theoretical; we examine our ideas through computational experiments, and the topics contain implementation issues and computational results respectively. In particular, we could actually optimize middle-sized problems both in theory and applications, which could not be solved by enumeration or other alternatives so far, under various measures. The optimal solutions that have never been obtained so far give us novel insights into problems and objective functions, such as how far we can *actually* improve the solution, not just a theoretical bound.

First, we focus on the formulations. There are several independent studies related to the optimization of triangulations as we referred in Section 1.3.1. We generalize and examine them by classifying into two groups: those can be reduced to (1) the stable set problem, and (2) the set partitioning problem. Further for the former, we give a more efficient formulation as an instance of the generalized stable set problem by focusing on lower dimensional simplices and thus removing the redundancy. We also consider several objective functions in both two and three dimensions, which are interesting from a theoretical or practical point of view.

Based on the investigations into formulations, we select one promising formulation, and give results and observations of computational experiments, including interesting examples for which the optimal solution is given for the first time. Our observations include the degeneracy of the problem, and the two main difficulties, namely, the small size of the soluble instances and the fractionality of the solutions. 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 Chapter 2 – as an IP problem, but still small if we consider applications. We obtain highly fractional solutions from the linear programming relaxation when the objective function is (1) a bottleneck value, which is popular in computational geometry, or (2) not weighted, which is interesting from a mathematical point of view.

We then cope with the difficulties by investigating them as IP problems and utilizing methods in IP. We introduce column generation methods to solve larger instances. The degeneracy of the problem and the geometrical interpretation of the dual problem is also discussed. To cope with the bottleneck cases, we introduce a binary search algorithm. For the non-weighted objective function, we devise two novel cutting planes based on geometrical properties of triangulations. Computational results indicate that these new cutting planes are practically effective, although traditional odd-cycle

cuts or clique cuts are effective only for very small instances. We also observed that our cutting planes were facet-defining for a small example in three dimensions. The branch and bound procedure is still required because we obtain fractional solutions after adding the cutting planes, and we consider an effective way of branching by introducing lower-dimensional simplices as variables.

We finally consider several applications and derivatives of IP-based optimization of triangulations, some are of theoretical interest, and some are of practical interest, mainly in two or three dimensions. For each of them, we give some novel insights by providing the optimal solution. As straight forward applications of our IP-based approach, we look into the minimum weight triangulation and data dependent triangulations. In order to address the degeneracy and symmetry of the point configuration, we introduce the minimum and maximum cardinality triangulations of regular and quasi-regular polytopes. Dissection, a subdivision slightly different from triangulation and highly related to the degeneracy of the point configuration, is also discussed mainly focusing on the extension of the formulation. We then consider quadrilateral and hexahedral mesh generation, which are very important in industrial applications, by extending the formulation. We also give computational results.

This thesis is organized as follows. In Chapter 2, we look into formulations of triangulation as IP problems. In Chapter 3, we give some results and observations of computational experiments. In Chapter 4, we cope with the difficult cases by investigating them as IP problems, and by utilizing methods in IP. In Chapter 5, we introduce some applications and extensions to our IP-based approach. Finally in Chapter 6, we summarize this thesis.

Chapter 2

Formulations

In this chapter, we look into formulations and objective functions. In particular, we consider two kinds of IP formulations; one as an instance of the stable set problem and the other as an instance of the set partitioning problem.

In Section 2.1, we review related studies that regard triangulation as an instance of the stable set problem. We then give an improved formulation. In Section 2.2, we review the results in computational algebra that regard triangulation as an instance of the set partitioning problem. In Section 2.3, we compare the formulations from several points of view, and finally conclude which one is the most suitable in practice. Finally in Section 2.4, we introduce several objective functions that are interesting from a theoretical or practical point of view.

Throughout this thesis, we consider the incidence vector $\boldsymbol{x^k}$ of k-simplices, of which each dimension corresponds to a k-simplex, and has value 1 when the k-simplex appears in the triangulation and value 0 otherwise. Let x_i^k denote the value of the i-th dimension. We will use t_i to refer to the i-th d-simplex, and f_i to the i-th $(d-1)$-simplex unless redefined explicitly. In particular, let v_i denote the volume of t_i, and c_i denote the value of the objective function for t_i.

2.1 Stable-Set-Problem Formulations

In this section, we investigate formulations of triangulation as instances of the stable set problem. There are several studies on triangulation that are related to the stable set problem. Takeuchi et al. gave an algorithm for enumerating triangulations in general dimensions by regarding a triangulation as a stable set of d-simplices [72]. We obtained the minimum weight triangulation (MWT) in two dimensions by regarding a triangulation as a stable set of line segments [44].

Based on these, we first introduce a formulation that regard triangulations as maximal stable sets of d-simplices. As it is not efficient enough, we give an improved formulation as an instance of the generalized stable set problem by introducing lower dimensional simplices [71].

2.1.1 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 in V when the corresponding two d-simplices intersect. A triangulation has no pair of simplices that intersect 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 (Figure 2.1). On the basis of this observation, Takeuchi et al. gave an algorithm for enumerating triangulations in general dimensions [72].

As we mentioned in Section 1.2, not all polytopes can be triangulated in dimensions higher than two. For example in three dimensions, there exist cases in which regions like Schönhardt's polytope (Figure 1.3) remain untriangulated inside the convex hull, and the maximal stable set of the tetrahedra is not a triangulation. Namely, not all the maximal stable sets correspond to triangulations.

Therefore, we have to ensure that the maximal stable set becomes a triangulation by imposing the condition that the sum of the volume v_i of d-simplex t_i is equal to the volume $V_{\mathcal{A}}$ of $conv(\mathcal{A})$.

Formulation (SS):

$$
\begin{aligned}
&\textbf{minimize } \boldsymbol{cx^d} \\
\text{s.t.}\quad & x_i^d \in \{0,1\} \\
& x_i^d + x_j^d \leq 1 \text{ (for } i,j \text{ s.t. } t_i \text{ and } t_j \text{ intersect)} \\
& \sum_i v_i x_i^d = V_{\mathcal{A}}
\end{aligned}
$$

Here, checking all 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 (Figure 2.2). In other words, if we explicitly handle lower-dimensional faces of d-simplices, the formulation can be more efficient.

2.1.2 The (generalized) stable set problem of i-simplices ($0 \leq i \leq d$)

We then consider the intersections among lower dimensional simplices. In the two-dimensional cases, we formulated the minimum weight triangulation (MWT) problem as the stable set problem on the intersection graph of *edges* (1-simplices) [44]. The MWT is a famous problem for which it is not known whether a solution can be obtained in polynomial time [33]. In two dimensions, the maximal stable sets of edges are always maximum too, and the number of edges M is invariant. Thus the MWT is obtained

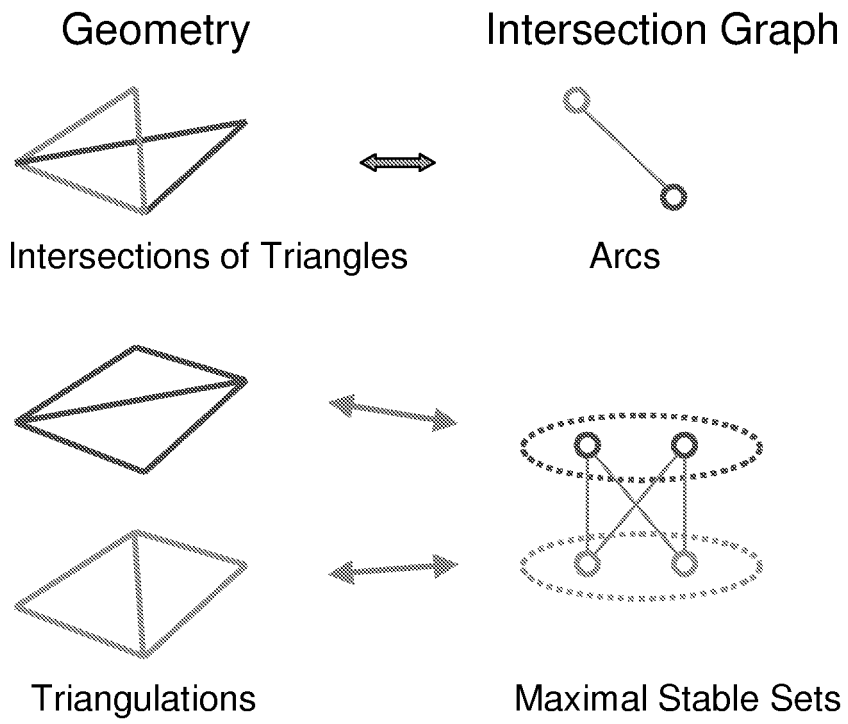

Figure 2.1: Triangulations $\subset$ the maximal stable sets in the intersection graph

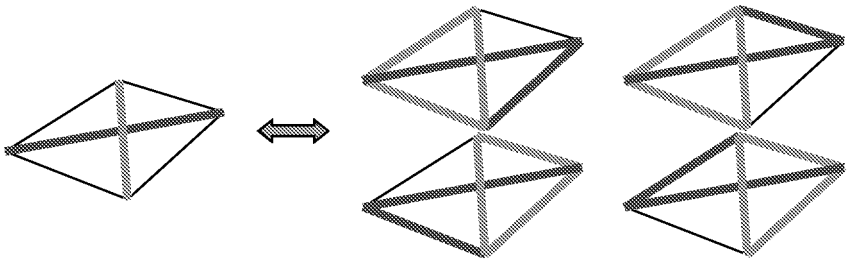

Figure 2.2: Four intersections of triangles are derived from an intersection of diagonals

by using the following formulation when $\bar{c}_i$ denote the length of the i-th edge e_i: [44]

$$\begin{aligned} & \textbf{minimize } \tilde{\boldsymbol{c}}\boldsymbol{x}^{\mathbf{1}} \qquad (2.1) \\ \text{s.t.} \\ & x_i^1 \in \{0,1\} \\ & x_i^1 + x_j^1 \leq 1 \text{ (for } i,j \text{ s.t. } e_i \text{ and } e_j \text{ intersect)} \\ & \sum_i x_i^1 = M \end{aligned}$$

As the number of edges M is invariant and all the maximal independent set of triangles correspond to triangulations in two dimensions, we can eliminate the constraint on the cardinality by modifying the objective function and changing it to the maximization problem. Then the problem becomes the pure maximal stable set problem.

In the following formulation, constant L is set to be larger than all the cost of the edges. Namely, $\forall i \;\; 0 < \bar{c}_i < L$.

$$\begin{aligned} & \textbf{maximize } \sum_i (L - \bar{c}_i) x_i^1 \qquad (2.2) \\ \text{s.t.} \\ & x_i^1 \in \{0,1\} \\ & x_i^1 + x_j^1 \leq 1 \;\; \text{(for } i,j \text{ s.t. } e_i \text{ and } e_j \text{ intersect)} \end{aligned}$$

The linear programming relaxation of the pure maximal stable set problem only has the values of $(0, \frac{1}{2}, 1)$ [52], and Nemhauser and Trotter showed that the variables with $(0,1)$ values had the same values in the original integer programming problem [53]. Based on this property, we can design an algorithm that gradually reduce the size of the problem by fixing the integer-valued variables [44].

Extending to higher dimensions

The above formulations (2.1) and (2.2) are valid only in two dimensions, and the cost and variables are assigned only to edges. Thus 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 a convex position. In the lattice-like structure in Figure 2.3, the i-th layer corresponds to $(i-1)$-simplices, and straight arcs among layers correspond to face relations of simplices among different dimensions. For example, edge {01}, {04}, and {14} are facets of, and necessary for triangle {014}. 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 2.3, edge {04} and triangle {123} intersect. On the other hand, point {4} and tetrahedron {0123} intersect in Figure 2.4. If we consider a degenerate point configuration, we can observe a intersection between a point and a triangle (Figure 2.5). Further, if we revisit the two dimensional cases

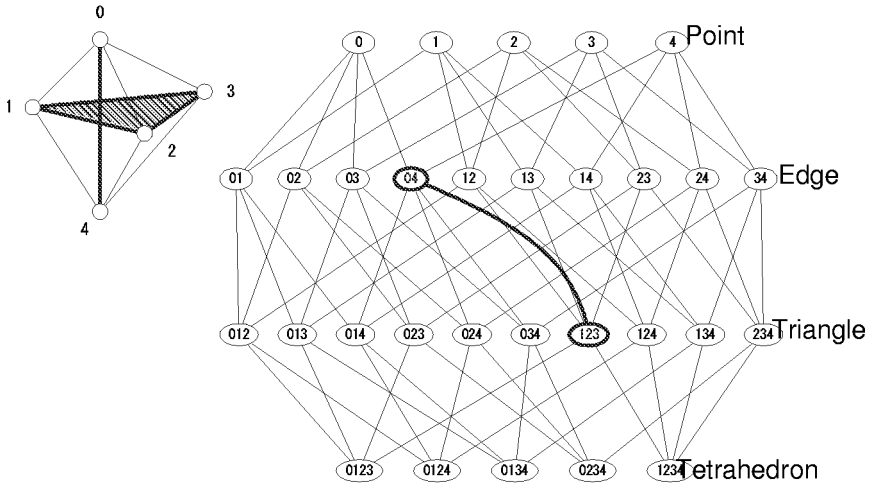

Figure 2.3: Face relations among simplices

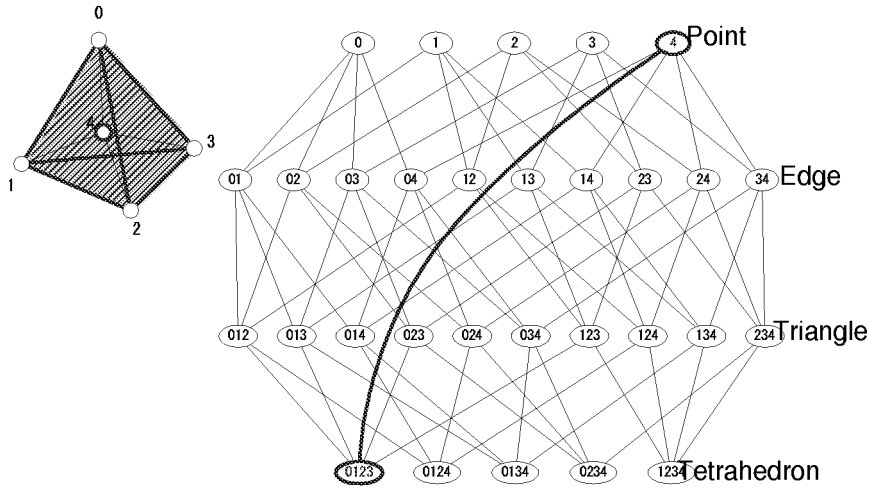

Figure 2.4: Intersection of a 0-simplex and a 3-simplex

(Figure 2.6), we can see that in Formulation (2.1) and (2.2) we just considered the special cases where intersections occur only among edges (1-simplices).

When edge {04} and triangle {123} intersect (Figure 2.3), 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. Suppose edge {04} does not appear in the triangulation, then it is derived from the lattice constraints that all the simplices that have edge {04} as faces, such as triangle {014} and tetrahedron {0124}, do not appear in the triangulation, either. Thus, by combining the lattice constraints and the intersection constraints among simplices of various dimensions, we can reduce the number of constraints compared with Formulation (SS).

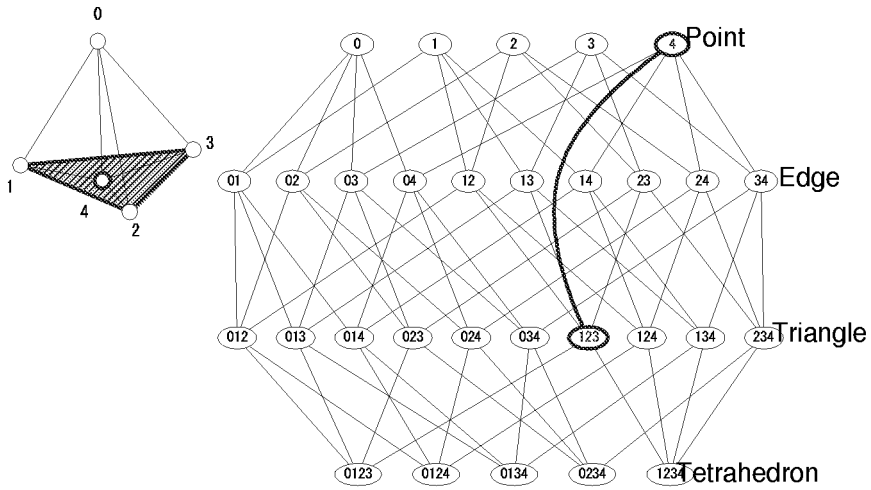

Figure 2.5: Intersection of a 0-simplex and a 2-simplex, a degenerate case

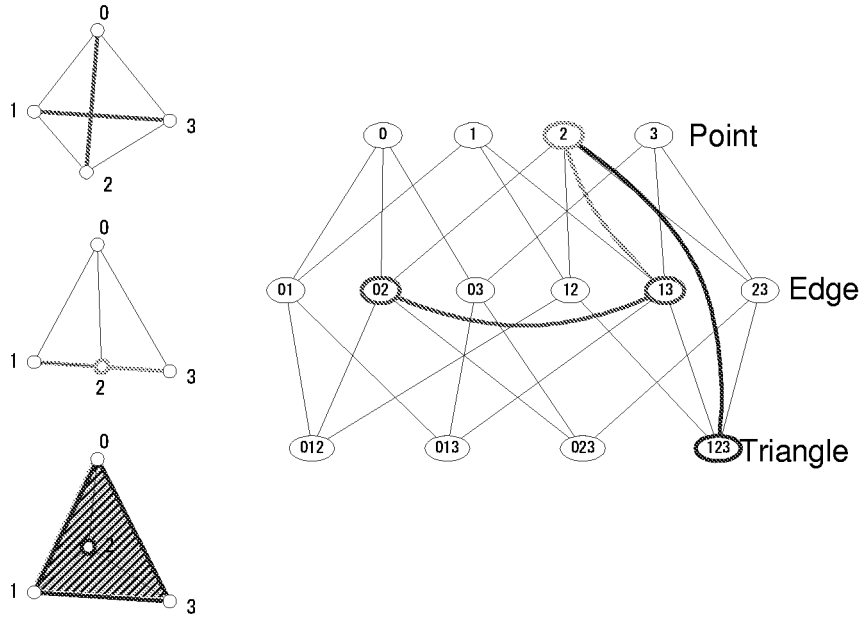

Figure 2.6: Intersections in a two dimensional triangulation

The following statement is important to see how far we can reduce the number of constraints:

Lemma 2.1 *All the vertices of simplices are assumed to be in a 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* $k + j = d$.

Proof: As it is obvious to be sufficient, we will show the "only if" part below.

From the assumption that all the vertices are in a general position, when two d-simplices t_i and t_j intersect, the intersection I_{ij} is a d-dimensional polytope, and has at least $d+1$ vertices. When at least one of the vertices of $t_j(/t_i)$ is inside $t_i(/t_j)$, this vertex(0-simplex) and $t_i(/t_j)$ intersect and the sum of the numbers of the dimensions is equal to d. This pair satisfies the condition.

Now we assume t_i does not contain vertices of t_j. If all the vertices of I_{ij} correspond to the vertices of t_i, t_j contains t_i inside and we can choose t_j itself and a vertex of t_i as above. Otherwise, out of at least $d+1$ vertices of I_{ij}, there exists a vertex p that is neither a vertex of t_i nor t_j. p is defined as an intersection of a face $f_{(i)}$ of t_i and a face $f_{(j)}$ of t_j. We can see that $\dim(f_{(i)}) + \dim(f_{(j)}) = d$. □

From Lemma 2.1, we obtain a set of d-simplices that do not intersect each other, by imposing the intersection constraints only on the pairs of simplices whose sum of dimensions is equal to d, together with the lattice constraints. The volume constraint is again necessary as in Formulation (SS), for the set to be a triangulation. At last, we obtain the following formulation that is an instance of the generalized stable set problem:

Formulation (GSS):

$$
\begin{aligned}
& \textbf{minimize } \boldsymbol{cx^d} \\
\text{s.t.} \quad & x_i^k \in \{0,1\}\ (0 \le k \le d) \\
& x_i^k + x_j^{d-k} \le 1 \text{ (for } i,j \text{ s.t. } k\text{-simplex } i \text{ and } (d-k)\text{-simplex } j \\
& \qquad \text{intersect, } 0 \le k \le d) \\
& x_i^k - x_j^{k+1} \ge 0 \text{ (for } i,j \text{ s.t. } k\text{-simplex } i \text{ is a facet of } (k+1)\text{-simplex } j, \\
& \qquad 0 \le k \le d-1) \\
& \sum_i v_i x_i^d = V_{\mathcal{A}}
\end{aligned}
$$

2.2 Set-Partitioning-Problem Formulations

In this section, we review related studies for formulating triangulation as an instance of the set partitioning problem.

The set partitioning formulation of triangulation has been discussed in the literature of computational algebra. We call *chambers* the smallest cells obtained by dividing

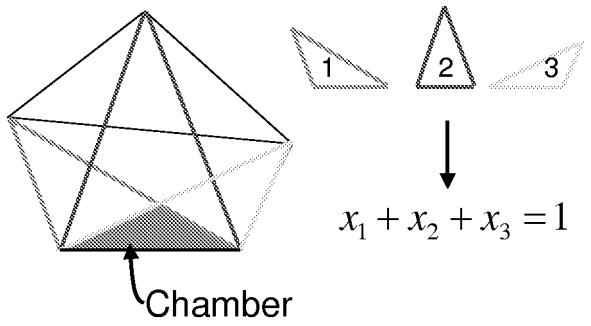

Figure 2.7: An example of a chamber constraint in 2D

$conv(\mathcal{A})$ with all the hyperplanes corresponding to $(d-1)$-simplices. Alekseyevskaya introduced a set partitioning formulation of triangulation by regarding chambers as elements, and d-simplices as subsets [1]. De Loera et al. refined the formulation by introducing *cocircuit form constraints* [19]. We will review the detail in the following sections.

2.2.1 The set partitioning problem

We use the term *chambers* for the minimal cells obtained by dividing the convex hull with the hyperplanes corresponding to all the possible $(d-1)$-simplices. Then, obtaining a 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].

Figure 2.7 is a two dimensional example. The shaded chamber can be included in three triangles, and we can put the corresponding constraints $x_1 + x_2 + x_3 = 1$. Here we do not have to enumerate the constraints for all the chambers, and Alekseyevskaya gave a method to choose a sufficient set of chambers. The formulation is as follows:

Formulation (SP):

$$\begin{aligned} & \textbf{minimize } \boldsymbol{cx^d} \\ \text{s.t.} \quad & x_i^d \in \{0,1\} \\ & \boldsymbol{Mx^d} = \mathbf{1} \quad M_{ij} = \begin{cases} 1 & t_j \text{ contains } i\text{-th chamber} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

2.2.2 Cocircuit form constraints

A large amount of geometric computation is necessary for handling chambers, and it can derive numerical instabilities depending on the point configuration. Therefore we consider another type of constraints instead, which are called *cocircuit form* constraints in [19] as they correspond to cocircuits when we consider the oriented matroid of the affine dependency of the point configuration $\mathcal{A}$.

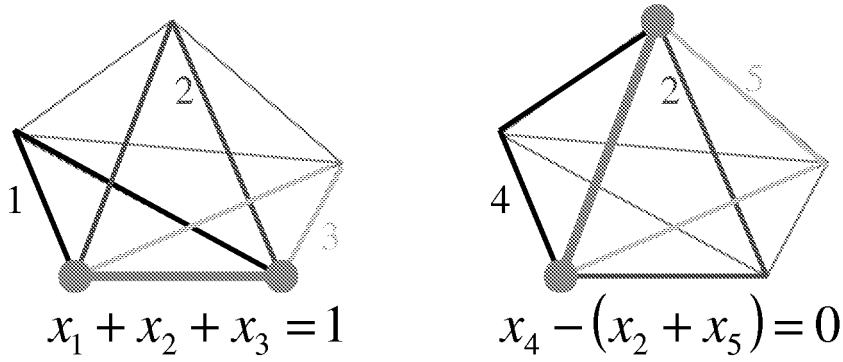

Figure 2.8: Examples of cocircuit form constraints in 2D

Let f_k be 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 $f_k \cup \{a\}$ be a d-simplex t_i defined by f_k and an element a of $\mathcal{A}$. The following constraint holds for all $(d-1)$-simplices [19].

$$\sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_k}} x_i^d - \sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_k}} x_i^d$$
$$= \begin{cases} 1 & f_k \text{ is on the boundary and oriented inside} \\ -1 & f_k \text{ is on the boundary and oriented outside} \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

For $(d-1)$-simplices on the boundary (Figure 2.8 left), the above equation is the same as the set partitioning constraint for the chamber that contains the corresponding $(d-1)$-simplex. For interior $(d-1)$-simplices (Figure 2.8 right), it corresponds to the difference of the constraints between the two chambers that are on both sides and share the $(d-1)$-simplex. If the $(d-1)$-simplex appear in the triangulation, both sums are equal to one, and zero otherwise. The difference is zero in both cases.

It is obvious that the incidence vectors of triangulations satisfy the constraints. Further, De Loera et al. proved important properties of the constraints by using matroidal operations and inductions [19]. That is, the cocircuit form constraints are sufficient to define the affine hull of the incidence vectors of d-simplices that constitute a triangulation. Let $P(\mathcal{A})$ denote the convex hull of the incidence vectors of d-simplices of all the triangulations of $\mathcal{A}$.

Theorem 2.1 *[19] The affine span of $P(\mathcal{A})$ is defined by the cocircuit form constraints for every interior $(d-1)$-simplex, together with one non-homogeneous linear equation valid on $P(\mathcal{A})$.*

If $conv(\mathcal{A})$ has a simplicial facet f_b, we can consider the cocircuit form constraint for f_b as the non-homogeneous constraint, otherwise we require a chamber constraint or the volume constraint.

Further, when $\mathcal{A}$ is in a general position, the rank of the constraints is also available. Let a^* denote an element of $\mathcal{A}$.

Theorem 2.2 *[19]*

1. *The cocircuit form constraints for $(d-1)$-simplices that do not have a^* as a vertex form a basis. The rank of the cocircuit form constraints is $\binom{n-1}{d}$.*
2. *$dim(P_{\mathcal{A}}) = \binom{n-1}{d+1}$.*

We also consider the cases where $\mathcal{A}$ is not in a general position in this thesis. Let $\mathcal{A}_k$ denote a subset of $\mathcal{A}$ that is located on $\mathcal{H}_{f_k}$ and not the vertices of f_k. We can put the non-homogeneous constraint for f_k only when f_k is on the boundary of $conv(\mathcal{A})$ and $\mathcal{A}_k$ is empty, that is, f_k is a facet of $conv(\mathcal{A})$.

Thus, if none of the facets of $conv(\mathcal{A})$ is a $(d-1)$-simplex, we need to put another non-homogeneous constraint. One simple way is to select a chamber h_{c^*} and put the set partitioning constraint for it.

Formulation (CO):

minimize $\boldsymbol{cx^d}$

s.t.

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

$$\sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_k}} x_i^d - \sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_k}} x_i^d = \begin{cases} 1 & f_k \text{ is on the boundary, oriented inside and } \mathcal{A}_k = \emptyset \\ -1 & f_k \text{ is on the boundary, oriented outside and } \mathcal{A}_k = \emptyset \\ 0 & f_k \text{ is not on the boundary} \end{cases}$$

$$\sum_i M_{c^* i} x_i^d = 1 \quad \text{(for a chamber } h_{c^*}\text{, only if } conv(\mathcal{A}) \text{ has no simplicial facet)}$$

2.3 Comparison of Formulations

In this section, we compare Formulations (SS), (GSS), and (CO). We do not consider Formulation (SP) because Formulation (CO) is proved to be equivalent to Formulation (SP) if the point configuration is in a general position [19].

Table 2.1 shows the size of each formulation, and Formulation (CO) is the most compact. On the other hand, we can observe that the number of constraints in Formulation (GSS) is quite reduced from that in Formulation (SS). This indicates the effectiveness of handling lower dimensional simplices explicitly. We will further see the significance of this approach in Chapter 4 in relation to cutting planes and the branch and bound procedure.

Polytopes defined by the linear programming relaxations

We investigate the relations among the polytopes defined by the linear programming relaxations of the formulations. As we generally solve the integer programming prob-

Table 2.1: Size of each formulation

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

#variables: total number of variables

#constraints: total number of constraints

#v/c: number of variables in a constraint

lems by first obtaining the relaxed linear programming solutions, the tightness of these polytopes is an very important factor for evaluation of formulations.

$P_{SS}(\mathcal{A})$, $\hat{P}_{GSS}(\mathcal{A})$, and $P_{CO}(\mathcal{A})$ denote the polytopes that correspond to the linear programming relaxation of Formulation (SS), (GSS), and (CO), respectively. The projection of $\hat{P}_{GSS}(\mathcal{A})$ to the subspace that corresponds to the variables for d-simplices is denoted by $P_{GSS}(\mathcal{A})$.

First we look into the relation between the two stable set formulations.

Theorem 2.3 *If $\mathcal{A}$ is in a general position,*

$$P_{SS}(\mathcal{A}) = P_{GSS}(\mathcal{A})$$

Proof: First we show $P_{SS}(\mathcal{A}) \subseteq P_{GSS}(\mathcal{A})$.

Consider a lifting procedure from the space corresponding to d-simplices to the space corresponding to k-simplices ($0 \leq k \leq d$) by assigning the values as follows:

$$x_i^k = \max_j x_j^{k+1} \quad \text{(s.t. } k\text{-simplex } i \text{ is a facet of } (k+1)\text{-simplex } j)$$

A point p_G , lifted with the above procedure from a point p in $P_{SS}(\mathcal{A})$, satisfies the lattice constraints for $\hat{P}_{GSS}(\mathcal{A})$. The volume constraint of $\hat{P}_{GSS}(\mathcal{A})$ is satisfied by definition. If we consider the geometry corresponding to p_G, from the lifting procedure and Lemma 2.1, for each pair of simplices (s_i, s_j) ($\dim(s_i)+\dim(s_j) = d$) that intersect each other, there exists a pair of d-simplices, $t_{(i)}$ and $t_{(j)}$, that has the same weight as $s_i(/s_j)$, has $s_i(/s_j)$ as a face, and intersects each other. As p_G is lifted from p in $P_{SS}(\mathcal{A})$, $x_{(i)}^d + x_{(j)}^d \leq 1$. Thus, $x_i^{dim(s_i)} + x_j^{dim(s_j)} = x_{(i)}^d + x_{(j)}^d \leq 1$. Now p_G satisfies all the constraints for $\hat{P}_{GSS}(\mathcal{A})$ and $p_G \in P_{GSS}(\mathcal{A})$. Therefore $P_{SS}(\mathcal{A}) \subseteq P_{GSS}(\mathcal{A})$.

Next we show $P_{GSS}(\mathcal{A}) \subseteq P_{SS}(\mathcal{A})$. Consider the geometry corresponding to a point q_G in $\hat{P}_{GSS}(\mathcal{A})$. From Lemma 2.1, for each pair of d-simplices t_a and t_b that intersect each other, there exists a pair of simplices $s_{(a)}$ and $s_{(b)}$, such that $s_{(a)}$ is a face of t_a, $s_{(b)}$ is a face of t_b, and $dim(s_{(a)}) + dim(s_{(b)}) = d$. From the definition of $\hat{P}_{GSS}(\mathcal{A})$,

$$x_a^d \leq x_{(a)}^{dim(s_{(a)})}, \quad x_b^d \leq x_{(b)}^{dim(s_{(b)})}, \quad x_{(a)}^{dim(s_{(a)})} + x_{(b)}^{dim(s_{(b)})} \leq 1.$$

Thus $x_a^d + x_b^d \leq 1$. As q_G also satisfies the volume constraint, the projection of q_G to the subspace corresponding to d-simplices is inside $P_{SS}(\mathcal{A})$. Therefore $P_{GSS}(\mathcal{A}) \subseteq P_{SS}(\mathcal{A})$. □

We have seen the equivalence between the two stable set formulations. On the other hand, the set partitioning constraints correspond to cliques in the intersection graph of d-simplices. Namely, Formulation (SP) and (CO) may give tighter constraints than stable set constraints. We will prove this by focusing on a situation that is slightly perturbed from a convex sum of two triangulations.

Lemma 2.2 *For any $d+3$ points in d (≤ 3) dimensions in a convex position, there exists a pair of points that are not connected by an edge on the boundary of the convex hull.*

Proof: It is obvious for two dimensional cases.

For three dimensional cases, we consider the convex hull C of $d+2$ ($=5$) points. As they are in a convex position, a bistellar flip is defined for C. One of the corresponding two triangulations of C uses an interior edge (see Figure 1.1), and we can take the both end of the edge as the pair. □

It is still open if we can choose two points that are not connected by an edge, in dimensions higher than three.

Lemma 2.3 *For the convex hull of $d+3$ points in d (≤ 3) dimensions and in a general position, 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.*

Proof: If the points are not in a convex position, we can consider flips and easily generate the situation. Thus, we assume that the $d+3$ points are in a convex position. Let C denote the convex hull of the points. We show below how to construct two triangulations that share no d-simplex. See Figure 2.9 for a two dimensional example.

We choose two points p and q that are not neighbors based on Lemma 2.2. Let C_p denote the convex hull of the $d+2$ points except p. There are exactly two triangulations of C_p, which correspond to a bistellar flip. There is a d-simplex t that does not contain q as a vertex. We call one of the two triangulations that contains t as $\mathcal{T}_p^q$, and the other as $\mathcal{T}_p^{\bar{q}}$.

Triangulations of C can be obtained from the triangulations of C_p and a d-simplex obtained by coning from p. $\mathcal{T}^q$ and $\mathcal{T}^{\bar{q}}$ are thus generated from $\mathcal{T}_p^q$ and $\mathcal{T}_p^{\bar{q}}$, respectively.

Here, $t \cup C \setminus C_p$ is the convex hull of $d+2$ points except q and can be flipped from $\mathcal{T}^q$. Then we obtain $\mathcal{T}^{q'}$. $\mathcal{T}^{q'}$ and $\mathcal{T}^{\bar{q}}$ share no d-simplex. and t does not belong to $\mathcal{T}^{q'}$ nor $\mathcal{T}^{\bar{q}}$. □

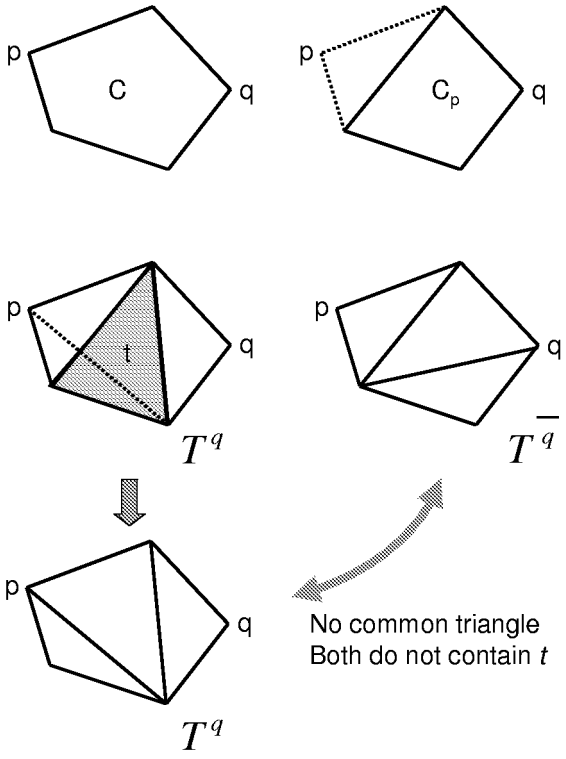

Figure 2.9: An illustration for Lemma 2.3

Theorem 2.4 *If $\mathcal{A}$ is in a general position,*

$$P_{CO}(\mathcal{A}) \subseteq P_{SS}(\mathcal{A})$$

Especially when the number of points $n \geq d+3$ and $d \leq 3$,

$$P_{CO}(\mathcal{A}) \subset P_{SS}(\mathcal{A})$$

Proof: First we show $P_{CO}(\mathcal{A}) \subseteq P_{SS}(\mathcal{A})$. We choose an arbitrary point p_{CO} in $P_{SS}(\mathcal{A})$. As $\mathcal{A}$ is in a general position and p_{CO} satisfies the chamber constraints, p_{CO} also satisfies the volume constraint. For each pair of d-simplices (t_i, t_j) that intersect with each other, there exists a chamber that belongs to both of them. From the chamber constraint for it, $x_i^d + x_j^d \leq 1$ holds. Now p_{CO} satisfies the volume constraints and all the intersection constraints for $P_{SS}(\mathcal{A})$, and $p_{CO} \in P_{SS}(\mathcal{A})$. Thus, $P_{CO}(\mathcal{A}) \subseteq P_{SS}(\mathcal{A})$.

Then we show $P_{CO}(\mathcal{A}) \subset P_{SS}(\mathcal{A})$ when the number of points $n \geq d+3$. We choose $d+3$ points out of n points in $\mathcal{A}$, and obtain two triangulations of the convex hull C of the $d+3$ points that share no d-simplex based on Lemma 2.3. We can extend from a triangulation of C to a triangulation of $conv(\mathcal{A})$ by using coning. We call $\mathcal{T}_A$ and $\mathcal{T}_B$, the two triangulations of $conv(\mathcal{A})$ that are thus generated and are the same except the interior of C.

Let t_k a d-simplex which is in C and an element of only $\mathcal{T}_A$, and t_m a d-simplex which is in C and is not an element of neither $\mathcal{T}_A$ nor $\mathcal{T}_B$. Let $\boldsymbol{e_k}(/\boldsymbol{e_m})$ denote the incidence vector of $t_k(/t_m)$ – which is also a unit vector –, respectively. Also let $\boldsymbol{x_A^d}(/\boldsymbol{x_B^d})$ denote the incidence vector of $\mathcal{T}_A(/\mathcal{T}_B)$, respectively.

We consider an incidence vector $\boldsymbol{x_{PTB}^d}$ which is slightly perturbed from the convex sum of $\boldsymbol{x_A^d}$ and $\boldsymbol{x_B^d}$ as:

$$\boldsymbol{x_{PTB}^d} = \frac{1}{2}\boldsymbol{x_A^d} + \frac{1}{2}\boldsymbol{x_B^d} - \varepsilon \boldsymbol{e_k} + \varepsilon(v_k/v_m)\boldsymbol{e_m} \quad (\varepsilon \ll \frac{1}{2})$$

$\boldsymbol{x_{PTB}^d}$ satisfies the volume constraint from the definition above. All the d-simplices in C has the values less than or equal to $\frac{1}{2}$, and $\boldsymbol{x_{PTB}^d}$ also satisfies the stable set constraints. Thus, $\boldsymbol{x_{PTB}^d} \in P_{SS}(\mathcal{A})$. Here, for some of the chambers in t_m the chamber constraints are not satisfied and $\boldsymbol{x_{PTB}^d} \notin P_{CO}(\mathcal{A})$. Therefore $P_{CO}(\mathcal{A}) \subset P_{SS}(\mathcal{A})$ if $n \geq d+3$. □

Extendabilities of the formulations

In the previous section, we showed that Formulation (CO) is the most compact both in size and the polytope obtained by the linear programming relaxation. We can say that Formulation (CO) is the best choice for the optimization of triangulations of **a point configuration.**

However, if we consider the triangulatability of non-convex polytopes which is important from the viewpoint of applications, or the minimal or maximal dissections

which are theoretically very interesting and we will further discuss in Section 5.3, Formulation (CO) can not cover all the situations.

On the other hand, Formulation (SS) and (GSS), which are based on the stable set problem, can give some insights into the triangulatability of non-convex polytopes by maximizing the volume to be triangulated. By loosing the definition of intersections and allowing the situation that d-simplices intersect but do not share interior points, we can easily cope with dissections.

Thus, the stable set formulations are flexible, and suitable for various extensions.

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. With Formulation (GSS), we have an alternative way that fixes the variables for 0-simplices (points) to be 1. For random point sets, the expected number of empty d-simplices is $O(n^d)$ [4], 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 $\binom{n}{d-1}$ [4].

2.4 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 this chapter. The other is bottleneck optimization, such as minimization of 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 as we will introduce more details in Section 3.4. Binary search is one way to avoid this numerical instability as we will present in Section 4.2.1.

Some of the following will be further investigated with computational experiments in Chapter 3.

Minimize the maximum aspect ratio. In applications such as FEM, we should avoid flatness of triangles or tetrahedra for the sake of computational stability and accuracy. The most straightforward index of the flatness is the aspect ratio (AR), which is defined as 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. We have two kinds of *bad* angles, large

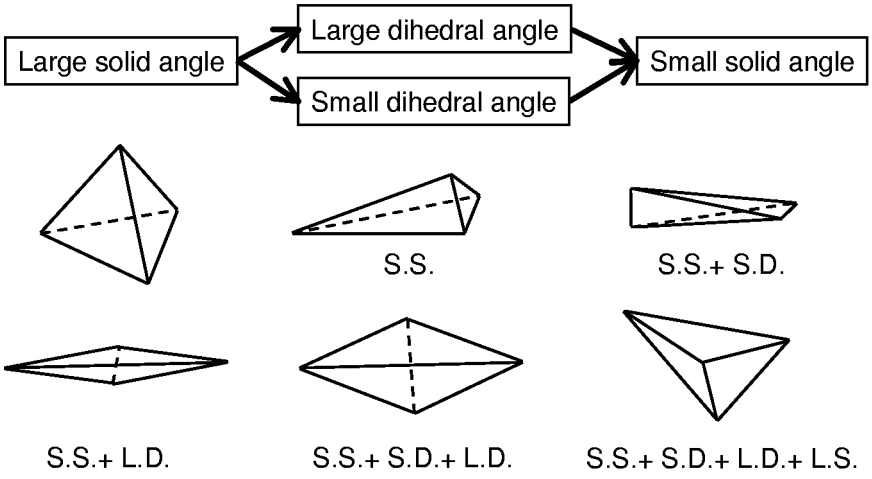

Figure 2.10: Varieties of bad angles of tetrahedra

($> 1/2 - \varepsilon$) ones and small ($< \varepsilon$) ones. They have relations as shown in Figure 2.10. For example, if a tetrahedron has a *small* dihedral angle, it surely has a small solid angle, and if it has a large solid angle, it also has all the kinds of other bad angles. Thus, tetrahedra are classified into 6 kinds as in Figure 2.10 [8].

There are several meshing algorithms that avoid some of the bad – too large or too small – angles (see [8], 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, using our IP-based approach.

Maximize "a tetrahedron shape measure" [25]. Dompierre et al. suggested *tetrahedron shape measures* [25] to be used to measure the quality of a tetrahedron and therefore a tetrahedral mesh, together with some benchmark test cases.

Definition 2.1 *[25]* A tetrahedron shape measure *is a continuous function that evaluates the quality of a tetrahedron. It must be invariant under translation, rotation, reflection and uniform scaling of the tetrahedron. It must be maximum for the regular tetrahedron and it must be minimum for a degenerate tetrahedron. There is no local maximum other than the global maximum for a regular tetrahedron and there is no local minimum other than the global minimum for a degenerate tetrahedron. For the ease of comparison, it* should *be scaled to the interval* $[0, 1]$*, and be* 1 *for the regular tetrahedron and* 0 *for a degenerate tetrahedron.*

Some researchers in the meshing community started to base their studies on it. For example, Freitag and Knupp gave an algorithm to improve the quality of mesh according to one of tetrahedron shape measures by moving the vertices [30].

It is also interesting how far we can improve such tetrahedron shape measures with the given point configuration.

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

Open Problem 2.1 *[9] Is there a polynomial-time algorithm for triangulating an arbitrary convex polyhedron with the minimum number of tetrahedra?*

A problem to obtain a spanning triangulation of a point configuration with the minimum number of tetrahedra would have the same or more computational complexity. Maximization of the cardinality is also an interesting problem.

Minimize the weight. The minimum weight triangulation(MWT) in two dimensions is a famous and interesting problem included in Garey and Johnson's list of problems that are neither known to be solvable in polynomial time or not [33]. We will further discuss on the MWT in Section 5.1. In three dimensions, the weight is extended from the edge length to the surface area of the triangles used in the triangulation.

Chapter 3

Experimental Analysis on the IP-based Optimization

This chapter gives the results of computational experiments, and some observations and investigations on them. All the experiments were carried out on an IBM RS/6000 model F50 (PowerPC 604e 332 MHz, 768 MB Memory) using IBM Optimization Solutions Library (*OSL*) Ver.2.0 [54] for solving mathematical programming problems. In particular, for linear programming, we applied the dual simplex algorithm without pre-/post-processing unless explicitly stated. For handling geometric objects, we used the library *CGAL* Ver.1.2 [12]. For obtaining Delaunay triangulation, we used the program *Triangle* [74] for two dimensions, and the program *DeWall* [22] for three dimensions.

Throughout the experiments, we limited triangulations to be spanning, in other words, to use all the points.

In Section 2.3, we investigated and compared the formulations and concluded that the formulation using cocircuit form constraints is the most compact way of describing the problem, and the most efficient way of solving it. Therefore, the following discussions in this thesis are based on Formulation (CO) in page 18.

From a geometrical point of view, optimizing the bottleneck such as minimization of the largest aspect ratio, is often required. In such cases, we introduce another

Table 3.1: 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

IP: CPU time for solving the IP problem

Others: CPU time for the rest

variable z as the upper bound[1] :

$$\textbf{minimize } z \tag{3.1}$$

s.t.

$$\begin{aligned}
x_i^d &\in \{0,1\} \\
c_i x_i^d &\leq z \quad \text{(for each } t_i\text{)} \\
&\sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}_{f_k}^+} x_i^d - \sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}_{f_k}^-} x_i^d \\
&= \begin{cases} 1 & f_k \text{ is on the boundary, oriented inside and } \mathcal{A}_k = \emptyset \\ -1 & f_k \text{ is on the boundary, oriented outside and } \mathcal{A}_k = \emptyset \\ 0 & f_k \text{ is not on the boundary} \end{cases} \\
\sum_i M_{c^* i} x_i^d &= 1 \quad \text{(for a chamber } h_{c^*}\text{, only if } conv(\mathcal{A}) \text{ has no simplicial facet)}
\end{aligned}$$

For example, Delaunay triangulation in two dimensions lexicographically maximizes the minimum angles. But this formulation focuses only on the largest(/smallest) value, and not in a lexicographic way. It is very difficult to cope with lexicographical optimization under integer programming framework, but it is still not a large drawback, because the lexicographic order is meaningless when the cardinality varies in higher dimensions.

3.1 Problem Size and Required Computation

We measured the size of the IP problem and the CPU time for obtaining the minimum weight (spanning) triangulation of uniformly distributed point sets in two and three dimensions (Table 3.1 and 3.2). The size of solvable instances is quite small in three dimensions. Further, if the point set were in a convex position, all the triangles/tetrahedra would be non-empty, the number of columns would become larger, and the size of solvable instances would become smaller.

[1] Max-Min problems can be solved by replacing c_i with $C - c_i$ where C is a constant larger than $\max_i c_i$

Table 3.2: 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

3.2 Optimalities of Triangulation Properties

Figures 3.1, 3.2, and 3.3 show the cardinality, the sum of the weight and the maximum aspect ratio, respectively, of triangulations in three dimensions. We used 10 instances of cardinality 10, 20, and 30 randomly generated in a unit cube with different seeds. The figures were drawn with `boxplot` of *MAPLE*, which shows 1) a box with a central line showing the median and a lower(/upper) line showing the first(/third) quartile respectively, 2) 2 lines extending from the box of maximal length 3/2 the interquartile range but not extending the range of the data, 3) points that lie outside the extent of the previous elements. D, W, C, and AR in these figures represent Delaunay triangulation, the minimum weight triangulation, the minimum cardinality triangulation (MWT), and the triangulation with the minimum maximum aspect ratio, respectively.

Figure 3.1 shows that Delaunay triangulation is not at all close to the MWT in three dimensions. It is interesting because there are studies to approximate the MWT with Delaunay triangulation in two dimensions [46].

Figure 3.2 shows that Delaunay triangulation avoids flat tetrahedra quite well. There is still a gap between Delaunay triangulation and the optimal triangulation, which is obtained by the triangulation with the minimum maximum aspect ratio. We, however, can say that Delaunay triangulation is good enough if we consider that it can be obtained efficiently (in $O(n \log n)$ time) and can handle very large instances.

From Figure 3.3 we can observe that Delaunay triangulation tends to contain more tetrahedra than the other triangulations. However, the maximum cardinality triangulation is quite different from the above four types of triangulations. We can see that it contains far more tetrahedra than others from Figure 3.4, which shows the distribution of the minimum, maximum, and Delaunay triangulation's cardinality with the input of 10, 20, and 30 points uniformly distributed in a unit cube.

3.3 Some Special Point Configurations

In this section, we introduce some point configurations and their optimal triangulations that we found interesting.

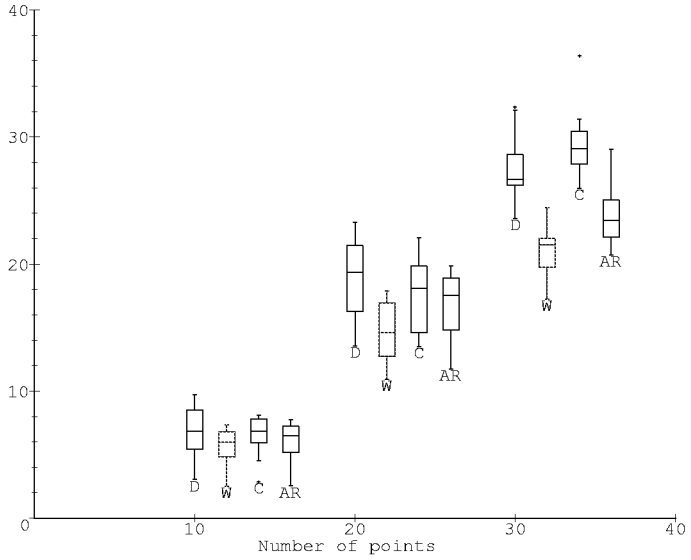

Figure 3.1: Weight of triangulations

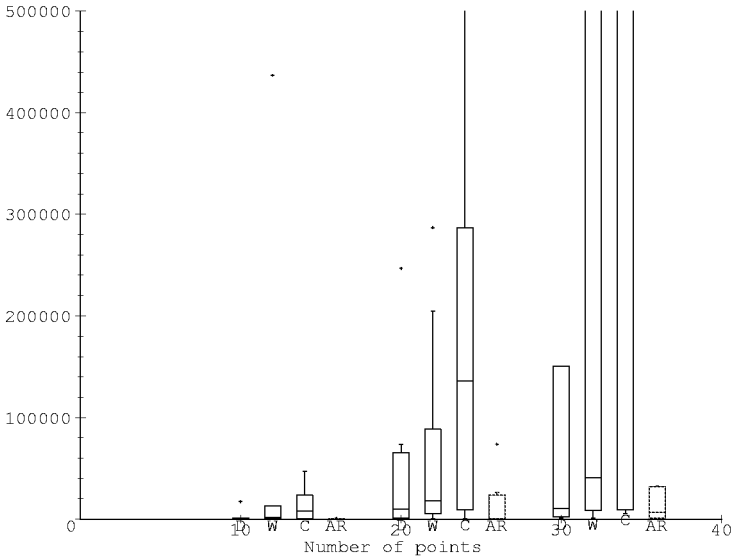

Figure 3.2: Minimum maximum aspect ratio of triangulations

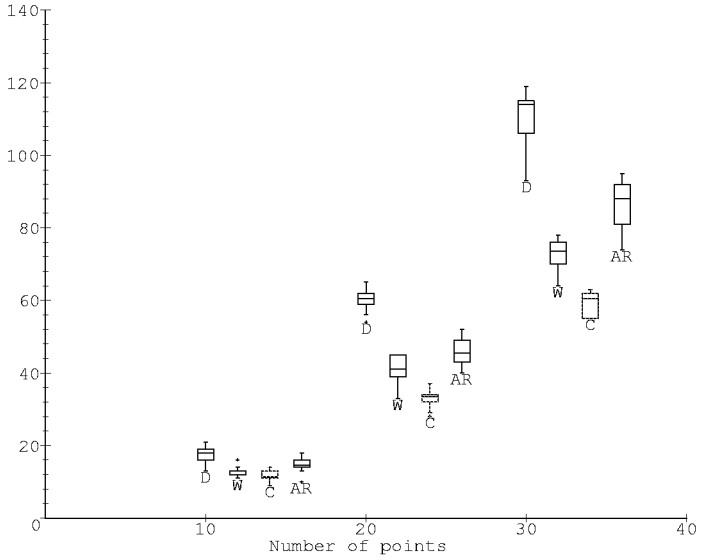

Figure 3.3: Cardinality of triangulations

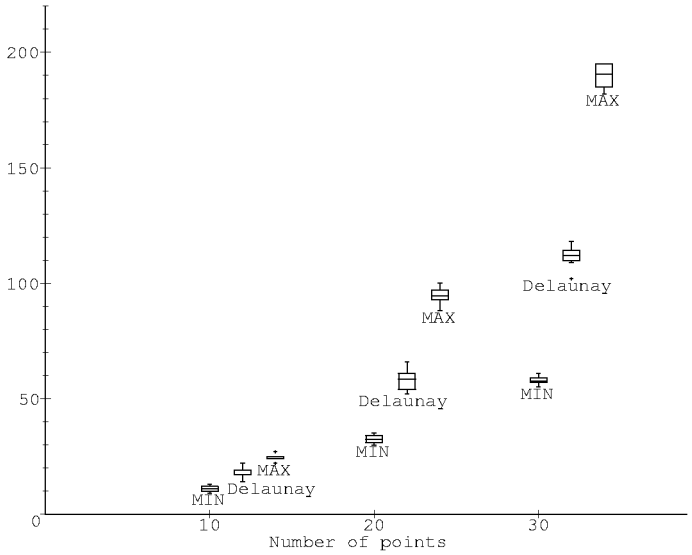

Figure 3.4: Cardinality of Delaunay triangulation

3.3.1 10 points in a cube

We first illustrate a simple example of 10 points in a cube, one instance of those we used in the previous section. In Figure 3.5, the one in the top is Delaunay triangulation. Its largest aspect ratio is 3112.9, and the tetrahedron is almost a line when we view it from the side as in the figure. When we minimize the maximum aspect ratio, the flattest tetrahedron in Delaunay triangulation is splitted into two tetrahedra by introducing another point as in the bottom-left one in Figure 3.5, and the flattest tetrahedron has aspect ratio of 79.8. The bottom-right one in Figure 3.5 is the side view of the flattest tetrahedron. We can see that it is much thicker than in the Delaunay case.

3.3.2 Lattice points

We consider lattice points obtained by aligning unit cubes. Let $\mathcal{A}(p,q,r)$ denote a point configuration corresponding to the vertices of $p \times q \times r$ unit cubes that are aligned and form a hexahedral in total. For example, $\mathcal{A}(1,1,1)$ denote 8 points that correspond to the vertices of a unit cube.

We calculated the minimum cardinality triangulations of some instances of $\mathcal{A}(p,q,r)$. The maximum cardinality for $\mathcal{A}(p,q,r)$ is always equals to $6pqr$, because the smallest possible volume of a tetrahedron is equal to $\frac{1}{6}$, which means the upper bound is $6pqr$. We can easily obtain a triangulation of cardinality $6pqr$ just by triangulating each unit cube into 6 tetrahedra, of which Delaunay triangulation is an instance.

Table 3.3.2 shows the minimum cardinalities for several $\mathcal{A}(p,q,r)$s. As is well known, the minimum cardinality triangulation of a cube has 5 tetrahedra, one of which has the volume of $\frac{1}{3}$, and the others have the volume of $\frac{1}{6}$. For special cases of $\mathcal{A}(p,1,1)$, we can observe that the minimum cardinality triangulation has one large tetrahedra corresponding to the one with volume $\frac{1}{3}$ in the unit cube case, and with other four tetrahedra with $\frac{1}{6}$ divided into p tetrahedra (Figure 3.6). Thus we conjecture as follows:

Conjecture 3.1 *The minimum cardinality triangulation of* $\mathcal{A}(p,1,1)$ *has* $4p+1$ *tetrahedra.*

We can also observe from Table 3.3.2 that As q and r become larger, the gap between the minimum cardinality and $(4p+1) \times q \times r$ becomes larger.

3.3.3 Cyclic polytopes

A cyclic polytope can be constructed by taking the convex hull of the points on the moment curve, namely, with the coordinate $(x_i,\ x_i^2, \ldots,\ x_i^d)$, and has the same face lattice independent of the values of x_i [38].

In three dimensions, the lower bound of the number of tetrahedra in a triangulation is $n-3$ and the upper bound is $\binom{n-1}{2} - n' + 2$ (n': points on the convex hull). Cyclic polytopes are known to have both the triangulations that realize the upper/lower

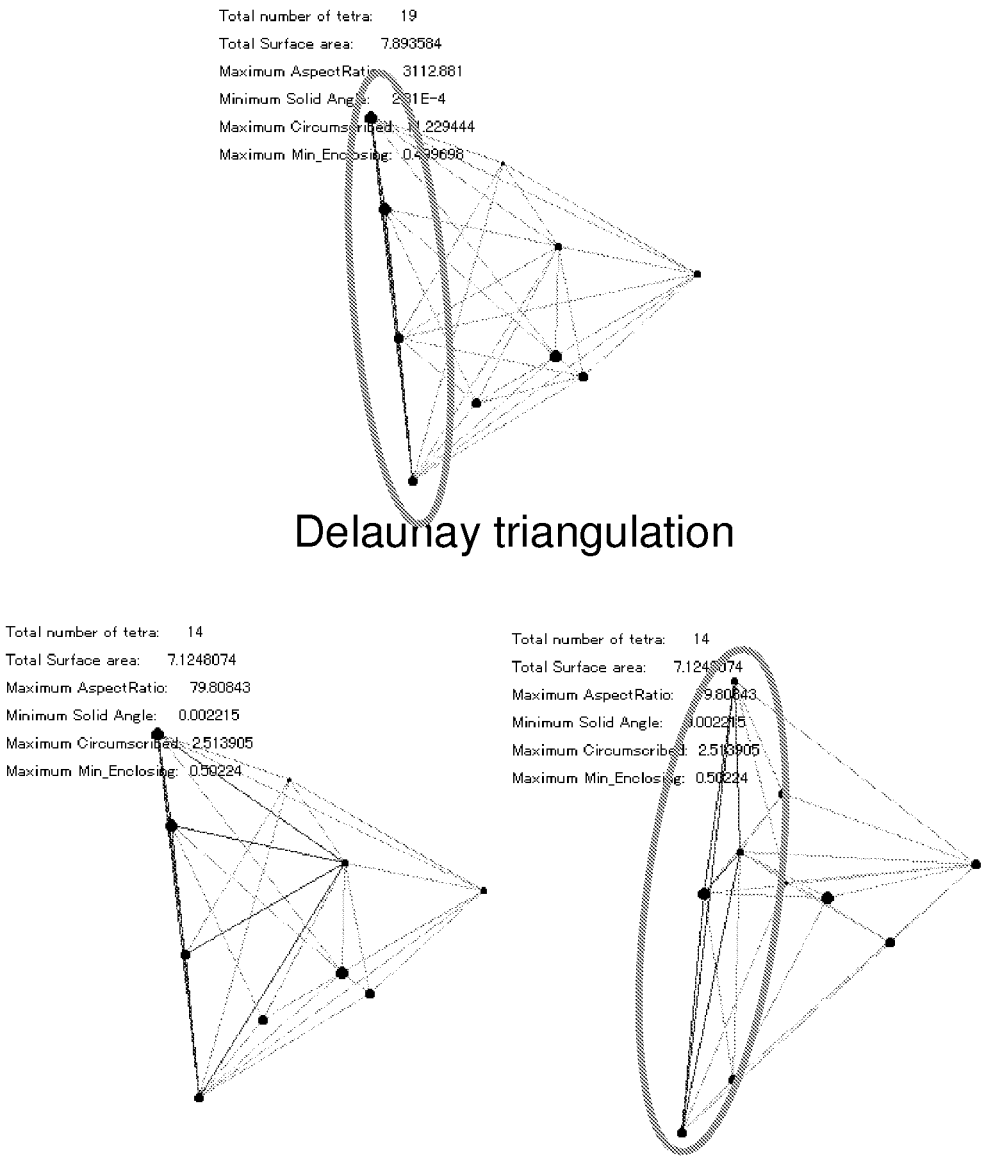

Figure 3.5: The flattest tetrahedron in triangulations

Table 3.3: The minimum and maximum cardinalities of triangulations of $\mathcal{A}(p,q,r)$

Point configuration	Minimum	Maximum
$\mathcal{A}(1,1,1)$	5	6
$\mathcal{A}(2,1,1)$	9	12
$\mathcal{A}(3,1,1)$	13	18
$\mathcal{A}(4,1,1)$	17	24
$\mathcal{A}(5,1,1)$	21	30
$\mathcal{A}(2,2,1)$	18	24
$\mathcal{A}(2,2,2)$	35	48
$\mathcal{A}(2,2,3)$	49	72

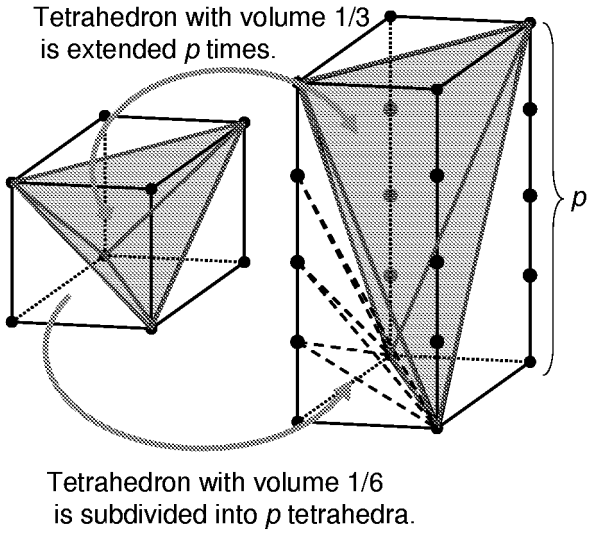

Figure 3.6: The minimum cardinality triangulation of $\mathcal{A}(p,1,1)$

bound, respectively [27]. We confirmed this fact by minimizing and maximizing the cardinality of triangulations of n points on a moment curve $\{p_i = (x_i,\ x_i^2,\ x_i^3) \mid x_i = i+1\ (i = 0, \ldots, n-1)\}$, and looking into the output. We observed that the minimum cardinality triangulation consisted of $n-3$ tetrahedra $\{(p_0, p_i, p_{i+1}, p_{n-1}) \mid (1 \leq i \leq n-3)\}$, and the maximum cardinality triangulation consisted of $\binom{n-2}{2}$ tetrahedra $\{(p_i, p_{i+1}, p_j, p_{j+1}) \mid (0 \leq i < j \leq n-3)\}$.

With this point set, Delaunay triangulation is also the maximum cardinality triangulation. But it is just a special case and we can easily generate a counter example where Delaunay triangulation is not the maximum cardinality triangulation, by moving the x-coordinates of the points. The maximal cardinality triangulation is known to be the lower hull of the four dimensional cyclic polytope obtained by lifting the above point set with the fourth coordinate of x_i^4 [27], where Delaunay triangulation corresponds to the lifting with the fourth coordinate value of $(x_i)^2 + (x_i^2)^2 + (x_i^3)^2 = x_i^2 + x_i^4 + x_i^6$

3.4 Observations

Through computational experiments based on Formulation (CO) in page 18, we obtained the following observations:

1. Optimizing the bottleneck by using the formulation of (3.1), namely, by introducing another variable for the upper bound, does not work. We obtain too many highly fractional variables (Figure 3.7), the branch and bound procedure just goes down deeper, and the lower bound never improves.

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 except for the minimum cardinality triangulation, in other words, the non-weighted case. Thus, practically we do not have fractional solutions with two dimensional instances.

3. The problem is highly primal degenerate. While the rank is $\binom{n-1}{d}$ (See Theorem 2.2), the number of d-simplices in a triangulation, namely, the number of positive coordinates in a integer feasible solution is $O(n^{\frac{d+1}{2}})$ which is much smaller than the rank. Table 3.4 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 unweighted 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, but still it is not fast with our instances. We should also note that the interior point method requires more memory than the simplex methods, and the size of soluble instances are smaller.

Table 3.4: Primal/dual degeneracy and the effectiveness of LP algorithms

No. 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: No. of basic variables with 0 value
Dual degeneracy: No. 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. 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. This means that an algorithm that specify the cardinality then minimize the weight to obtain the minimum(/maximum) cardinality, does not work in practice.

5. 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 (CO).

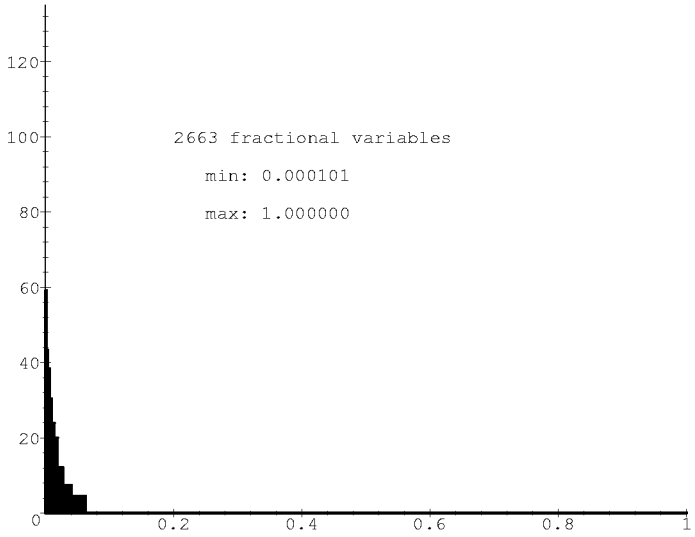

Figure 3.7: Distribution of positive values: 30 points in 3D, bottleneck

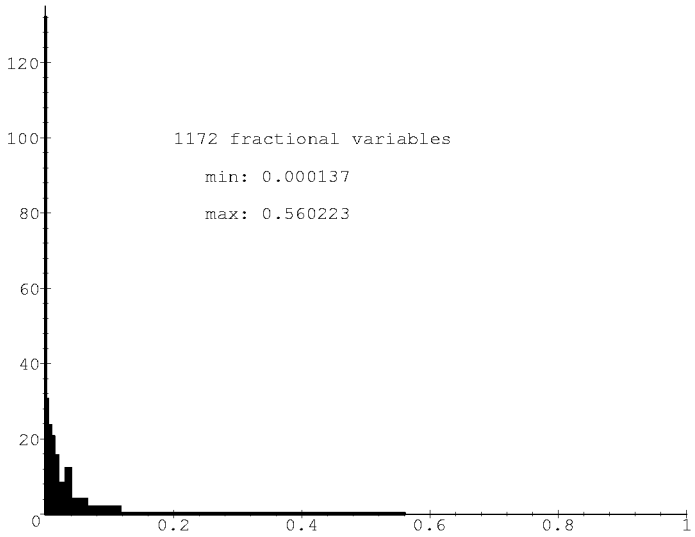

Figure 3.8: Distribution of positive values: 30 points in 3D, non-weighted

Chapter 4

Coping with Difficult Cases

In Chapter 3, we investigated the IP-based optimization of triangulations of a point configuration mainly in the three-dimensional space through computational experiments. Although Formulation (CO) in page 18, which uses cocircuit form constraints introduced by De Loera et al., is a good choice, we observed that there still remained difficulties to solve practical instances. One is that the size of instances we can solve is small. Another is that we obtain highly fractional solutions with certain objective functions.

In this chapter, we face the difficulties by investigating them as IP problems, and by utilizing methods in IP. In Section 4.1, we introduce column generation methods to solve larger instances. In Section 4.2, we cope with fractional solutions. First we apply a binary search algorithm focusing on the cases of the bottleneck optimization. We then focus on the cases with the unweighted objective function. Based on the observations of fractional examples, we introduce two kinds of new cutting planes which utilize the geometrical aspect of triangulations. We also look into the branch and bound procedure, for we still have fractional solutions after adding the cutting planes.

4.1 Column Generation Methods for Solving Large Instances

If we consider applications such as FEM, the size of currently soluble instances that we reported in Section 3.1 is not enough. In this section, we investigate column generation methods 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 therefore can improve the current solution, until we can find no more candidates.[1]

[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. This procedure is called *branch and price*.

4.1.1 A geometrical interpretation of column generation

First we consider the dual of the linear relaxation of Formulation (CO) in page 18. Each d-simplex has $d+1$ vertices, and within it we can consider $d+1$ pairs of a vertex and a facet. Let coefficient $s(i,j)$ be 1 if the j-th ($1 \leq j \leq d+1$) vertex of d-simplex t_i is on the positive side of the counterpart facet, and -1 otherwise. Also let $n(i,j)$ denote the suffix of the j-th ($1 \leq j \leq d+1$) facet of t_i. We assign a variable y_k as the dual variable that corresponds to the cocircuit form constraint for $(d-1)$ simplex f_k in the primal problem, namely, Formulation (CO). b_k is the right hand side value of the constraint.

Formulation (DualCO):

maximize ***by***

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

Here, the reduced cost corresponding to t_i is defined as

$$\tilde{c_i} = c_i - \sum_{j=1}^{d+1} s(i,j) y_{n(i,j)}$$

This $\tilde{c_i}$ is the difference between the right hand side and the left hand side of the constraint in Formulation (DualCO). Only when some of these values are negative, the current solution becomes dual infeasible and the primal objective function can have a way to improve.

Thus, starting with a subset of columns, and iteratively solving the current problem and introducing a set of new columns with negative reduced cost, we can gradually obtain better feasible solutions. Finally when there is no column with negative reduced cost, we can say that the current solution is optimal.

Figure 4.1 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 the objective function is improved ($c_1 + c_2 > c_3 + c_4$).

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, and the objective function is improved at that time.

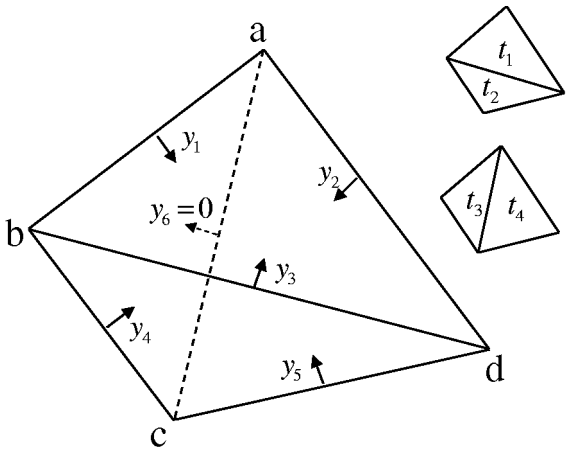

Figure 4.1: A small sample of column generation

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 that has never been used by the d-simplices already took into account, the solution will never be improved until at least one d-simplex is introduced on both sides, as in Case 1 and 3. In Formulation (DualCO), taking d-simplex t_k with $\tilde{c_k} < 0$ into consideration corresponds to adding a cut

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

which is violated by the current y. When d-simplices are introduced only on one side of $(d-1)$-simplex f_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 applying bistellar flips is an efficient way in the early stage of column generation, because it always introduces d-simplices on both sides of the newly introduced $(d-1)$-simplices.

4.1.2 SPRINT

SPRINT is a method to solve large instances of the linear programming problem, especially when it has many columns, which was applied to practical airline crew scheduling problems [2]. The idea is similar to column generation, but it takes a fixed number of candidates with small or negative reduced cost into account in each iteration. Namely, sometimes candidates with positive reduced cost can be introduced.

4.1.3 Computational experiments: Primal cases

We did some computational experiments using small instances of 30 points in three dimensions. We used simplex methods in each iteration of the column generation. As we mentioned in Section 3.4, the problem is highly primal degenerate, and it caused two difficulties.

One is that the primal simplex method is very slow with our problem as we have shown in Table 3.4. In each iteration, we start from a primal feasible solution that we have obtained in the previous iteration, and to use the primal simplex method seems a reasonable way. We however observed that actually it was often slower than solving with the dual simplex method from scratch.

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

As we mentioned in Section 4.1.2, SPRINT approach can give a good performance practically. According to our experiments, it terminated in less than 20 iterations with SPRINT, while it took more than 100 iterations with the naive column generation. Still the slowness of the primal simplex method in each iteration remains.

Interior point methods are known to be robust against the degeneracy (e.g. [49]), and we observed it in Table 3.4. We actually observed that the column generation procedure terminated soon after the optimal solution was obtained if the interior point method was used for solving the LP problems. Unfortunately, interior point methods require more memory than simplex methods, and we cannot solve large instances yet.

4.1.4 Column generation in dual

We have observed that the column generation did not work well because of the primal degeneracy of the problem. As the column generation corresponds to the primal simplex method in essence, problems that cannot be solved with the primal simplex method efficiently also cannot be solved efficiently with the column generation.

On the other hand, our problem can be solved efficiently with the dual simplex method. Thus, we investigate the column generation in dual, in other words, we are going to add constraints gradually. As there are much more variables than constraints, we cannot reduce the total problem size if we take all the variables into account in advance. We need to increase both variables and constraints gradually during the procedure. The criteria for variables and constraints to be newly added are:

Constraints (in primal) To be violated by the current solution (*). As a constraint corresponds to a facet, it means the situation where the sum of d-simplices on one side of the facet is different from that of the other side.

Variables (in primal) To have negative reduced cost that is calculated using the current dual variables (**).

We now introduce a procedure of column and row generation.

1. Initialization: A triangulation which can be obtained efficiently such as Delaunay triangulation.
   Variables: All the d-simplices in the triangulation.
   constraints: All the $(d-1)$-simplices in the triangulation.
   Go to Step 3.

2. *Column generation*: Add variables according to Condition (**)
   If no variable is added, go to Step 4.

3. Solve the linear programming relaxation. Go to Step 2.

4. *Row generation*: Add constraints according to Condition (*)
   If no constraint is added, terminate.

5. Solve the linear programming relaxation. Go to Step 2.

Proposition 4.1 *The procedure above gives the optimal triangulation.*

Proof: Repeating Step 2 and 3 until no variable is added, is a simple column generation procedure and gives the optimal solution with the currently available constraints. Thus, every time we add constraints in Step 4, we obtain the exact solution to the current problem by calling Step 2 and 3. This is equivalent to the cut generation with all the columns available, and gives the optimal solution to the original problem. □

As it is not efficient to loop until there is completely no candidate to be added, we repeat column generation and row generation in turn in practice.

4.1.5 Computational experiments: Dual cases

We looked at the performance of the row and column generation with sets of points uniformly distributed in a unit cube as input, and with the sum of the weight as the objective function to be minimized (Table 4.1). All the linear programming problems were solved with the dual simplex method.

It terminates at about 40 iterations, and it is faster compared with the primal column generation method that requires more than 100 iterations. But still it is much slower than naively solving the whole problem at once. The final size of the matrix is smaller with the row and column generation and it can be useful when we try to solve large instances exactly.

Table 4.1: Performance of a row and column generation method

Points	Naive method			Column and cut generation			
	Rows	Cols	Pivotings	Rows	Cols	Iterations	Pivotings(total)
30	3654	19638	3826	2508	5706	36	61282
40	9139	57735	21702	7209	18822	38	394687
50	18424	139982	105242	13631	37938	42	1602016

4.1.6 Remarks

We have observed that the column generation methods both in primal and dual required smaller number of columns and rows than solving the whole problem naively, but took much more time and pivotings. Further, the method in dual was better in the convergence. It indicates that we can solve larger instances than the current largest if we spend enough time.

On the other hand, the column generation in primal can be regarded as an approximation procedure. As it always maintains the current best feasible solution, we can stop the iteration intermediately and take it as an approximated solution. We do not have to solve the whole problem, and can handle larger instances than in Table 4.1 with the current computational power. With the dual method, we go through the primal infeasible region until the optimal solution is obtained, and we cannot obtain a *triangulation* (feasible solution) until the end of the procedure.

Thus, as far as we use the dual simplex method to solve LP problems, we should apply the column generation in dual when we require an exact solution, and apply the column generation in primal when we require an approximate solution. If more computational power becomes available and interior point methods and their implementations further improve, the column generation in primal using interior point methods would become the best choice for following two reasons: (1) interior point methods are known to be able to successfully utilize a feasible solution even with primally degenerate problems, in other words, the warm start works fine, and (2) the iterations of column generation converge fast with interior point methods as we observed in Section 4.1.3.

4.2 IP Approaches for Fractional Solutions

In this section, we cope with one of the difficulties we observed in Section 3.4, that is, we obtain a highly fractional solution from the linear programming relaxation when the objective function is a bottleneck value, or unweighted.

For bottleneck type problems, we introduce a binary search algorithm. For the unweighted objective function, we further look into the problem as an IP problem. Based on the observation of fractional instances, we introduce two kinds of novel cutting planes which are based on the geometric aspects of triangulation, and specific

to the optimization of triangulation. As fractional values can remain even after adding the cutting planes, we look into the improvement of the branch and bound procedure by introducing variables correspond to lower dimensional simplices.

4.2.1 A binary search algorithm for bottleneck optimization

As we observed in Section 3.4, when we optimize the bottleneck value of triangulations, introduction of the additional variable as in Formulation (3.1) does not work at all. The branch and bound procedure just goes deeper and we cannot solve even small instances such as 20 points in a cube.

We here introduce a binary search method for bottleneck optimization. We assume the problem to be the minimization of the maximum, such as obtaining the triangulation with the minimum maximum aspect ratio. Suppose there are N d-simplices, $t_0, \ldots, t_N$, and they are sorted in ascending order of their cost. Let U denote the suffix of the d-simplex with the largest cost in a currently available triangulation, and L denote the suffix such that there is no triangulation only using a subset of d-simplices $\{t_i | 0 \leq i \leq L\}$. 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 above procedure does not address the degenerate cases where multiple d-simplices have the same cost, but it can easily be coped with by grouping the d-simplices as one. 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 3.4, we only have to solve the relaxed LP problem to obtain an integral solution in practice. Further, Step 3 turns out to be infeasible in most of the iterations, and we can stop the calculation in the early phase of the simplex method.

Table 4.2 shows the results of the computational experiments. We used points randomly distributed in a cube as input. To obtain the triangulation with the minimum maximum aspect ratio, we did 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.

Table 4.2: Binary search for the bottleneck value

No. of points	No. of iterations	No. of feasible itr.	No. of 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

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.

4.2.2 Cutting planes

First we investigate small examples of the fractional solutions, then based on the observation, we introduce two kinds of cutting planes.

Small fractional examples

De Loera et al. introduced an fractional example in two dimensions [19]. 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 $p_{\{123\}} = p_{\{234\}} = p_{\{345\}} = p_{\{145\}} = p_{\{125\}} = p_{\{013\}} = p_{\{024\}} = p_{\{035\}} = p_{\{014\}} = p_{\{025\}} = \frac{1}{2}$ satisfies all the cocircuit form constraints, namely, is a feasible solution of the linear programming relaxation. It is important to refer De Loera et al.'s result

Theorem 4.1 *[19] Let $\mathcal{A} \in \mathcal{R}^d$ be a configuration of n points. $P_{CO}(\mathcal{A})$ is equal to $P(\mathcal{A})$ (the convex hull of the incidence vectors of triangulations of $\mathcal{A}$) in the following cases:*
(i) $d = 2$ and all points lie on the boundary of a convex polygon.
(ii) $d = 1$.
(iii) $n \leq d + 3$.

This theorem means that the fractional example above is the smallest one in two dimensions.

The subgraph of the intersection graph of triangles, induced by the nodes corresponding to the triangles of value $\frac{1}{2}$ that do (/do not) contain vertex 0, constitutes an odd-cycle of size 5, respectively. Both of the corresponding constraints

$$p_{\{123\}} + p_{\{234\}} + p_{\{345\}} + p_{\{145\}} + p_{\{125\}} \leq 2 \tag{4.1}$$

$$p_{\{013\}} + p_{\{024\}} + p_{\{035\}} + p_{\{014\}} + p_{\{025\}} \leq 2 \tag{4.2}$$

cut off p. Namely, traditional odd-cycle cuts can be effective for Formulation (CO) in page 18.

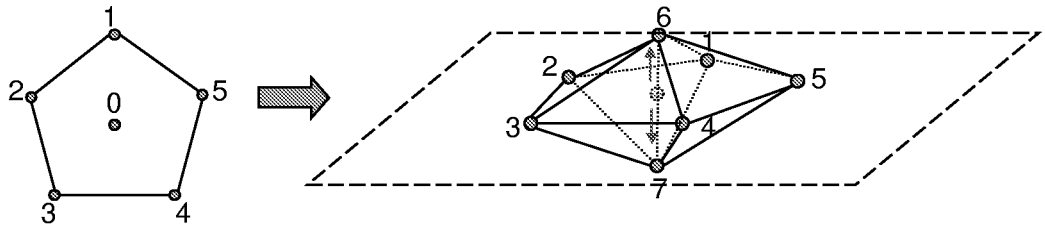

Figure 4.2: Generating a fractional example in three dimensions

From a geometrical point of view, the number of triangles around a point inside the convex hull must be equal to or larger than 3;

$$p_{\{013\}} + p_{\{024\}} + p_{\{035\}} + p_{\{014\}} + p_{\{025\}} + p_{\{012\}} + p_{\{023\}} + p_{\{034\}} + p_{\{045\}} + p_{\{015\}} \geq 3 \tag{4.3}$$

This also cuts off p.

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 (Figure 4.2). Let $\mathcal{A}_7$ denote this point configuration. This corresponds to the lift used to obtain the Lawrence polytope [78]. The polyhedral cone obtained by just lifting vertex 0 to the third dimension corresponds to a point configuration of 5 points in a convex position in two dimensions, and Formulation (CO) gives a integral polytope for it. Thus we need to use 7 points as above. Again, Theorem 4.1 shows that this is the smallest example in three dimensions.

$q \in \mathcal{R}^{30}$ with $q_{\{1367\}} = q_{\{2467\}} = q_{\{3567\}} = q_{\{1467\}} = q_{\{2567\}} = q_{\{1236\}} = q_{\{2346\}} = q_{\{3456\}} = q_{\{1456\}} = q_{\{1256\}} = q_{\{1237\}} = q_{\{2347\}} = q_{\{3457\}} = q_{\{1457\}} = q_{\{1257\}} = \frac{1}{2}$ 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.

Convex polygon cuts

As cutting planes for Formulation (2.1) to obtain the MWT in two dimensions, Kyoda et al. 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 [44]. Formulation (SS) in page 10 and Formulation (GSS) in page 15, which are the generalization of Formulation (2.1), are based on the stable set problem and these cutting planes can cut off fractional solutions of their linear programming relaxations. We have observed in the previous section that odd-cycle cuts could be effective for Formulation (CO) in page 18. Here, we investigate the effectiveness of the

convex polygon cuts with respect to Formulation (CO).

The convex polygon cut is based on the property that the number of edges is invariant for triangulations in two dimensions. Thus, this cut is valid only in two dimensions. Although it was originally 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 4.1 *[44] Let $\mathcal{V}$ denote a configuration of points in two dimensions, and $M_{\mathcal{V}}$ denote the cardinality of the spanning triangulations of $\mathcal{V}$. Further let x denote the incidence vector of a spanning triangulation of a point configuration $\mathcal{A}$ ($\mathcal{A} \supseteq \mathcal{V}$, $(\mathcal{A} \backslash \mathcal{V}) \cap conv(\mathcal{V}) = \emptyset$), and let $D_{\mathcal{V}}$ denote the dimensions of $\boldsymbol{x}$ whose corresponding triangles only have the elements of $\mathcal{V}$ as their vertices. The following inequality holds:*

$$\textbf{Convex polygon cut for } \mathcal{V} \quad \sum_{i \in D_{\mathcal{V}}} x_i \leq M_{\mathcal{V}}$$

Proposition 4.2 *A convex polygon cut forms a face of $P(\mathcal{A})$.*

Proof: It is obvious that the inequality is satisfied by a triangulation. Further, the equality holds when the spanning triangulation of $\mathcal{V}$ is a subset of the triangulation of $\mathcal{A}$. □

As we have seen in the previous section, the odd-cycle cuts could cut off the fractional solution obtained using Formulation (CO). On the other hand, we cannot find a convex polygon cut that is violated by p, namely the convex polygon cuts are redundant. We give a theoretical background to this observation.

Proposition 4.3 *A convex polygon cut forms a face of, and therefore is redundant for $P_{CO}(\mathcal{A})$.*

Proof: If $\mathcal{A} = \mathcal{V}$, obviously

$$\sum_{i \in D_{\mathcal{V}}} x_i \leq M_{\mathcal{V}}.$$

Thus we assume below $\mathcal{A} \supset \mathcal{V}$.

We consider a point $\boldsymbol{p}$ in $P_{CO}(\mathcal{A})$. For a chamber c inside $conv(\mathcal{V})$, let S_c denote the set of triangles that contains c. As $\boldsymbol{p}$ satisfies the chamber constraints,

$$\sum_{i \in D_{\mathcal{V}} \cap S_c} p_i \leq 1$$

If the equality holds for all the chambers inside $conv(\mathcal{V})$, we can ignore the region outside $conv(\mathcal{V})$ and the situation is the same as the case of $\mathcal{A} = \mathcal{V}$.

If there exists a chamber for which the inequality holds strictly, we can surely find a triangle that contains the chamber, whose vertices are elements of $\mathcal{V}$, and within which the inequality holds strictly. By repeatedly adding the triangle so that the inequality becomes equality until we have no chamber with strict inequality, we obtain the situation above where the equality holds for every chamber, and still $\sum_{i \in D_{\mathcal{V}}} x_i \leq M_{\mathcal{V}}$ holds.

Thus, a convex polygon cut is always satisfied by a point in $P_{CO}(\mathcal{A})$. From Proposition 4.2 and $P(\mathcal{A}) \subseteq P_{CO}(\mathcal{A})$, a convex polygon cut is also a face of $P_{CO}(\mathcal{A})$. □

$(d-2)$-face cuts

We have shown that the convex polygon cuts, which are based on geometric information on triangulation, were not useful. Although we observed that the traditional cutting planes such as odd-cycle cuts were effective, we can seldom find ones that cut off the fractional solution when the instance is large and the fractional values are very small. In other words, the traditional cutting planes for the stable set and set partitioning problem are not useful in practice.

Figure 3.8 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. Further, the cutting planes that were effective to the fractional examples in the previous section were around a point in two dimensions, or around an edge in three dimensions. We will generalize them into cuts around $(d-2)$-simplices.

We introduce variable x_i^{d-2} corresponding to the i-th $(d-2)$-simplex e_i which is not on the boundary of the convex hull of $\mathcal{A}$. x_i^{d-2} takes value 1 if e_i appears in the triangulation, and 0 otherwise. 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[2] of d-simplex t_j around e_i, which is normalized so that the perigon is equal to 1.

Proposition 4.4 *The following equation holds for each $(d-2)$-face not on the boundary of the convex hull of $\mathcal{A}$:*

$$x_i^{d-2} = \sum_{j \in S_i} a_{ij} x_j^d \tag{4.4}$$

Proof: If e_i appears in the triangulation and $x_i^{d-2} = 1$, the variables corresponding to the d-simplices that appear in the triangulation and that have e_i as a face, are all equal to 1, and the sum of the dihedral angles around e_i of the d-simplices is equal to the perigon. As angles are normalized so that the perigon is equal to 1, (4.4) holds. Otherwise, all the variables are equal to zero and (4.4) holds. □

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

[2] The dihedral angle in three dimensions. In general dimensions, 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 4.5 *The following inequality is valid for any triangulation:*

$$\textbf{Upper Bound (UB) cut} \sum_{j \in N_i} x_j^d \leq C_i x_i^{d-2} \tag{4.5}$$

Proof: If e_i appears in the triangulation, at most C_i d-simplices out of N_i can appear in the triangulation, namely, $\sum_{j \in N_i} x_j^d \leq C_i$, which is identical to the inequality obtained by substituting $x_i^{d-2} = 1$ in (4.5). Otherwise, all the variable must be equal to zero, and (4.5) holds. □

We revisit the two dimensional example, then the variable corresponding to vertex ($= (d-2)$-face) 0 is equal to 1 ($= \frac{2}{5} \times \frac{1}{2} \times 5$), the maximum cardinality of the odd-cycle of size 5 is 2, and (4.2) is the special case of this inequality.

The inequality (4.3) can also be generalized as follows:

Proposition 4.6 *The following inequality is valid for any triangulation:*

$$\textbf{Lower Bound (LB) cut} \sum_{j \in S_i} x_j^d \geq 3 x_i^{d-2} \tag{4.6}$$

Proof: If e_i appears in the triangulation, at least three d-simplices must exist around e_i, namely, $\sum_{j \in S_i} x_j^d \geq 3$, which is identical to the inequality obtained by substituting $x_i^{d-2} = 1$ in (4.6). Otherwise, all the variable must be equal to zero, and (4.6) holds. □

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 for finding violated constraints is not expensive.

$(d-2)$-face cuts in the example We here revisit the example of 7 points in three dimensions in Figure 4.2, $\mathcal{A}_7$, to see how good $(d-2)$-face cuts are. There are 30 tetrahedra and we assign variables to them as x_1: {1236}, x_2: {1237}, x_3: {1246}, x_4: {1247}, x_5: {1256}, x_6: {1257}, x_7: {1267}, x_8: {1346}, x_9: {1347}, x_{10}: {1356}, x_{11}: {1357}, x_{12}: {1367}, x_{13}: {1456}, x_{14}: {1457}, x_{15}: {1467}, x_{16}: {1567}, x_{17}: {2346}, x_{18}: {2347}, x_{19}: {2356}, x_{20}: {2357}, x_{21}: {2367}, x_{22}: {2456}, x_{23}: {2457}, x_{24}: {2467}, x_{25}: {2567}, x_{26}: {3456}, x_{27}: {3457}, x_{28}: {3467}, x_{29}: {3567}, and x_{30}: {4567}.

We first compare polytope $P_{CO}(\mathcal{A}_7)$ defined as the linear programming relaxation of Formulation (CO) and polytope $P(\mathcal{A}_7)$ defined as the convex hull of the incidence vectors of triangulations, by using *PORTA*, a collection of programs to analyze polytopes [56]. Figure 4.3 and Figure 4.4 are the output of *PORTA*. Figure 4.3 shows the vertices of $P(\mathcal{A}_7)$ and $P_{CO}(\mathcal{A}_7)$. We can see that $P_{CO}(\mathcal{A}_7)$ has only one fractional vertex that corresponds to q in page 45, and both of the polytopes are 10 dimensional. Figure 4.4 shows that $P(\mathcal{A}_7)$ and $P_{CO}(\mathcal{A}_7)$ differ in only one inequality:

$$-x_{21} + x_{25} - x_{30} \leq 0 \tag{4.7}$$

Vertices of $P(\mathcal{A}_7)$:

```
DIM = 30
CONV_SECTION
1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0
1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
EMD
DIMENSION OF THE POLYHEDRON : 10
```

Vertices of $P_{CO}(\mathcal{A}_7)$:

```
DIM = 30
CONV_SECTION
1/2 1/2 0 0 1/2 1/2 0 0 0 0 0 0 1/2 1/2 1/2 1/2 0 1/2 1/2 0 0 0 0 0 1/2 1/2 1/2 1/2 0 1/2 0
1 1 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0
1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0
1 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0 0 0 1
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1 1 0 0 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 1 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 1 1 0 0 0
0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0
0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 1 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0
EMD
DIMENSION OF THE POLYHEDRON : 10
```

Figure 4.3: Vertices of $P(\mathcal{A}_7)$ and $P_{CO}(\mathcal{A}_7)$

We observed that this inequality was satisfied with equation by 10 vertices of $P(\mathcal{A}_7)$, and the convex hull of the 10 vertices was a polytope of dimension 9. Namely, $P(\mathcal{A}_7)$ has an additional facet defined by this inequality.

We then check by using *PORTA* whether $(d-2)$-face cuts are facet-defining or not. Let S_a ($/S_b$) denote the indices corresponding to the tetrahedra congruent to tetrahedron $\{1267\}$ ($/\{1367\}$), respectively. Namely, $S_a = \{7, 16, 21, 28, 30\}$, $S_b = \{12, 15, 24, 25, 29\}$. As the dihedral angle of tetrahedron $\{1267\}$ ($/\{1367\}$) around edge $\{67\}$ is $\frac{1}{5}$ ($/\frac{2}{5}$), (4.4) can be rewritten as:

$$x^1 = \sum_{i \in S_a} \frac{1}{5} x_i + \sum_{i \in S_b} \frac{2}{5} x_i \tag{4.8}$$

As S_b also constitute an odd-cycle of size 5, the UB cut (4.5) around edge $\{67\}$ is:

$$\begin{aligned} & \sum_{i \in S_b} x_i && \leq 2x^1 \\ \rightarrow \quad & \sum_{i \in S_b} x_i && \leq 2\left(\sum_{i \in S_a} \frac{1}{5} x_i + \sum_{i \in S_b} \frac{2}{5} x_i\right) \\ \rightarrow \quad & -2 \sum_{i \in S_a} x_i + \sum_{i \in S_b} x_i && \leq 0 \end{aligned}$$

The LB cut (4.6) around edge $\{67\}$ is:

$$\sum_{i \in S_a \cup S_b} x_i \qquad \geq 3x^1$$

Equations and inequalities to define $P(\mathcal{A}_7)$:

```
DIM = 30

VALID
0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0

INEQUALITIES_SECTION
(  1)                                                                +x19-x20                                                  == 0
(  2) +x1-x2                                                                                                                   == 0
(  3)                    +x7                +x12         -x15-x16                                                               == 0
(  4)                                   +x13-x14                                                                               == 0
(  5)             +x5-x6                                                                                                       == 0
(  6)                           +x10-x11                                                                                       == 0
(  7)          -x4              -x9                 +x14-x15                                                                   == 0
(  8)                                                      +x17-x18                                                            == 0
(  9)                      +x8-x9                                                                                              == 0
( 10)    +x2                  -x9     -x11-x12                                                                                 == 0
( 11)                +x6      -x9         -x12     +x14     +x16     -x18     -x20-x21                                        == 0
( 12)                                              +x14                                  +x23             +x27          +x30 == 1
( 13)                                                    +x16                                      +x25           -x29-x30 == 0
( 14)                                  +x11              +x15+x16              +x20             -x24+x25     -x27-x28        == 0
( 15)                                                                                      +x26-x27                        == 0
( 16)        +x3-x4                                                                                                          == 0
( 17)                                                                                +x22-x23                              == 0
( 18)                                     +x12         -x15-x16                 +x21         +x24-x25                        == 0
( 19)                           +x9                 -x14+x15         +x18                 -x23-x24                        == 0
( 20)                                                    +x15+x16                              -x24+x25         -x28-x29     == 0

(  1)                               -x28          <= 0
(  2)                                    -x30     <= 0
(  3)                         -x25               <= 0
(  4)                                   -x29      <= 0
(  5)     +x20                -x27     +x29      <= 0
(  6)     -x20                                   <= 0
(  7)          -x21                              <= 0
(  8)                    -x24                    <= 0
(  9)                         +x25         -x29-x30 <= 0
( 10) -x18         +x23+x24                      <= 0
( 11)                    -x24         -x28     +x30 <= 0
( 12)               -x23                         <= 0
( 13)          -x21     -x24+x25                 <= 0
( 14)          -x21          +x25           -x30 <= 0
( 15)          +x21                    -x28-x29 <= 0
( 16) +x18                        +x27+x28       <= 1

END
```

Equations and inequalities to define $P_{CO}(\mathcal{A}_7)$:

```
DIM = 30

VALID
0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0

INEQUALITIES_SECTION
(  1)                                                                +x19-x20                                                  == 0
(  2) +x1-x2                                                                                                                   == 0
(  3)                    +x7                +x12         -x15-x16                                                               == 0
(  4)                                   +x13-x14                                                                               == 0
(  5)             +x5-x6                                                                                                       == 0
(  6)                           +x10-x11                                                                                       == 0
(  7)          -x4              -x9                 +x14-x15                                                                   == 0
(  8)                                                      +x17-x18                                                            == 0
(  9)                      +x8-x9                                                                                              == 0
( 10)    +x2                  -x9     -x11-x12                                                                                 == 0
( 11)                +x6      -x9         -x12     +x14     +x16     -x18     -x20-x21                                        == 0
( 12)                                              +x14                                  +x23             +x27          +x30 == 1
( 13)                                                    +x16                                      +x25           -x29-x30 == 0
( 14)                                  +x11              +x15+x16              +x20             -x24+x25     -x27-x28        == 0
( 15)                                                                                      +x26-x27                        == 0
( 16)        +x3-x4                                                                                                          == 0
( 17)                                                                                +x22-x23                              == 0
( 18)                                     +x12         -x15-x16                 +x21         +x24-x25                        == 0
( 19)                           +x9                 -x14+x15         +x18                 -x23-x24                        == 0
( 20)                                                    +x15+x16                              -x24+x25         -x28-x29     == 0

(  1)                               -x28          <= 0
(  2)                                   -x29      <= 0
(  3)               -x23                         <= 0
(  4)                         -x25               <= 0
(  5)     +x20                -x27     +x29      <= 0
(  6)                    -x24                    <= 0
(  7)                                  -x30     <= 0
(  8)     -x20                                   <= 0
(  9)          -x21                              <= 0
( 10)                    +x25         -x29-x30 <= 0
( 11)          +x21                    -x28-x29 <= 0
( 12) -x18         +x23+x24                      <= 0
( 13)          -x21     -x24+x25                 <= 0
( 14)                    -x24         -x28     +x30 <= 0
( 15) +x18                        +x27+x28       <= 1

END
```

Figure 4.4: Equations and inequalities to define $P(\mathcal{A}_7)$ and $P_{CO}(\mathcal{A}_7)$

$$\begin{aligned}
\rightarrow \quad & \sum_{i \in S_a \cup S_b} x_i & \geq 3\,(\sum_{i \in S_a} \frac{1}{5} x_i + \sum_{i \in S_b} \frac{2}{5} x_i) \\
\rightarrow \quad & -2 \sum_{i \in S_a} x_i + \sum_{i \in S_b} x_i & \leq 0
\end{aligned}$$

Thus, the UB and LB cuts are identical in this case and can be rewritten as:

$$-2x_7 + x_{12} + x_{15} - 2x_{16} - 2x_{21} + x_{24} + x_{25} - 2x_{28} + x_{29} - 2x_{30} \leq 0 \quad (4.9)$$

We observed that all the vertices of $P(\mathcal{A}_7)$ satisfied (4.9), 10 of them satisfied (4.9) with equation, and the 10 vertices were the same as in the case of (4.7). Thus, (4.9) defines the one and only facet that is unique to $P(\mathcal{A}_7)$ and is not valid for $P_{CO}(\mathcal{A}_7)$.

We should consider two practical issues; (1) the dihedral angles are not always rational, and (2) the cuts can vary according to the point configuration even if the situation is combinatorially the same. For the former, we can safely perturb the dihedral angles so that the values are rational, provided that the range of perturbation is enough smaller than the smallest positive dihedral angle.

To investigate into the latter, we consider a situation where pentagon {12345} is not regular and the dihedral angles around edge {67} of tetrahedra {1267}, {2367}, {3467}, {4567}, and {1567} are $\frac{18}{100}$, $\frac{19}{100}$, $\frac{20}{100}$, $\frac{21}{100}$, and $\frac{22}{100}$, respectively. In this case, the LB cut and the UB cut are:

$$\begin{aligned}
-\frac{46}{100}x_7 + \frac{11}{100}x_{12} + \frac{29}{100}x_{15} - \frac{34}{100}x_{16} - \frac{43}{100}x_{21} + \frac{17}{100}x_{24} & \\
+\frac{20}{100}x_{25} - \frac{40}{100}x_{28} + \frac{23}{100}x_{29} - \frac{37}{100}x_{30} \quad <= \quad 0 & \\
+\frac{36}{100}x_7 - \frac{26}{100}x_{12} - \frac{14}{100}x_{15} + \frac{44}{100}x_{16} + \frac{38}{100}x_{21} - \frac{22}{100}x_{24} & \\
-\frac{20}{100}x_{25} + \frac{40}{100}x_{28} - \frac{18}{100}x_{29} + \frac{42}{100}x_{30} \quad >= \quad 0 &
\end{aligned}$$

We confirmed by using *PORTA* that both of them were satisfied with equation by the same 10 vertices of $P(\mathcal{A}_7)$ as in the case of (4.7), and therefore the cuts above were facet-defining.

The odd-cycle cuts in the example We should also look into the odd-cycle cuts for comparison. We can consider two odd-cycle cuts corresponding to (4.1), one using point {6} and the other using point {7}:

$$\begin{aligned}
x_1 + x_5 + x_{13} + x_{17} + x_{26} \quad & \leq \quad 2 \\
x_2 + x_6 + x_{14} + x_{18} + x_{27} \quad & \leq \quad 2
\end{aligned}$$

Both of them were satisfied with equation by the same 10 vertices of $P(\mathcal{A}_7)$ as in the case of (4.7), and were facet-defining.

The odd-cycle cut corresponding to (4.2):

$$x_{12} + x_{15} + x_{24} + x_{25} + x_{29} \leq 2 \quad (4.10)$$

was satisfied with equation by only 5 vertices, the convex hull of them was a polytope of dimension 4, and (4.10) just defined a face and did not define a facet of $P(\mathcal{A}_7)$.

```
DIM = 30

CONV_SECTION
(  1) 2/5 2/5   0   0 3/5 3/5 0   0   0   0   0 2/5 2/5 2/5 2/5 0 2/5 2/5 1/5 1/5 0   0   0 2/5 2/5 3/5 3/5 0 2/5 0
(  2) 2/5 2/5   0   0 3/5 3/5 0   0   0   0   0 2/5 2/5 2/5 2/5 0 3/5 3/5   0   0 0 1/5 1/5 2/5 2/5 2/5 2/5 0 2/5 0
(  3) 2/5 2/5 1/5 1/5 2/5 2/5 0   0   0   0   0 2/5 3/5 3/5 2/5 0 3/5 3/5   0   0 0   0   0 2/5 2/5 2/5 2/5 0 2/5 0
(  4) 3/5 3/5   0   0 2/5 2/5 0   0   0 1/5 1/5 2/5 2/5 2/5 2/5 0 2/5 2/5   0   0 0   0   0 2/5 2/5 3/5 3/5 0 2/5 0
(  5) 3/5 3/5   0   0 2/5 2/5 0 1/5 1/5   0   0 2/5 3/5 3/5 2/5 0 2/5 2/5   0   0 0   0   0 2/5 2/5 2/5 2/5 0 2/5 0
(  6)   0   0   0   0   0   0 1   0   0   0   0   0   0   0   0 1   0   0   0   0 1   0   0   0   0   0   0 1   0 1
(  7)   0   0   0   0   0   0 1   0   0   0   0   0   0   0   0 1   1   1   0   0 0   0   0   1   0   0   0 0   0 1
(  8)   1   1   0   0   0   0 0   0   0   0   0   1   0   0   0 1   0   0   0   0 0   0   0   0   0   0   0 1   0 1
(  9)   0   0   0   0   1   1 0   0   0   0   0   0   0   0   0 0   1   1   0   0 0   0   0   1   1   0   0 0   0 1
( 10)   1   1   0   0   0   0 0   0   0   0   0   1   0   0   0 1   0   0   0   0 0   0   0   0   0   1   1 0   1 0
( 11)   1   1   0   0   0   0 0   0   0   0   0   1   1   1   1 0   0   0   0   0 0   0   0   0   0   0   0 1   0 0
( 12)   0   0   0   0   0   0 1   0   0   0   0   0   0   0   0 1   0   0   0   0 1   0   0   0   0   1   1 0   1 0
( 13)   0   0   0   0   1   1 0   0   0   0   0   0   0   0   0 0   0   0   1   1 0   0   0   0   0   1   1 0   0 0
( 14)   0   0   0   0   1   1 0   0   0   0   0   0   0   0   0 0   1   1   0   0 0   1   1   0   0   0   0 0   0 0
( 15)   0   0   0   0   0   0 1   0   0   0   0   0   1   1   1 0   0   0   0   0 1   0   0   0   0   0   0 1   0 0
( 16)   1   1   0   0   0   0 0   1   1   0   0   0   1   1   0 0   0   0   0   0 0   0   0   0   0   0   0 0   0 0
( 17)   0   0   0   0   1   1 0   0   0   0   0   0   0   0   0 0   0   0   0   0 1   0   0   0   1   0   0 1   0 1
( 18)   0   0   1   1   0   0 0   0   0   0   0   0   1   1   0 0   1   1   0   0 0   0   0   0   0   0   0 0   0 0
( 19)   1   1   0   0   0   0 0   0   0   1   1   0   0   0   0 0   0   0   0   0 0   0   0   0   0   1   1 0   0 0
( 20)   0   0   0   0   0   0 1   0   0   0   0   0   0   1   1 0   1   1   0   0 0   0   0   1   0   0   0 0   0 0
( 21)   0   0   0   0   1   1 0   0   0   0   0   0   0   0   0 0   0   0   0   0 1   0   0   0   1   1   1 0   1 0

END
```

Figure 4.5: Vertices of $P_{CO}(\mathcal{A}_7)$ after applying the odd-cycle cut (4.10)

Further, we added (4.10) to $P_{CO}(\mathcal{A}_7)$ and obtained another polytope with 5 fractional vertices (Figure 4.5).

If we afford to spend more time for computation, we can tighten (4.10) by *lifting* (see [51], for example). The lifting procedure for an odd-cycle is; (1) find a variable that is connected to all the variables of the inequality in the intersection graph, in other words, find a tetrahedron that intersects all the tetrahedra in the cut, and (2) set the coefficient for the variable to be equal to the right hand side value of the cut. The procedure should be repeated until we can find no additional variable. The resulting cut can vary according to the order we introduce additional variables. For (4.10), we can add the 5 tetrahedra that have vertex {6} and intersect edge {67}, such as x_3: {1246}.

$$x_{12} + x_{15} + x_{24} + x_{25} + x_{29} + 2x_3 + 2x_8 + 2x_{10} + 2x_{19} + 2x_{22} \leq 2 \qquad (4.11)$$

This lifted inequality was satisfied with equation by the same 10 vertices of $P(\mathcal{A}_7)$ as in the case of (4.7), and was facet-defining.

Fractional cut

When the objective function is unweighted, we can also add a cutting plane that cut off the fractional part of the current value of objective function C_f:

$$\begin{array}{ll} \sum_i x_i^d \geq \lceil C_f \rceil & \text{for minimization problems} \\ \sum_i x_i^d \leq \lfloor C_f \rfloor & \text{for minimization problems} \end{array} \qquad (4.12)$$

We can update this constraint if the lower bound (/upper bound) is updated during the branch and bound procedure. It can further bring additional $(d-2)$-face cuts to be applied.

Table 4.3: 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

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 unweighted, we also add the fractional cut in (4.12).

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 (*OSL*) [54] for the branch and bound. Table 4.3 shows the results. Each row corresponds to the number of the UB cuts, the number of the LB 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, the UB cuts, the LB cuts, both of the two kinds of cuts, and both of them plus the fractional cut, respectively.

We can observe that the LB cuts are especially effective, whereas the UB 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 observations

We did further experiments on these cutting planes in the previous section to obtain the minimum and maximum cardinality triangulations of regular and quasi-regular

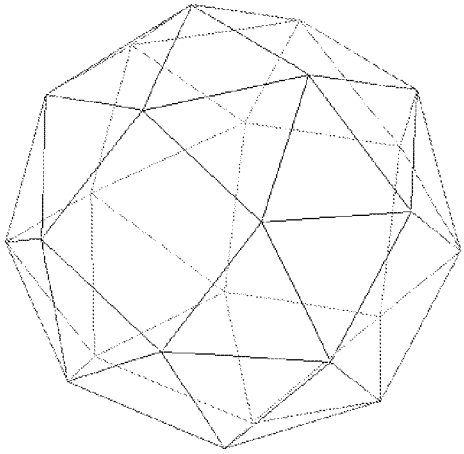

Figure 4.6: The snub cube drawn with *kaleido* [41]

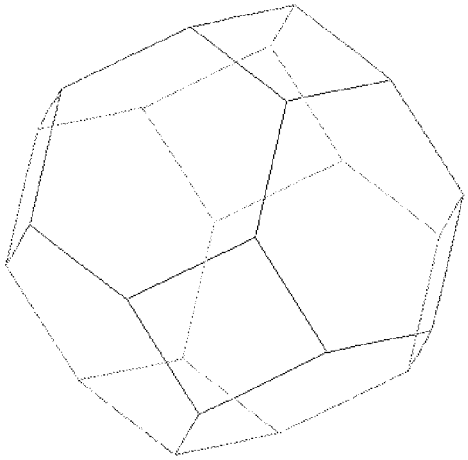

Figure 4.7: The truncated octahedron drawn with *kaleido* [41]

polytopes, which we will look into precisely in Section 5.2.

One of the most successful cases is the minimum cardinality triangulation of the snub cube (Figure 4.6). We had 46 fractional values after solving the linear programming relaxation, and applied 6 LB cuts. Then we obtained an integral solution.

We mentioned in the previous section that the LB cuts worked better, but we also observed an example where only the UB cuts were effective. Figure 4.8 shows an example that we encountered when we obtained the maximum cardinality triangulation of the dodecahedron (Figure 5.5). We had six tetrahedra sharing an edge E (Figure 4.8 left). In the figure, two points T_1, B_1 and E are coplanar, and so as T_2, B_2 and E, and T_2, B_2 and E.

The tetrahedron defined by T_1, B_2 and E (denoted as $\{T_1 B_2\}$) share a facet with $\{B_2 T_3\}$. Also, $\{B_2 T_3\}$ share another facet with $\{T_3 B_1\}$. Thus, the six tetrahedra in the figure satisfy the cocircuit form constraints if they have the same weight. Actually we had a fractional solution of weight $\frac{2}{9}$ for the six tetrahedra, and $\frac{4}{9}$ for E.

Here, $\frac{2}{9}+\frac{2}{9}+\frac{2}{9}+\frac{2}{9}+\frac{2}{9}+\frac{2}{9} = \frac{4}{9}\times 3$, and the corresponding LB cut does not cut off the fractional solution. We, however, can find two UB cuts, for $\{B_1 T_2\}\{B_2 T_3\}\{B_3 T_1\}$, and for $\{T_1 B_2\}\{T_2 B_3\}\{T_3 B_1\}$, where $\frac{2}{9}+\frac{2}{9}+\frac{2}{9} \not\leq \frac{4}{9}$. It is important that the traditional clique cuts cannot be applied to this situation.

Table 4.4 shows the $(d-2)$-face cuts applied in the first iteration in order to obtain the maximum cardinality triangulation of the truncated octahedron (Figure 4.7). We can see that the sum of the fractional values of the tetrahedra that were involved in a cut is very small, and the traditional cuts such as clique cuts and odd-cycle cuts cannot be applied.

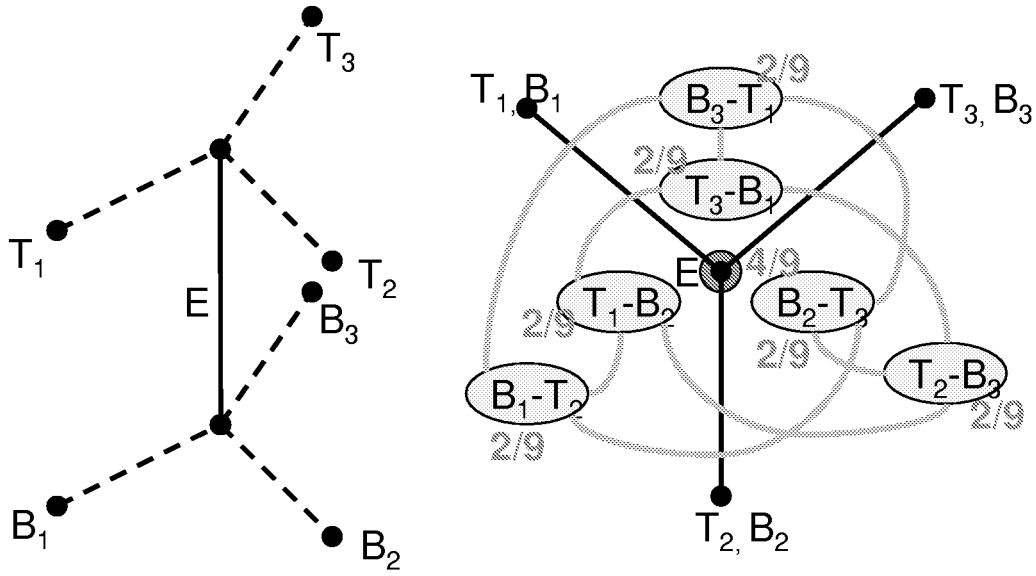

Figure 4.8: An example where UB cuts are effective

Table 4.4: $(d-2)$-face cuts applied for the *fractional* maximum cardinality triangulation of the truncated octahedron in the 1st iteration

UB/LB	Sum of tets.	Edge frac.	Tets/Edge	Constraint
LB	0.636464	0.245102	2.596728	≥ 3
LB	0.331867	0.121826	2.724102	≥ 3
LB	0.139087	0.053945	2.578339	≥ 3
LB	0.597477	0.211438	2.825783	≥ 3
LB	0.223356	0.088990	2.509907	≥ 3
LB	0.136315	0.045644	2.986492	≥ 3
LB	0.212868	0.084624	2.515457	≥ 3
LB	0.723793	0.244189	2.964070	≥ 3
LB	0.567697	0.221794	2.559575	≥ 3
LB	0.420413	0.167228	2.514019	≥ 3
LB	0.452749	0.179517	2.522038	≥ 3
LB	0.269799	0.097988	2.753401	≥ 3
LB	0.368873	0.138173	2.669653	≥ 3
LB	0.120992	0.043267	2.796406	≥ 3
LB	0.572800	0.229120	2.500000	≥ 3
LB	0.703570	0.248525	2.830989	≥ 3
UB	0.155438	0.121826	1.275898	≤ 1
UB	0.088643	0.074308	1.192921	≤ 1
UB	0.189819	0.187535	1.012179	≤ 1
UB	0.259264	0.227814	1.138052	≤ 1

4.2.3 Improving the branch and bound procedure

As fractional values can remain after adding the cutting planes, it is also important to improve the branch and bound procedure.

There are several techniques to accelerate the branch and bound procedure, including prioritizing variables, preprocessing, special ordered sets, and many of them are implemented in commercial codes and automatically applied [76]. As commercial codes are tuned to work good for a lot of test problems, we can tune the performance by introducing some domain specific information.

In particular, the choice of branching variables is a major issue. As Formulation (CO) has d-simplices as variables, any branching corresponds to "whether to use $(= 1)$ the d-simplex in the triangulation or not $(= 0)$". Apparently, the search space are not reduced so much in the "$= 0$" case (we will call this case *forgetting* the d-simplex) as in the "$= 1$" case (we will call this case *fixing* the d-simplex). Consequently, the branch and bound tree gets unbalanced.

We then consider improving the branch and bound procedure by introducing variables with which the search space can be divided in a balanced way. We again focus on the lower dimensional simplices. Suppose we introduce variables corresponding to edges (1-simplices) and triangles (2-simplices) in three dimensions. Fixing a tetrahedron corresponds to fixing 6 edges, and fixing an edge means less than fixing a tetrahedron. On the other hand, forgetting an edge corresponds to forgetting all the $O(n^2)$ tetrahedra that have the edge as a face, and means quite more than forgetting a tetrahedron. In the same way, forgetting a triangle corresponds to forgetting $O(n)$ tetrahedra.

We can use the equation for $(d-2)$-face (4.4) to introduce variables for edges. For variables for triangles, we can modify the cocircuit form constraints (2.3) so that the two summations have the same value as the variable for the facet;

$$\sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_k}} x_i^d = \sum_{t_i = f_k \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_k}} x_i^d = x_k^{d-1}$$

It is still open (1) how we can define the variables for simplices whose number of dimensions are lower than $d-2$ in general dimensions, and (2) which number of dimensions is the most effective to balance the branch and bound tree.

Computational experiments

We examined the effectiveness of the variables for lower dimensional simplices through the maximization of the cardinality with 20 points uniformly distributed in a unit cube. As we used the mixed integer programming module of *OSL* [54] and could not control the branching strategy, we just defined all the corresponding variables, passed them to the module, and observed the results.

Table 4.5 shows the results. Although the number of nodes was reduced when the variables for triangles are introduced, the total CPU time always took longer than the

Table 4.5: The effect of introducing the variables for lower dimensional simplices

Variables	B&B nodes	No. of pivotings	CPU time for B&B
Tets	18	3579	26.33
Tets & edges	22	8264	91.47
Tets & triangles	14	35738	414.08
Tets, triangles & edges	42	49811	1099.73

naive case. We can consider two reasons; one reason is that introducing all the possible variables makes the matrix much larger. The other is that introducing variables for edges ($(d-2)$-faces) causes numerical instability because the coefficients of (4.4) are fractional and we used a fixed length format (MPS format) to pass the problem to the solver.

To improve the performance practically, we will need to (1) control the branch and bound procedure of the solver, (2) introduce only the appropriate variables when necessary, and (3) call the solver through APIs so that we can avoid the precision problem with flat files. This means that we need to design and implement a branch and bound code specific to the optimization of triangulations. For example, the record-holding Traveling Salesman Problem (TSP) solver by Applegate et al. [3] does it, combined with a huge computational power of parallel processors. It also branches on cuts not only on variables, that is, it divides the situation to which it can apply a cutting plane, into sub-situations by utilizing the information on the cutting plane [16].

Chapter 5

Applications and Derivatives

In this chapter, we introduce some applications of our IP-based optimization of triangulations. We also introduce some extensions to the formulation in order to cope with structures that are slightly different from triangulations. First several topics are of theoretical interest, and the others are addressing the practical usefulness. For both of them, we can give some novel insights by providing the optimal solution to each problem.

In Section 5.1, we further look into the minimum weight triangulation in two dimensions that we have referred to several times in this thesis. We also consider the squared weight instead, as an alternative objective function. In Section 5.2, we apply our IP-based approach and obtain the minimum and maximum cardinality triangulations of regular and quasi-regular polytopes, which are very interesting three dimensional examples to illustrate our approach. We encounter several difficulties because of the symmetry and the degeneracy of the polytopes. We introduce how to handle the difficulties, including implementation issues. In Section 5.3, we consider *dissection*, which is a slightly general concept than triangulation. The properties of dissection is not known compared with those of triangulation, and one of them is the cardinality. We extend and apply our IP-based approach to obtain the maximum and minimum cardinality of dissections. In Section 5.4, we obtain data dependent triangulations, which are used to interpolate a function or a distribution of data. All the existing studies are focusing on the two dimensional cases and applying local search based on flips, and we introduce a different approach that obtain the optimal solution and can handle higher dimensional cases. In Section 5.5, we focus on a practical problem of quadrilateral mesh generation. Practical instances are very large and heuristics are mainly used for them. We, however, can give the optimal solutions for small instances, and based on them, derive some insights into objective functions and input data. As techniques for three dimensional mesh generation are also required in the fields, we also search for ways to hexahedral mesh generation in Section 5.6.

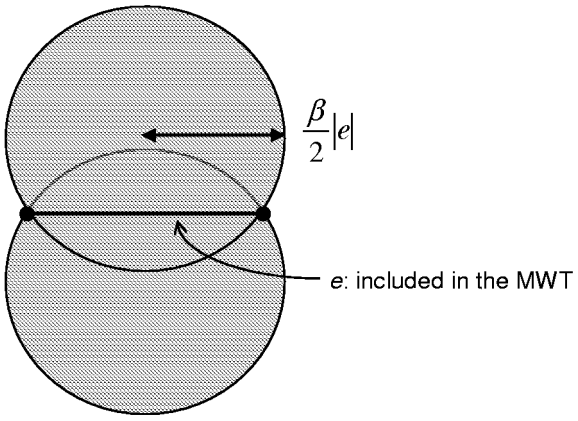

Figure 5.1: The region with no points for an edge e in the β-skeleton

5.1 The Minimum Weight Triangulation

As we mentioned in Section 2.4, obtaining the minimum weight triangulation (MWT) in two dimensions is a longstanding open problem in computational geometry and included in Garey and Johnson's list of problems as neither known to be NP-complete nor known to be solvable in polynomial time. Only when the point set is in a convex position, dynamic programming can obtain the MWT in $O(n^3)$ time.

On the other hand, subgraphs contained in the MWT, called *skeletons*, can be computed in polynomial time by making use of geometric properties [14, 75, 15, 23]. There are two kinds of such skeletons. One is *LMT-skeleton* which is a set of locally minimum edges in a sense that each of them is the shortest possible diagonal for all the pairs of triangles that share the edge. LMT-skeleton can be obtained in $O(n^3 \log n)$ time [14]. The other is *β-skeleton* which is a set of edges that have no other points in two disks that pass through both ends of the edge and have the $\frac{\beta}{2}$ times longer radii than the edge (Figure 5.1). β-skeleton with $\beta \geq 1.17682$ was proved to be contained in the MWT in [13]. These skeletons work fine if the number of connected components of the skeleton is bounded. Especially when the skeleton is a connected subgraph of the MWT, we can obtain the MWT in $O(n^3)$ time by using dynamic programming. Unfortunately, there exist cases where the skeletons are highly disconnected.

We thus applied an IP approach based on the stable set problem of line segments in [44], and also considered in Section 2.4 as an instance of the optimization of triangulation. In Table 3.1 and 3.2, we presented the computational results of the MWT in two dimensions, and the MWT in three dimensions by extending the notion of weight from the edge length in two dimensions to the triangle area in three dimensions. Different from skeletons, the performance of our IP approach is independent of the point configuration. Further, so far we have never encountered the cases where we cannot obtain the MWT by just solving the linear programming relaxation, namely, in weakly

polynomial time.

In two dimensional cases, we can utilize skeletons and reduce the candidate triangles in preprocess. β-skeleton is a subset of the edges in Delaunay triangulation and can be always obtained in $O(n \log n)$ time. Thus, if we focus on the MWT, we could solve larger instances faster than the result in Table 3.1.

5.1.1 The minimum squared weight triangulation

Instead of the lengths of the edges in the triangulation, we can consider the sum of the squared edge lengths as the objective function to be minimized. This corresponds to the objective function in Laplace smoothing,[1] one of the mesh optimization techniques [42]. In the meshing community, the term "optimization" means node-movement strategies for improving the quality of mesh that are based on a certain objective function. Namely they are trying to move the geometry by optimizing – the local minimum, precisely – a nonlinear and continuous objective function under the fixed topology, whereas we are trying to improve the topology as a combinatorial optimization problem under the fixed geometry.

It is interesting to optimize the same objective function as mesh optimization under the fixed geometry. Although the skeletons are not available any more, we can obtain the minimum squared weight triangulation (MSWT) by using our IP-based approach, just by replacing the objective function assigned to each triangle with the sum of the squared lengths of the three edges (facets) of the triangle.

Figure 5.2 shows the distribution of the sum of the edge lengths of the MWT, the MSWT, and Delaunay triangulation of randomly generated 50 (/100) points in a unit square out of 10 trials. Figure 5.3 shows the distribution of the sum of the squared edge lengths under the same settings. We can hardly see the difference between the MWT and the MSWT in both of the figures. Actually, we further experimented and obtained only 73 instances where the MWT and the MSWT were different, out of 100 randomly generated instances of 100 points. Figure 5.4 shows the difference between the MWT and the MSWT in one of the instances. We can only observe slight differences near the boundary, where the MSWT has diagonals whose difference in length is smaller than those in the MWT.

5.1.2 Remarks

As in the cases with skeletons for the MWT, algorithms in computational geometry are tuned for particular problems, and sometimes cannot be applied to even slightly different problems. On the other hand, IP-based approaches are flexible, can cope with such cases, and give some insights on them, although it can only handle small instances.

[1] The operation results in a local movement of a point to the gravity center of neighbors that are connected to the point by edges [35].

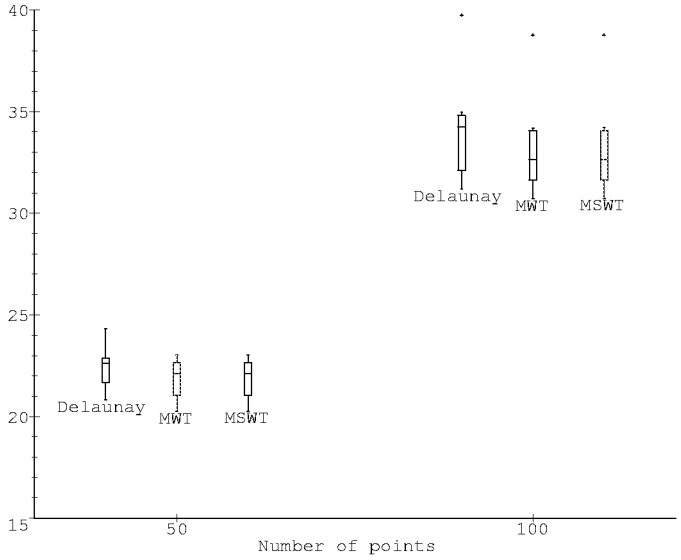

Figure 5.2: Weight of triangulations in two dimensions

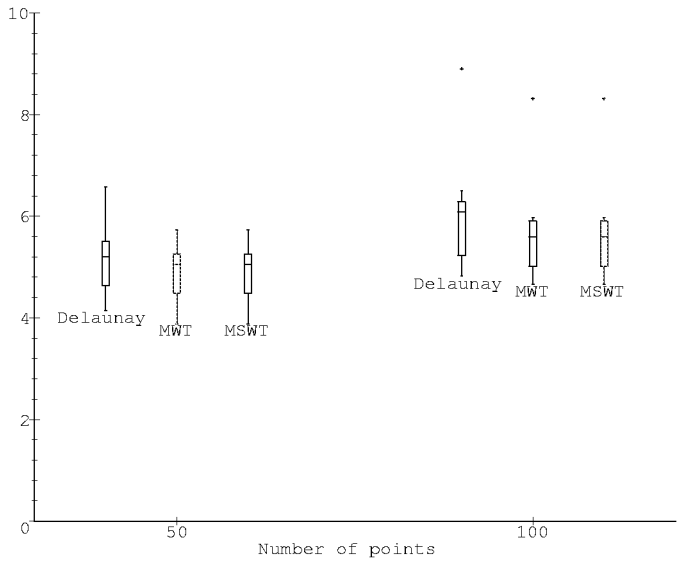

Figure 5.3: Squared weight of triangulations in two dimensions

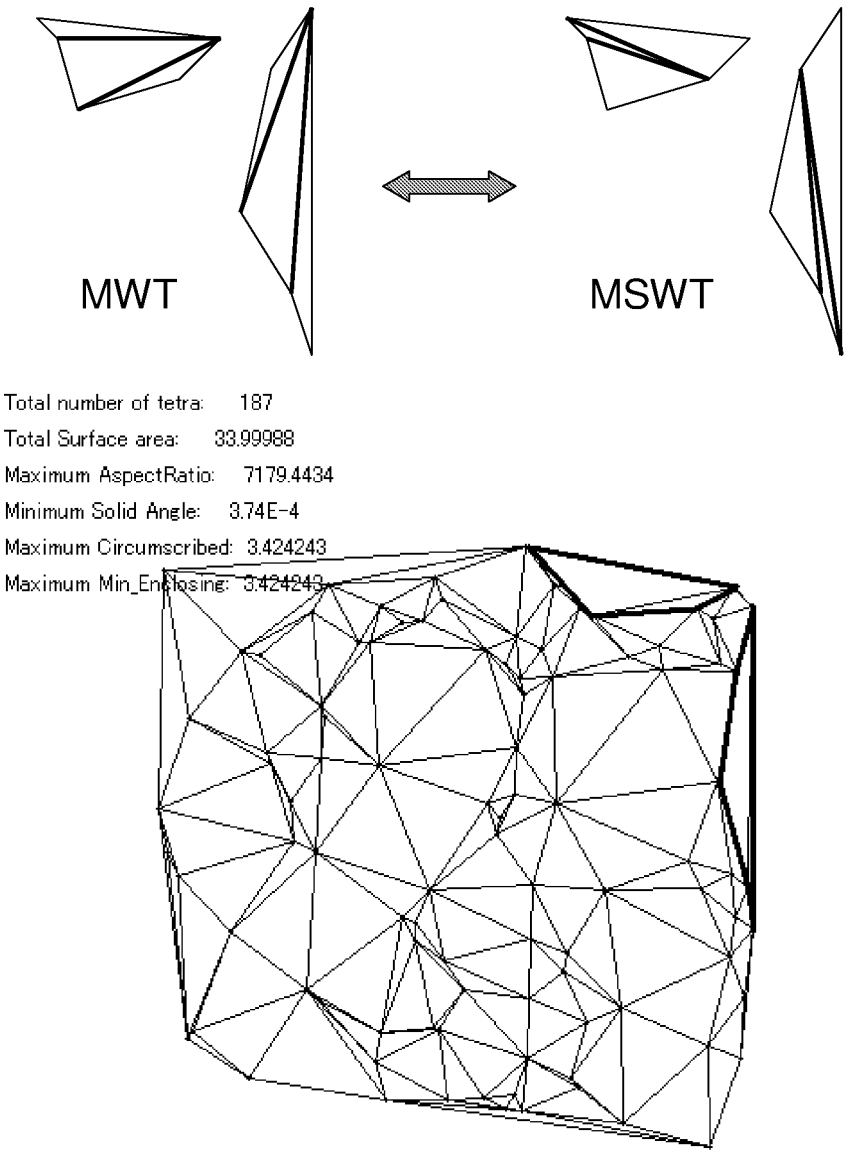

Figure 5.4: The difference between the MWT and the MSWT

The main focus of this thesis is on triangulations in three dimensions, but it is important to note that our IP-based approach can contribute to this type of analysis, even in two dimensions.

5.2 Regular and Quasi-regular Polytopes

In this section, we investigate the minimum and maximum cardinality triangulations of regular and quasi-regular polytopes, which are very interesting three dimensional examples to illustrate our approach. They have a lot of symmetry and degeneracy that cause problems both in theory and implementations. Further, all the vertices are located on a sphere and therefore all the triangulations are Delaunay triangulations. In other words, Delaunay triangulation means nothing for them. Thus, other criteria such as the maximum or minimum cardinality are important. We should notice that De Loera is also working on this topic [18] and we are inspired by the discussion with him.

A polytope is said to be *regular* if its facets are regular and equal, while its vertices are all surrounded alike [17, 66]. There are five regular polytopes, the tetrahedron, the octahedron, the cube, the icosahedron, and the dodecahedron. A quasi-regular polytope is defined as having regular faces but relaxed to have more than one kind of regular polygon, in other words, the cones defined by facets sharing vertices are all congruent [17, 66]. There are thirteen quasi-regular polytopes.

If we go on to obtain the maximum and minimum cardinality triangulations of those polytopes, we encounter two difficulties caused by the symmetries of the polytopes. One is that, most of them have irrational numbers in their coordinate values and it is difficult to handle them in a numerically correct way.

The other is that, even if we apply cutting planes such as $(d-2)$-face cuts we introduced in Section 4.2.2, the fractional solution just slips away by rotating into another position with the same geometry, and the cuts never get effective until we apply them to all the positions that have geometrically the same structure.

5.2.1 Point configurations of regular and quasi-regular polytopes

We first look into the problem on the coordinate values of the vertices. Based on Formulation (CO) in page 18, we require only one geometric predicate: *Which side of hyper plane $\mathcal{H}$ defined by $\mathbf{d}$ points, point $\mathbf{p}$ is located?* Here, the problem is that regular and quasi-regular polytopes have irrational numbers in their coordinates [17]. For example, the dodecahedron (Figure 5.5) has the golden ratio $\frac{1+\sqrt{5}}{2}$ in the coordinates of some of its vertices. Namely, a naive implementation using the floating point data types cannot handle them exactly. Further, a perturbation can bring wrong results as in the example of Figure 5.6.

It is possible to perturb the vertices so that the face lattice of the dodecahedron is preserved, in other words, so that the coplanarities of five points on each facet are

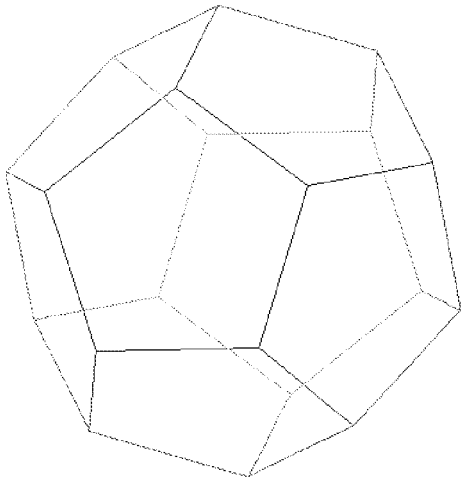

Figure 5.5: The dodecahedron drawn with *kaleido* [41]

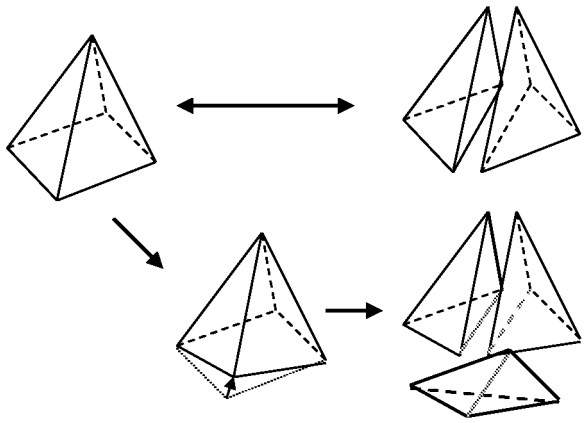

Figure 5.6: An example that a perturbation changes the maximum cardinality

preserved, but it seems impossible to preserve all the coplanarities that the dodecahedron has inside. In other words, to realize the oriented matroid of the dodecahedron, we need to allow irrational numbers in coordinate values. This is highly related with Universality Theorems of oriented matroids.

Universality Theorems

There are many versions of Universality Theorems of oriented matroids [61], and we will refer here a simple version that fits our current interest.

Theorem 5.1 *[61]*
There is a polynomial algorithm that takes as input a system Ω of polynomial equations and strict inequalities with integer coefficients and produces an oriented matroid $\mathcal{M}(\Omega)$

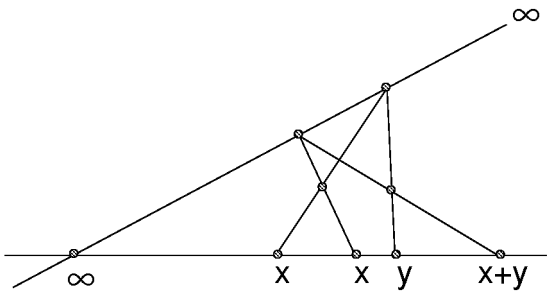

Figure 5.7: An example of Von Staudt construction (addition)

such that the realizability problem for $\mathcal{M}(\Omega)$ *is equivalent to the solvability problem of* Ω.

Intuitively, this theorem says that, we can construct an oriented matroid whose realization is equivalent to the solution of a system of polynomial equations. For example, we can construct an oriented matroid corresponding to $x^2 = 2$ whose solution is $x = \sqrt{2}$, and the realization of oriented matroid surely has at least one irrational number. The following is a brief sketch of the construction.

The outline of Theorem 5.1

- Given a system of polynomial equations Ω, we can obtain *Shor's normal form* $S(\Omega)$

$$1 < x_1 < x_2 < \ldots < x_n$$
$$x_i + x_j = x_k \quad (i \leq j < k)$$
$$x_i \cdot x_j = x_k \quad (i \leq j < k)$$

  that is equivalent to Ω ([70])
- For each constraint above, we can apply a *von Staudt construction* (see Figure 5.7 for example) that geometrically represent an addition or a multiplication.
- The realization space of the configured oriented matroid is equivalent to the solution space of Ω.

We have not proven yet that the oriented matroid of regular and quasi-regular polytopes cannot be realized using rational coordinates. But Theorem 5.1 corresponds to point configurations in two dimensions, and three dimensional cases can surely be more difficult. Hence we should look for alternative ways.

Implementation issues

One naive way to cope with irrational coordinates is to use inexact data type and to define an error bound. Namely, we calculate all the time in an inexact way, and if the

distance between a point and a plane is smaller than the error bound, we assume that the point is on the plane. This method has no theoretical background on how to give the error bound. Fortunately, the point configurations of regular and quasi-regular polytopes have no delicate cases where a point is very close to a plane but not on the plane, and the method above works correctly if the error bound is appropriately defined.

An alternative way with a theoretical background is to use a data type that can handle irrational numbers. The data type `real` of *LEDA* [45], a library of data types and algorithms of combinatorial computing, is one candidate.

> It supports exact computation with k-th roots for arbitrary natural number k, the rational operators $+$, $-$, $*$, and $/$, and the comparison operators $==$, $!=$, $<$, $\leq$, $\geq$, and $>$. [48]

With `real`, the system first computes the value by using a fixed precision floating point data type such as `double`. Only when the situation is delicate and the fixed precision data type can give a wrong answer, it uses an arbitrary precision floating point data type `bigfloat` with the bit length that is theoretically guaranteed to give the correct answer. The bit length is derived from the degree of the acyclic directed graph that represents the symbolic expression of the value to be evaluated [48].

The following is a sample `c++`-code to see how `real` works. It always gives the correct answer "$z = 0$", but it also preserves the initialization information and the computation gets slower and slower, as the iteration proceeds, the graph expression gets complicated, and the required bit length gets longer.

```
real tau = (sqrt((real)5)+1)/2;     // a solution to x^2-x-1=0
for (i = 0; i < 20; i++) {
    real z = tau*tau-tau-1;          // z should be zero
    printf("%f SIGN:%d\n", z.to_double(), z.sign());
    tau = tau*tau-1;                // recursive definition
}
```

Fortunately, the geometric computation we require is simple as we mentioned before, and `real` is sufficient for handling the coordinates of regular and quasi-regular polytopes. Further, *LEDA* is supported to be used with *CGAL* [12], which is a library for handling geometric objects and we actually use in our experiments.

We used `real` to check the coplanarity of all the $\binom{20}{3} \times (20 - 3)$ pairs of planes and points for the dodecahedron, which include both obviously non-coplanar cases and degenerate cases, and it took about 160 times longer than using the naive error bound method that we described above. This performance of `real` compared with `double` is comparable to other reports [29]. In our cases, the computation is far small compared with the mathematical programming part, and we can say that the use of `real` is a good choice to cope with the irrational coordinates of regular and quasi-regular polytopes.

Delaunay triangulation We should also note that just to obtain Delaunay triangulations of regular and quasi-regular polytopes is very difficult. To be precise, as all the points are on a sphere for those polytopes and all the triangulations are Delaunay triangulations, we just need to obtain *a triangulation*. It is possible to obtain one if we implement by ourselves an algorithm that is robust against degeneracy and also can cope with irrational numbers by using *LEDA*, but we can not make use of existing public software for Delaunay triangulation.

To our knowledge, there is no implementation available that can accept irrational values such as "(sqrt(5)+1)/2" as input, and we need to give approximated coordinate values of a fixed length. Further, most of the packages just halted claiming that there were more than four points on a sphere, and did not give a result. Even when we obtained a result by perturbing the points, the output triangulation often had flat (degenerate) tetrahedra inside the convex hull, and the elimination of them resulted in a dissection, but not a triangulation.

5.2.2 Cutting planes and regular polyhedral groups

We know focus on the other problem, that is; when we apply cutting planes such as $(d-2)$-face cuts that we introduced in Section 4.2.2, the fractional solution just slips away by rotating into another position with the same geometry, and the cuts never get effective until we apply them to all the positions that have geometrically the same structure.

Further during the branch and bound procedure, the similar situation can occur. Assuming that one d-simplex is not used in the triangulation has very little effect on the overall search space because there are a lot of geometrically symmetric alternatives. Consequently, the branch and bound procedure takes a lot of branching nodes and time.

It is still open if we can reduce the number of variables (d-simplices) in the preprocess by considering the symmetry of the point configuration. We, however, can cope with the problem on cutting planes by making use of a property of regular and quasi-regular polytopes.

Regular polyhedral groups

As studies on polyhedral groups have a long history, we review some of the results based on [17].

Given a symmetrical figure, we can consider a congruent transformation that corresponds to permuting the component elements. Such a congruent transformation is called a symmetry operation, and all the symmetry operations of a figure form a group, called the symmetry group. Here we focus on finite groups, and it is known that a finite symmetry group is a rotation group.

We then look into the rotation groups of the regular polytopes. The centroid of the

polytope configure axes by being joined with the vertices, the mid-edge points, and the centroids of the facets. Then we can consider three kinds of rotations around those axes. By enumeration, we see that the tetrahedral group has order 12, the octahedral group, which is also the rotation group of the cube, has order 24, and the icosahedral group, which is also the rotation group of the dodecahedron, has order 60.

For two dimensional regular p-gons, we can consider the cyclic groups of rank p that correspond to rotations in two dimensions, and the dihedral groups of rank $2p$ that also consider the reflections.

These five groups; the tetrahedral group, the octahedral group, the icosahedral group, the cyclic group, and the dihedral group are called *the regular polyhedral groups*, and it is known that they are the only finite rotation groups in three dimensional Euclidean space.

Thus, every quasi-regular polytope also has its rotation groups among the regular polyhedral groups. As these rotations correspond to the permutations of indices of vertices, we can efficiently enumerate all the cuts symmetric to the one violated by the current fractional solution.

5.2.3 Computational experiments

De Loera gave the minimum and maximum cardinalities of some of regular and quasi-regular polytopes [18]. We did further experiments on all the regular and quasi-regular polytopes for the minimum and maximum triangulations (Table 5.2.3). To obtain the coordinate values (in a fixed length format) of the vertices of the polytopes, we used program *kaleido* [41]. We applied $(d-2)$-face LB cuts and the fractional cut in Section 4.2.2.

We could obtain a little more results than De Loera's, but still there are a lot to be done. The points are in a convex position and there is no non-empty tetrahedron to be ignored, and currently we can not handle polytopes with more than 30 vertices. Further, the degeneracy of the linear programming problem exists both in primal and dual as we mentioned in Section 3.4, and simplex methods are very slow. The situation would change if a good implementation of interior point methods becomes available.

5.2.4 Remarks

Using regular and quasi-regular polytopes, we looked into the degeneracy and the symmetry of the problems, not only in geometry but also in mathematical programming. As we observed in Section 3.4, simplex methods are not good at degenerate problems. Further, it is well known that mixed integer programming packages are not good at symmetric problems, and it is important to introduce the domain specific knowledge that cancels the symmetry. Enumerating effective cutting planes by using regular polyhedral groups is one of such techniques, but it is not sufficient and techniques for the branch and bound procedure are quite important and required to solve large

Table 5.1: The minimum and maximum cardinality of triangulations of regular and quasi-regular polytopes

polytope(# vertices)	Max/Min	De Loera's	Our results				
				Frac.	Cuts(itr.)	Frac. aft. cuts	B&B nodes
Tetrahedron	Min	N.A.	1	0	–	–	–
(4)	Max	N.A.	1	0	–	–	–
Octahedron	Min	N.A.	4	0	–	–	–
(6)	Max	N.A.	4	0	–	–	–
Cube	Min	N.A.	5	0	–	–	–
(8)	Max	N.A.	6	0	–	–	–
Cuboctahedron	Min	13	13	0	–	–	–
(12)	Max	17	17	0	–	–	–
Icosahedron	Min	15	15	0	–	–	–
(12)	Max	20	20	0	–	–	–
Truncated	Min	10	10	0	–	–	–
Tetrahedron(12)	Max	13	13	0	–	–	–
Dodecahedron	Min	23	23	45	0+1(1)	44	1
(20)	Max	36	36	107	2+1(2)	246	2
Rhombicuboctahedron	Min	35	35	0	–	–	–
(24)	Max	N.A.	56-60	971	113+1(19)	1135	N.A.
Snub Cube	Min	N.A.	38	46	6+1(1)	0	–
(24)	Max	N.A.	74	1264	25+1(1)	515	3
Truncated Cube	Min	27	25	0	–	–	–
(24)	Max	48	48	0	–	–	–
Truncated	Min	27	27	0	–	–	–
Octahedron (24)	Max	N.A.	49	1031	124+1(14)	1089	6
Icosidodecahedron	Min	45	45	833	0+1(1)	0	–
(30)	Max	N.A.	77 - 81	2149	87+1(6)	2444	N.A.
Truncated	Min	N.A.	N.A.	–	–	–	–
Cuboctahedron(48)	Max	N.A.	N.A.	–	–	–	–
Rhombicosidodecahedron	Min	N.A.	N.A.	–	–	–	–
(60)	Max	N.A.	N.A.	–	–	–	–
Snub Dodecahedron	Min	N.A.	N.A.	–	–	–	–
(60)	Max	N.A.	N.A.	–	–	–	–
Truncated	Min	N.A.	N.A.	–	–	–	–
Dodecahedron(60)	Max	N.A.	N.A.	–	–	–	–
Truncated	Min	N.A.	N.A.	–	–	–	–
Icosahedron(60)	Max	N.A.	N.A.	–	–	–	–
Truncated	Min	N.A.	N.A.	–	–	–	–
Icosidodecahedron(120)	Max	N.A.	N.A.	–	–	–	–

instances.

5.3 Dissections

A dissection of a point configuration $\mathcal{A}$ (Figure 1.2) is a partition of the convex hull of $\mathcal{A}$ into d-simplices whose vertices are the elements of $\mathcal{A}$. A dissection of a d-polytope corresponds to the dissection of its vertex set. Thus, a triangulation is a special case of dissections in a sense that it is a simplicial complex. We will denote a dissection is *mismatching* [20], if it is not a triangulation and there exists a pair of d-simplices whose intersection is non-empty and is not a face of at least one of the two d-simplices. Apparently a dissection can be mismatching only when $\mathcal{A}$ is not in a general position.

In this section, we will focus on dissections of 3-polytopes, and extend our formulation to optimize dissections. The basic idea is that, if we take into account *flat* (degenerate) tetrahedra, whose four vertices are on a plane and thus have been ignored so far, and assign a variable to each of them, then they will cancel the mismatched regions. A mismatched region is a polygon that is triangulated in two different ways on both sides of it. Based on the fact that all the triangulations are connected by bistellar flips in two dimensions, the two triangulations on both side of the mismatched region are connected with a sequence of flips, and the sequence of flips corresponds to a set of flat tetrahedra.

It is important to note that De Loera et.al. are working on dissections of convex polytopes, especially focusing on the minimum and maximum cardinality cases [20]. They also apply the idea of flat tetrahedra for proving some of their theorems. We will utilize some of their theoretical results to consider the formulation.

Lemma 5.1 *[20] All mismatched regions for a dissection of a convex 3-polytope P are convex polygons with all vertices among the vertices of P. Distinct mismatched regions have disjoint relative interiors.*

Proposition 5.1 *[20] The mismatched region of a dissection of a 4-polytope can be a non-convex polyhedron.*

In dimensions more than two, we cannot say that all the triangulations are connected by bistellar flips. Santos presented an example of an isolated triangulation in six dimensions [64]. From this fact and Proposition 5.1, we will limit the following discussions in this section to three dimensional cases.

5.3.1 An integer programming formulation of dissecting convex 3-polytopes

We extend Formulation (CO) in page 18 in order to handle dissections of 3-polytope P, by introducing variables corresponding to the flat tetrahedra that have been ignored in triangulations. There are two realizations of a flat tetrahedron and each of them corresponds to a direction of flips (Figure 5.8). We will distinguish them in the

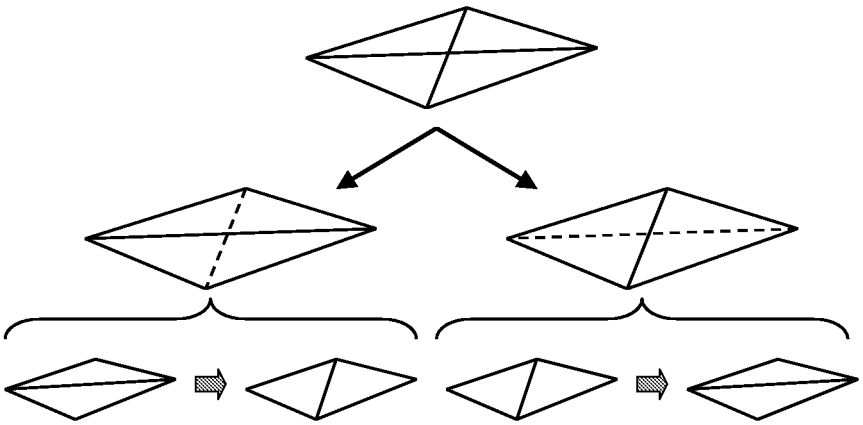

Figure 5.8: Two realizations of a flat tetrahedron

following formulation, according to which side of a facet the other vertex is assumed to be located.

Different from the notations used so far, $\mathcal{A}$ denotes the set of vertices of P. $\mathcal{A}_k$ again denotes a set of points on the plane defined by $(d-1)$-simplex f_k and are not the vertices of f_k. Let $\bar{t}_i^+(/\bar{t}_i^-)$ denote a flat tetrahedron defined by f_k and an element a of $\mathcal{A}_k$ on the positive(/negative) side of f_k, respectively. Variable $\bar{x}_i^+(/\bar{x}_i^-)$ represents if $\bar{t}_i^+(/\bar{t}_i^-)$ is used in the dissection or not, respectively. Also we will use x_i instead of x_i^3 as we are limiting the dimensions to be three.

Formulation (COD):

minimize $\boldsymbol{cx}$

s.t.

$$
\begin{aligned}
x_i &\in \{0,1\} \\
\bar{x}_i^+ &\in \{0,1\} \\
\bar{x}_i^- &\in \{0,1\} \\
&\sum_{t_i=f_k\cup\{a\},a\in\mathcal{A}\cap\mathcal{H}_{f_k}^+} x_i + \sum_{\bar{t}_i^+=f_k\cup\{a\},a\in\mathcal{A}_k} \bar{x}_i^+ \\
&- \sum_{t_i=f_k\cup\{a\},a\in\mathcal{A}\cap\mathcal{H}_{f_k}^-} x_i - \sum_{\bar{t}_i^-=f_k\cup\{a\},a\in\mathcal{A}_k} \bar{x}_i^- \\
&= \begin{cases} 1 & f_k \text{ is on the boundary, oriented inside and } \mathcal{A}_k=\emptyset \\ -1 & f_k \text{ is on the boundary, oriented outside and } \mathcal{A}_k=\emptyset \\ 0 & f_k \text{ is not on the boundary} \end{cases} \\
\sum_i M_{c^*i}x_i &= 1 \quad \text{(for a chamber } h_c^*\text{, only if } conv(\mathcal{A}) \text{ has no simplicial facet)}
\end{aligned}
$$

Proposition 5.2 *For every dissection $\mathcal{D}$ of a convex 3-polytope P, there exists a corresponding solution to Formulation (COD).*

Proof: If $\mathcal{D}$ is a triangulation, we can set all $\bar{x}_i^+$ and $\bar{x}_i^-$ to zero and x to the incidence

vector of the constituent tetrahedra.

If $\mathcal{D}$ has mismatched regions, they are convex polygons with all vertices among the vertices of P from Lemma 5.1. For each mismatched region, we have two triangulations of the convex polygon, one for each side of it. By considering sequences of flips to Delaunay triangulation from the two triangulations, we can make a sequence of flips from one of the triangulations to the other without duplicated flips. Then we set the variables for the flat tetrahedra corresponding to the flips to 1 and obtain a feasible solution to Formulation (COD), combined with the incidence vector of the constituent non-flat tetrahedra. □

It is still open if every integer solution to Formulation (COD) gives a dissection of P, but we think it does.

Conjecture 5.1 *For every integer solution to Formulation (COD), there exists a corresponding dissection $\mathcal{D}$ of a convex 3-polytope P.*

In case the above conjecture is false, we can find the chamber constraints violated by the current solution. Thus iteratively adding the violated constraints as cutting planes, we will obtain the optimal dissection. Fortunately, we encountered no case that was against the conjecture during the computational experiments below.

5.3.2 Computational experiments

We obtained the minimum and maximum cardinality dissections of some of regular and quasi-regular polytopes using Formulation (COD). We did not encounter cases where the obtained integer solutions did not correspond to dissections. Table 5.2 shows the results. As $(d-2)$-face cuts are not valid for dissections, we only applied the fractional cut before proceeding to the branch and bound procedure. As far as we could obtain the optimal solutions, all the polytopes had the same minimum and maximum cardinalities both in dissections and triangulations. Computationally, we often obtained much smaller lower bound or larger upper bound than the cases of triangulations by solving the linear programming relaxation. Further investigation on the fractional solutions would be interesting.

5.4 Data Dependent Triangulations

In this section, we investigate a method to interpolate functions based on triangulations of the domain of definition. We can obtain a better approximation by selecting a specific triangulation for each function, especially when the function has a preferred direction [26], and there has been several studies on obtaining such *data dependent triangulations* [62, 26, 65].

We consider approximating a function F defined over a region in R^d, with a set of piecewise polynomial function g_i defined over each d-simplex t_i of a triangulation

Table 5.2: The minimum and maximum cardinality dissections of regular and quasi-regular polytopes

polytope(# vertices)		Triangulation	Dissection	LP-relaxation	No. frac. tets
Octahedron	Min	4	4		
(6)	Max	4	4		
Cube	Min	5	5		
(8)	Max	6	6		
Cuboctahedron	Min	13	13		
(12)	Max	17	17		
Icosahedron	Min	15	15		
(12)	Max	20	20		
Truncated	Min	10	10		
Tetrahedron(12)	Max	13	13		
Dodecahedron	Min	23	23	14.0	398
(20)	Max	36	–	–	–
Rhombicuboctahedron	Min	35	35		
(24)	Max	56-60	–	61.25	952
Snub Cube	Min	38	38		
(24)	Max	74	74		
Truncated Cube	Min	25	25	21.0	20
(24)	Max	48	–	56.0	166
Truncated	Min	27	–	16.5	361
Octahedron (24)	Max	49	–	–	–

$\mathcal{T}$. We are actually interested in the special cases where the vertices of $\mathcal{T}$ is given as a point set $\mathcal{A}$, and the domain of definition of F equals to $conv(\mathcal{A})$. Further, g_i are limited to the linear function using the values of F at the vertices of t_i, and the only degree of freedom is the choice of triangulation $\mathcal{T}$ of $\mathcal{A}$.

All of the existing studies were limited to two dimensional cases and based on local search using edge flips. Dyn et al. just obtained the local minimum triangulation, and mainly focused on the choice of the flipping criteria [26]. Schumaker tried to get out of local optimals by applying simulated annealing [65], but it can never prove optimality as well known.

By applying our IP-based optimization of triangulations, not only we can obtain the optimal triangulation but also can cope with higher dimensional cases, where flipping do not help a lot as we mentioned in Section 1.2.3. Typical applications of data dependent triangulations are surface representation in computer aided geometric design and approximation of a scattered data set in two dimensions. But we can consider higher dimensional applications such as constructing four dimensional surfaces with the form of three dimensional coordinates and the corresponding temperature [5]. An algorithm suggested in [6] triangulates the domain into tetrahedra, interpolates the function within each tetrahedron, and renders the resulting four dimensional surface, perhaps using contours. Further we can generalize the problem to approximation of higher dimensional functions based on a triangulation of the domain of definition, although we can hardly visualize the result.

We should note that some of the edge flipping criteria require the values of F only at the vertices of the triangulation. It is useful in the cases where the information on

the function is not enough, which may occur when we approximate the function based on observations at scattered points.

In many cases, however, we can assume that more information on F is available, and data dependent triangulations give good approximations in such cases. Our IP-based optimization has advantages when we have enough information on F and can assign cost to each d-simplex. In two dimensions, Dyn et al. recommended in [26] to use the minimum error criterion

$$E(f_{\mathcal{T}}) = [\int_{conv(\mathcal{A})} (f_{\mathcal{T}} - F)^2 dxdy]^{\frac{1}{2}} \tag{5.1}$$

stating that "The numerical tests indicate that it is recommended to use this criterion whenever possible," where $f_{\mathcal{T}}$ is the interpolation within each triangle t_i in $\mathcal{T}$ using the values of F at the vertices of t_i in $\mathcal{T}$. Namely, the most recommended criteria is a summation style function that can be calculated for each triangle, and is congenial to IP-based optimization.

5.4.1 Optimization criteria for data dependent triangulations

We review some criteria introduced in the previous studies, and investigate them from the view point of integer programming.

The minimum roughness criterion $R(F, \mathcal{T})$ is given by

$$R(F, \mathcal{T}) = \sum_i \int_{t_i} [(\frac{\partial F}{\partial x})^2 + (\frac{\partial F}{\partial y})^2] dxdy$$

This evaluate the smoothness of F within each triangle. Rippa proved that Delaunay triangulation is always the optimal triangulation under this criterion [62].

The minimum roughness criterion, however, does not address the error when used for interpolation, and several other criteria have been introduced. Dyn et al. gave four criteria defined on edges, namely pairs of adjacent triangles in triangulations, assuming that the triangulation is lifted to three dimensional space according to the values of F at the vertices in the triangulation.

1. *The angle between normals*

   The angle defined by two planes used for interpolation at the triangles on both side of the edge.

2. *The jump in normal derivatives*

   The product of a unit vector orthogonal to the edge, and the difference of two normal vectors of two planes.

3. *The deviations from linear polynomials*

   The error at the other vertex of the quadrilateral formed by the two triangles based on the interpolation function of one of the triangles.

4. *The distances from planes*

   The distance of the other vertex of the quadrilateral formed by the two triangles from the plane corresponding to the interpolation function of one of the triangles.

The criteria for triangulations are defined by the l_1, l_2-norm or lexicographic ordering of the above criteria for edges. We cannot replicate these criteria with our IP approach, for these are defined for pairs of triangles.

Obviously, all these criteria are only valid in two dimensions, and we note again that Dyn et al. recommended the minimum error criterion (5.1), and used the sum of interpolation errors at uniformly distributed points for evaluating these criteria. It means these criteria have significance only when (1) the information on F is not sufficient and (2) F is defined on R^2.

Schumaker suggested to use the thin spline energy when F is a spline with higher smoothness [65]

$$\sum_i \int_{t_i} [(\frac{\partial^2 F}{\partial x^2})^2 + 2(\frac{\partial^2 F}{\partial x \partial y})^2 + (\frac{\partial^2 F}{\partial y^2})^2] dxdy$$

This is a summation over triangles and can be handled and optimized by IP.

5.4.2 An example of data dependent triangulations

We introduce here an example of data dependent triangulations. We approximated a function used in [26]:

$$\begin{aligned} F^{(1)} &= 3/4\, e^{-1/4(9x-2)^2-1/4(9y-2)^2} + 3/4\, e^{-1/49(9x+1)^2-\frac{9}{10}y-1/10} \\ &\quad +1/2\, e^{-1/4(9x-7)^2-1/4(9y-3)^2} - 1/5\, e^{-(9x-4)^2-(9y-7)^2} \end{aligned}$$

with a triangulation whose vertices are 100 randomly distributed points in a unit square. As the exact definition of F is available, we can approximate the minimum error criterion of triangle t_i:

$$E(t_i) = [\int_{t_i} (f_T - F)^2 dxdy]^{\frac{1}{2}}$$

with the sum of the errors at the finely distributed sampling points, say, 10000 grid points, and use it as the objective function to be minimized by using IP.

Figure 5.9 is a view of F over the domain of definition. Figure 5.10 and Figure 5.11 show Delaunay triangulation and the optimized triangulation for $F^{(1)}$ using the same vertex set, respectively. In Figure 5.11, we can observe many thin triangles where there is a direction in which the gradient of $F^{(1)}$ is stable compared with other directions. This is consistent with the observation in [26].

Delaunay triangulation gave the accumulated error of 0.047278, whereas the optimized triangulation gave the error of 0.040356. We reduced the error by 15%, which is a larger improvement than that in [26], although the vertex sets for triangulations are different. For more objective evaluation, we require common data sets with existing studies on data dependent triangulations, which are not available unfortunately.

It is important that we can now provide the best quality solution for the triangulation-based approximation, and it contributes to the design and evaluation of (1) heuristics which are faster than the optimization, and (2) criteria for them such as those we referred in Section 5.4.1.

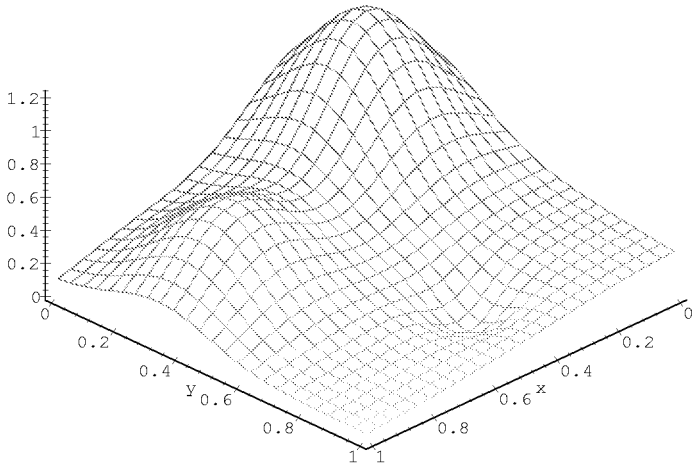

Figure 5.9: $F^{(1)}$ in [26]

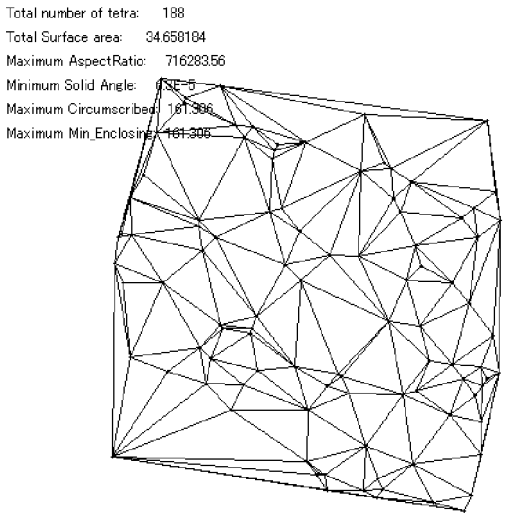

Figure 5.10: Delaunay triangulation

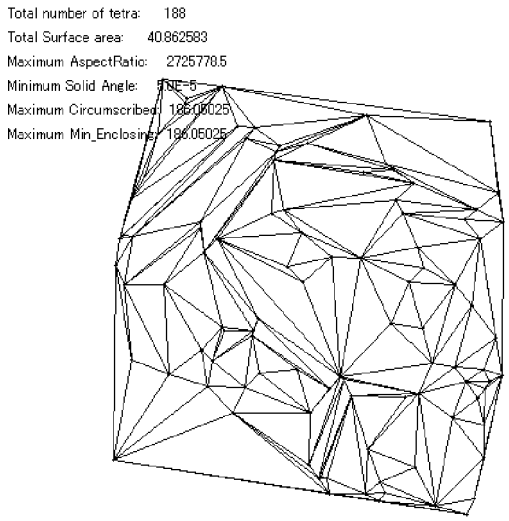

Figure 5.11: The optimal triangulation to approximate $F^{(1)}$

5.5 Quadrilateral Mesh Generation

In this section, we investigate quadrilateral mesh generation from the viewpoint of optimization, assuming that a region to be meshed and a point set are given as input. In particular, we introduce integer programming formulations for obtaining the optimal quadrilateral mesh. Focusing on the cases discussed by Itoh et al. [40], we also examine through computational experiments how far we can improve the quality of mesh by using optimization techniques.

The finite element method (FEM) has been a major analysis technique for engineering applications. From a practical point of view, quadrilateral meshes in two dimensions or hexahedral meshes in three dimensions are preferred to triangular or tetrahedral meshes for reasons of accuracy and efficiency [11]. However, it is well known that quadrilateral or hexahedral meshes of good quality are far more difficult to generate than triangular or tetrahedral meshes. Still, for quadrilateral meshes, several automated mesh generation techniques have been developed.

Techniques for quadrilateral mesh generation can be classified into two categories: direct and indirect approaches [55]. The former includes grid mapping, which maps a lattice grid inside the region to be meshed [39], and the advancing front method, which recursively fills mesh elements starting from the boundary of the given region [77].

Generally speaking, indirect methods consist of three steps: (1) generate a point set within the region to be meshed, (2) obtain the triangulation, and (3) generate a quadrilateral mesh by pairing triangles. Indirect methods are good at controlling the

size, the orientation, and the shape of mesh elements [40], especially when combined with good point generation algorithms such as [68].

Indirect methods are also quite interesting as combinatorial optimization problems. The third step above can be formulated as a matching problem, although most of the previous studies applied greedy heuristics [55]. Further, most of them are based on Delaunay triangulation, which can be obtained efficiently and has various optimal properties in two dimensions such as maximization of the minimum angle [7]. On the other hand, we can optimize triangulations by using integer programming, and it is also possible to extend the framework to cope with quadrilaterals; that is, we can obtain the *optimal* quadrilateral mesh. The extent to which Delaunay triangulation-based meshes can be improved is very interesting.

5.5.1 Indirect methods for quadrilateral mesh generation

We first review indirect methods for quadrilateral mesh generation. For the third step to pair triangles and generate a quadrilateral mesh element, most existing methods use greedy heuristics, which sort the candidate pairs according to some criteria and therefore take $O(N \log N)$ time, where N denotes the number of triangles. Since Delaunay triangulation used in the second step also requires $O(N \log N)$ time[2] , an $O(N \log N)$ time heuristic will be a reasonable choice.

Itoh et al.'s method [40]

In preparation for the computational experiments in Section 5.5.4, we briefly introduce Itoh et al.'s method. To balance multiple requirements for meshes, they consider three functions F_a, F_b, and F_c for quadrilateral mesh elements, that is, for pairs of triangles. F_a evaluates the alignment of a mesh element to a given vector field such as the boundary of the domain and the fluid direction. F_b evaluates the shape of a mesh element that is desired to be square or rectangular. F_c is a positive value for avoiding isolated triangular elements in the final mesh. The objective function V is then defined as the weighted sum of the three functions:

$$V = aF_a + bF_b + cF_c \tag{5.2}$$

Itoh et al. sort all the possible pairs of triangles in descending order of V, and fix the pairs in a greedy way. During the greedy procedure, they update $F_c(p, q)$ from zero to a positive value when a triangular element t_p has only one remaining triangular neighbor t_q, in order to avoid t_p to be an isolated triangular element.

5.5.2 Applying a matching algorithm to quadrilateral mesh generation

We here focus on the third step in indirect methods, namely, generating a quadrilateral mesh by pairing triangles from a given triangulation $\mathcal{T}$ with N triangles. Most of the

[2]The number of points n is $O(N)$, from Euler's formula, and Delaunay triangulation can be obtained in $O(n \log n)$ time.

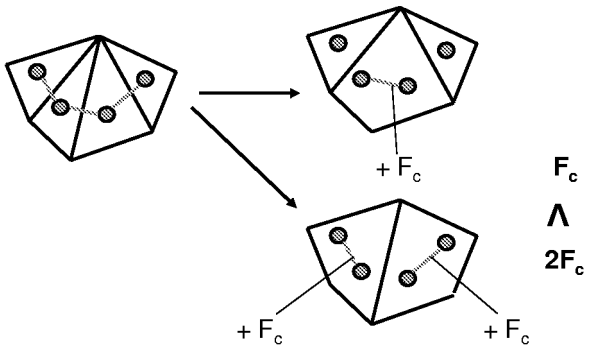

Figure 5.12: Constant F_cs for all the arcs also contribute to reducing the isolated triangles.

existing indirect methods apply greedy procedures that take $O(N \log N)$ time. On the other hand, we can regard the problem as an weighted matching problem in a general graph.

Let $G(V, E)$ denote a graph whose nodes correspond to triangles in $\mathcal{T}$. Arcs are defined between two nodes when the corresponding triangles share an edge. From Euler's formula, $|V| = N, |E| = O(N)$. For the weight to an arc defined between two nodes v_p and v_q, we assign the value of the objective function to be assigned to the quadrilateral element generated by joining triangles t_p and t_q.

In Itoh et al.'s method which we reviewed above, the function F_c is dynamically updated to avoid isolated triangles. We can expect to obtain a similar effect by assigning the constant value of F_c to all the possible pairs of triangles (Figure 5.12), although this does not exactly correspond to the original settings by Itoh et al.

With the settings above, obtaining the maximum weight matching in G gives the optimal quadrilateral mesh that can be obtained from $\mathcal{T}$. The problem is classified as an instance of the weighted matching problem in general graphs, and can be solved in $O(EV \log V)$ [32], which equals $O(N^2 \log N)$ in this case.

5.5.3 Integer programming formulations of quadrilateral mesh generation

We took a triangulation as input in Section 5.5.2, where the total number of edges available for mesh generation, $O(N)$, is quite limited. However, if we could optimize the mesh by taking account of all the $O(N^2)$ edges, the result would be better.

We now focus on obtaining the *optimal* quadrilateral mesh by using integer programming (IP). We investigate the extensions of the formulation of triangulation for handling quadrilaterals, by assigning variables to all the possible quadrilaterals, as we did for triangles in Chapter 2.

Here we extend the formulation in two different ways based on different ideas, then prove that the two extended formulations are equivalent. From now on, we fix the

number of dimensions to two for simplicity, and let variable x_i (not x_i^d as before) be assigned to *triangle* t_i.

Extension based on the underlying triangulation

Any quadrilateral mesh can be transformed into an underlying triangulation by introducing the diagonals of the constituent quadrilaterals. For the underlying triangulation, we can use the cocircuit form constraints of Formulation (CO) in page 18. Each triangle in the underlying triangulation then belongs to one element in the quadrilateral mesh, which is either a quadrilateral or a triangle (Figure 5.13).

We assign variable y_i to mesh element q_i, distinguishing two quadrilaterals according to the diagonal to be chosen for the underlying triangulation. Let S_k denote the set of mesh elements to which t_k can belong. We can define set partitioning constraints between mesh elements and underlying triangulation elements as follows:

Formulation (UT):

maximize $\boldsymbol{cy}$

s.t.

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

Extending cocircuit form constraints

We can also consider constraints among mesh elements similar to the cocircuit form constraints for triangulations (Figure 5.14). Let P_k ($/N_k$) denote the set of mesh elements that have edge f_k as a face and other faces on the positive (/negative) side of f_k, respectively.

Formulation (COQ):

maximize $\boldsymbol{cy}$

s.t.

$$
\begin{aligned}
y_i &\in \{0,1\} \\
\sum_{i \in P_k} y_i - \sum_{i \in N_k} y_i &= \begin{cases} 1 & f_k \text{ is on the boundary and oriented inside} \\ -1 & f_k \text{ is on the boundary and oriented outside} \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
$$

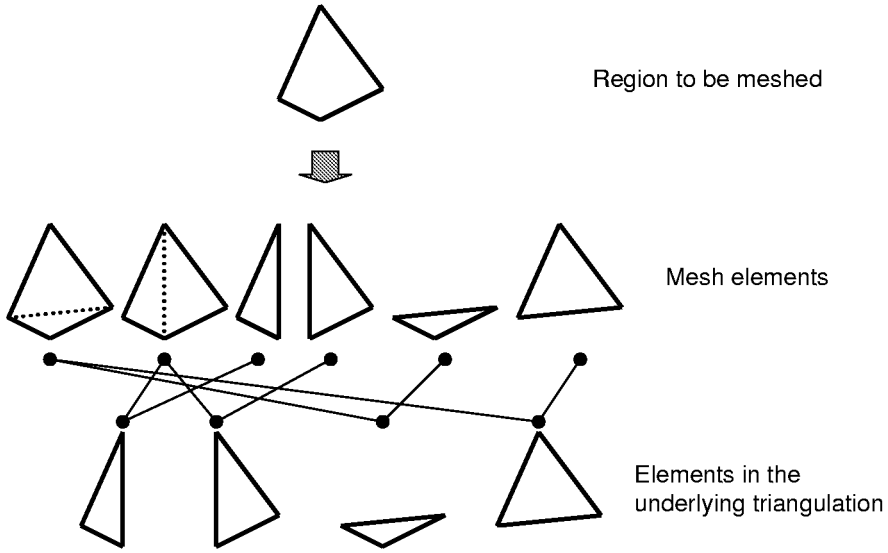

Figure 5.13: Elements in the quadrilateral mesh and the underlying triangulation

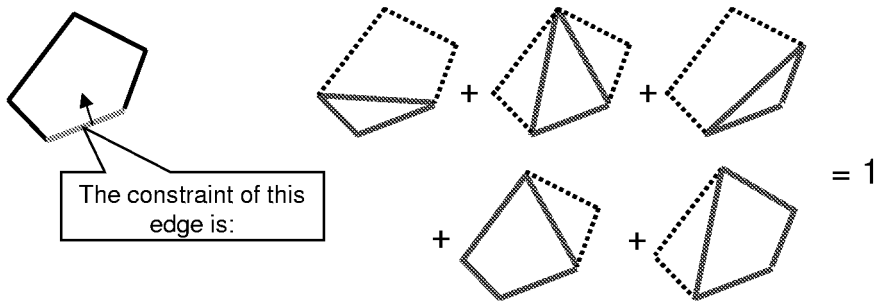

Figure 5.14: An extended cocircuit form constraint

Equivalence between the two formulations

Here we compare the polytopes defined by the linear programming relaxations of Formulation (UT) and (COQ). Let P_{UT} denote the polytope defined by the feasible region of $\boldsymbol{y}$ in the linear programming relaxation of Formulation (UT). That is to say, we do not consider the variables corresponding to the underlying triangles here. Let P_{COQ} denote the polytope corresponding to y in Formulation (COQ).

Proposition 5.3 $P_{UT} = P_{COQ}$

Proof: First we show that $P_{COQ} \subseteq P_{UT}$. Consider a point y_{COQ} in P_{COQ}, namely, a feasible solution to the linear programming relaxation of Formulation (COQ), which can be fractional. We split all the quadrilaterals corresponding to positive values in y_{COQ} into triangles of the same values, and generate a fractional incidence vector of triangles, which will be denoted as x_{COQ}. The split above introduces triangles with the same weight on both sides of the diagonal, and does not violate the cocircuit form constraints, thus x_{COQ} satisfies the constraints in Formulation (UT). By construction x_{COQ} and y_{COQ} satisfy the set partitioning constraints in Formulation (UT). Hence $y_{COQ} \in P_{UT}$.

We then show $P_{UT} \subseteq P_{COQ}$ by deriving the constraints in Formulation (COQ) from the constraints in Formulation (UT). Consider an edge on the boundary f_b. All the triangles that have f_b as a face belong to different mesh elements. Thus,

$$\sum_{t_i = f_b \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_b}} x_i = \sum_{t_i = f_b \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_b}} \sum_{j \in S_i} y_j = \sum_{i \in P_b} y_i$$

$$\sum_{t_i = f_b \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_b}} x_i = \sum_{t_i = f_b \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_b}} \sum_{j \in S_i} y_j = \sum_{i \in N_b} y_i$$

Next we consider an internal edge f_n. Let D_n be the set of quadrilateral mesh elements that have f_n as a diagonal.

$$\sum_{t_i = f_n \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_n}} x_i = \sum_{t_i = f_n \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^+_{f_n}} \sum_{j \in S_i} y_j = \sum_{i \in P_n} y_i + \sum_{i \in D_n} y_i$$

$$\sum_{t_i = f_n \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_n}} x_i = \sum_{t_i = f_n \cup \{a\}, a \in \mathcal{A} \cap \mathcal{H}^-_{f_n}} \sum_{j \in S_i} y_j = \sum_{i \in N_n} y_i + \sum_{i \in D_n} y_i$$

Thus, a point y_{UT} in P_{UT} satisfies all the constraints in Formulation (COQ). □

We have observed that the two extended formulations give the same polytope, and we can say that the latter formulation with extended cocircuit form constraints are better in the sense that both the number of constraints and the number of variables are smaller.

Further, we have so far distinguished two quadrilaterals with the same vertices but with different diagonals to be used for the conversion into triangulation. However, with the formulation using the extended cocircuit form constraints, we can ignore the difference and assign only one variable to the two quadrilaterals.

Table 5.3: Input data

Data set	No. of points	No. of triangles	Minimum no. of elements (*)
ITOT1	294	505	254
ITOT2	401	728	364
ITOT3	180	283	143
U50	50	89	49
U100	100	188	103
U150	150	288	153

(*): if based on Delaunay triangulation

Coping with non-convex boundaries

So far we have assumed that the region to be meshed is the convex hull of a point configuration, namely, a simple and convex polygon. The practical interest, however, is in meshing non-convex polygons with holes. Fortunately, in two dimensions, we can always triangulate a non-convex polygon, and the formulations have a feasible solution. Thus, we only need to preprocess and remove the meshing elements that intersect the boundaries.

5.5.4 Computational experiments

We present the results of some computational experiments that we conducted to compare the methods described in the previous sections. For weighted matching problems, we used the implementation of Gabow's $O(|V|^3)$ time algorithm [31] at the Dimacs Challenge 1 site [24].

Input data

We used three sets of data supplied by Itoh (ITOT1, ITOT2, ITOT3), and three randomly generated point sets uniformly distributed in a unit square (U50, U100, U150) (Table 5.3). Since the vector field is difficult to replicate, we fixed the corresponding coefficient to zero, and used $a = 0$, $b = 1.0$, and $c = 1.0$ for (5.2).

Results

From Table 5.4, we can see that the matching-based method gave better solutions than Itoh et al.'s method[3] , and that our IP-based method gave the best solutions. The IP-based method gave a smaller number of mesh elements than the smallest possible number of elements based on Delaunay triangulation, which is given in the rightmost column of Table 5.3.

Figure 5.15 shows the heuristic solution by Itoh et al. and our optimal solution. We could not obtain the optimal solutions for ITOT1 and ITOT2 (Table 5.5). Instances of

[3]The settings are not strictly the same as Itoh et al.'s, and the comparison in Table 5.4 is just for reference.

Table 5.4: Comparison of the results

Data set	Itoh		Matching		IP	
	No. of elements	Obj.	No. of elements	Obj.	No. of elements	Obj.
ITOT1	264	825.0	260	837.1	–	–
ITOT2	376	1221.4	372	1235.0	–	–
ITOT3	168	322.4	151	375.5	144	392.5
U50	–	–	49	69.6	47	75.7
U100	–	–	104	152.3	99	172.1
U150	–	–	153	249.7	148	279.5

Table 5.5: Size of the problems and CPU time

Data set	No. of rows	No. of columns	No. of B&B nodes	CPU time (sec.)
ITOT1	43071	424334	–	–
ITOT2	–	–	–	–
ITOT3	16110	143387	4	94876.82
U50	1225	9658	0	183.6
U100	4950	46968	0	6188.68
U150	11175	111730	3	51818.71

two to three hundred points seem to be the largest possible instances with the current computational resources.

We obtained a fractional solution with ITOT3 (Figure 5.16). It contains two odd-cycles of size 5 and 11, respectively. The former corresponds to a pentagon with a point inside, subdivided into five 0.5-weighted quadrilaterals using the center point.

Figure 5.17 shows the optimal solution for U100, where we can observe many non-Delaunay edges drawn in bold lines.

5.5.5 Remarks

From a practical point of view, the IP-based method is too slow, soluble instances are too small, and the optimal solutions are not always good as a *mesh*.[4] Still, it gives us some insight into the quality of a mesh; namely, it tells us how good the solutions are that we can obtain from the given input, and how suitable the current objective function is for our applications.

5.6 Toward Hexahedral Mesh Generation

In this section, we investigate a way to contribute to hexahedral mesh generation by using our IP-based approach.

We considered optimizing quadrilateral mesh generation in two dimensions in Section 5.5. As we mentioned there, there are also large needs for techniques for hex-

[4] Mesh generation is still an art and cannot be evaluated appropriately with an objective function.

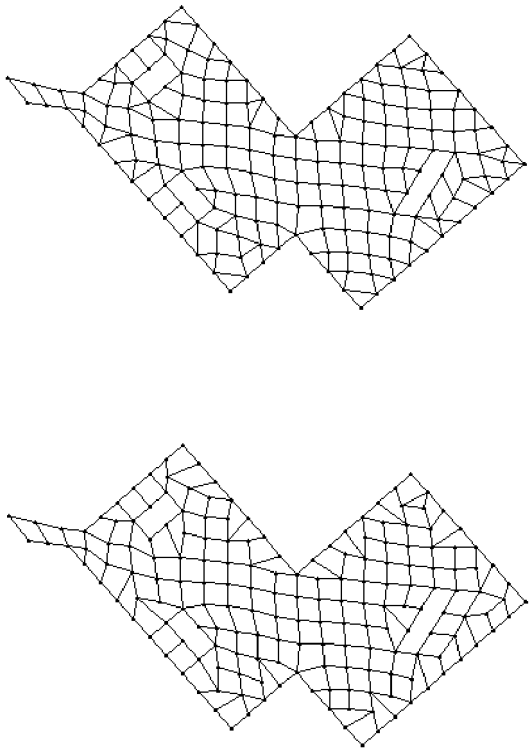

Figure 5.15: ITOT3: heuristic and optimal solutions

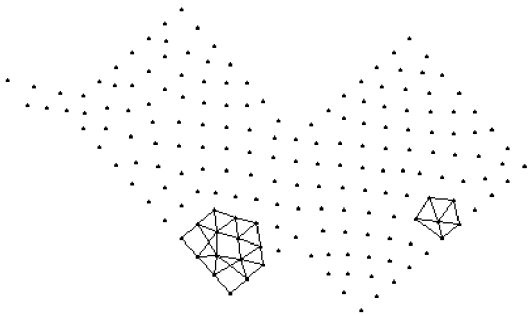

Figure 5.16: ITOT3: a fractional solution

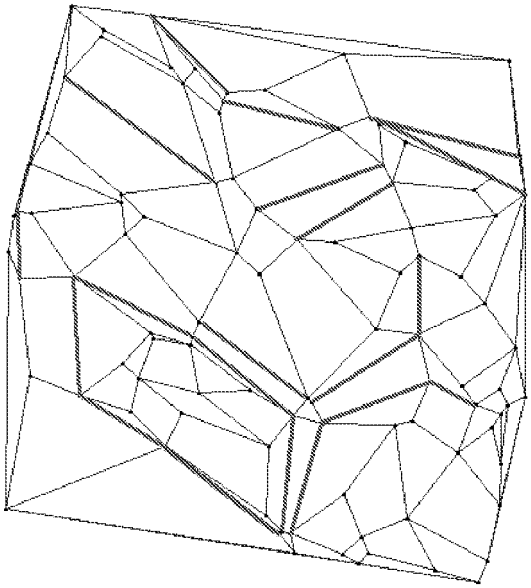

Figure 5.17: Non-Delaunay edges in the optimal mesh

ahedral mesh generation in three dimensions. All the commercial applications for hexahedral mesh generation require some extent of human-computer interactions, and it is still one of the major goals in mesh generation research to generate a high-quality hexahedral mesh automatically.

For hexahedral meshes, the point configuration must be in a special position such that four points are on the same plane for each face of mesh elements. Obviously, to generate such a point configuration is itself a very difficult problem, and indirect methods that handle point generation and mesh generation separately do not seem appropriate for hexahedral meshes.

Thus most of the existing hexahedral mesh generators are designed for structured mesh, namely, to apply predefined grid structures, called templates, to the region to be meshed [50]. As the form of the region gets complicated, users have to specify how to apply the templates. Namely, it is required to subdivide the region into elements of simple forms, and users have to define it manually. We will call this procedure as *specifying the topology.*

Automatically specifying the topology will quite reduce the human workload for mesh generation. If the points used to subdivide the region is provided by somehow, manually or automatically, this procedure can be regarded as a combinatorial optimization problem: *given a three dimensional region and a set of points within it, subdivide the region into simple topological elements such as tetrahedra, prisms, pyramids, and hexahedrals using the given points as vertices.* The largest difference from our stand points in this thesis is that, it just focuses on the topology. Even Schönhardt's poly-

tope (Figure 1.3) is a prism if we do not consider the three diagonals, and also can be triangulated into tetrahedra.

5.6.1 A procedure for topological subdivision into simple elements

It depends on the applications how flexible the elements can be. In order to maintain the consistency of the situation, we need to assume that the transitivity on coplanarity holds:

Assumption 5.1 *Consider five points p_i $(0 \leq i \leq 4)$. If four points p_i $(0 \leq i \leq 3)$, and p_i $(1 \leq i \leq 4)$ are topologically on the same planes respectively, the five points p_i $(0 \leq i \leq 4)$ are topologically on a plane.*

Then we can consider a subdivision procedure as follows:

1. Enumerate all the possible triangles and quadrilaterals that can be facets of elements, by selecting three or four coplanar points out of the given point set.
2. Enumerate all the possible elements such as tetrahedra, prisms, pyramids, and hexahedrals that can be mesh elements whose facets are a subset of the triangles or quadrilaterals enumerated in Step 1. Non-empty elements and elements intersecting the boundary should be eliminated here.
3. Define an objective function to evaluate the quality of the elements enumerated in Step 2, and also assign 0/1 variables to all of them.
4. Consider cocircuit-form-like constraints. Namely, for each facet enumerated in Step 1, the sum of elements on one side is equal to the sum of the elements on the other side.
5. Solve the IP problem to obtain a subdivision.

Unfortunately, the procedure above is just a conceptual sketch. The following open problems have to be solved to realize it.

- *How to locate points to obtain a good subdivision?*
  The procedure highly depends on the point configuration. More points should be located in complicated regions, and few points are enough for simple regions. One approach would be to define a function that quantify the complexity of the region, and locate points according to the value or the gradient of the function. Human interaction will also be a practical solution.
- *How can we define the coplanarity?*
  Points on a curved surface may be better considered to be coplanar. It means that just setting an error bound may not work in practical cases.

- *Can we loosen the transitivity in Assumption 5.1?*
  The transitivity assumption is too strict, but the consistency that is necessary for optimization can collapse without it. This is also related to the coplanarity above. How to balance the coplanarity and the transitivity would be a key issue.

- *How good are the cocircuit-form-like constraints in Step 4?*
  This is a common problem with the case of quadrilateral mesh generation. There is no theoretical result, and it is still open whether an integer solution surely correspond to a subdivision. Conversely, if we can find a pathological case, we will be able to obtain some insights.

- *Is there a good objective function suitable to evaluate topological elements?*
  This issue is highly dependent on applications. It also depends on the available templates. In any case, we require some measure in order to apply integer programming, and the measure should be designed carefully.

- *How can we handle the cases where the region is non-convex and cannot be subdivided with the enumerated elements?*
  This is related to Theorem 1.1 and 1.2. It is open whether they also hold with topological cases. Even Schönhardt's polytope is a prism if we do not consider the three diagonals. This means that the situation depends on the definition of coplanarity.

Chapter 6

Conclusions and Remarks

In this thesis, we investigated IP-based optimization of triangulations, covering multiple aspects ranging from formulations to applications. Throughout this thesis, we focused not only on d-simplices but also lower dimensional simplices as seen in $(d-2)$-face cuts and the generalized stable set formulation. We also kept examining our ideas through computational experiments, which sometimes gave further insights. In particular, we could actually optimize middle-sized problems both in theory and applications, which can not be solved by enumeration or other alternatives so far, under various measures.

In Chapter 2, we examined the existing studies by classifying into two groups, those could be reduced to the stable set problem, and those to the set partitioning problem. Further for the former, we gave a more efficient formulation as an instance of the generalized stable set problem by removing the redundancy based on the observation of lower dimensional simplices. After theoretical investigations, we concluded that the formulation using cocircuit form constraints was the most efficient and suitable for the triangulation of a point configuration. On the other hand, the (generalized) stable set formulation is suitable for several extensions such as dissections or non-convex boundaries. We observed the effectiveness of explicitly handling lower dimensional simplices. This was further utilized in Chapter 4 in relation to cutting planes and the branch and bound procedure.

In Chapter 3, we showed how large instances we could solve with the formulation using cocircuit form constraints through computational experiments, with the observations on the degeneracy of the problem in both primal and dual. We also gave several interesting examples of optimal triangulations. We then introduced practical difficulties to be coped with in Chapter 4.

In Chapter 4, we investigated column generation methods to solve large instances with relation to geometry. Then we introduced a binary search algorithm to successfully solve the bottleneck-type optimization problems, which could not be solved in a naive way. We introduced two cutting planes that were based on geometric aspects of triangulation, and were observed through computational experiments to be practically

very useful when the objective function was unweighted and the solution of the LP relaxation was highly fractional. We also proposed a method to improve the branch and bound procedure by focusing on lower dimensional simplices.

In Chapter 5, we gave not only the minimum weight triangulation but also the minimum squared weight triangulation, which could not be obtained with algorithms in computational geometry. We also showed that, with our IP-based approach, we could easily obtain data dependent triangulations, which had been long studied by using local search. We addressed the symmetry and degeneracy of the point configuration by focusing on regular and quasi-regular polytopes. We also gave the minimum and maximum cardinality of triangulations and dissections of those polytopes. We finally coped with industrial applications; quadrilateral and hexahedral mesh generation. In particular, we gave an extension of our formulation for quadrilateral mesh generation with successful computational results.

6.1 Future Work

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.

We sometimes obtained fractional solutions after applying $(d-2)$-face cuts, and further investigation into the universal polytope, the integer hull of the incidence vectors of triangulations, are important. $(d-2)$-face cuts are not proven to be facets of the universal polytope, and there are a lot to be done.

One of the most important objectives of this study is to contribute to practical applications, that is, to give significant insights into mesh generation, geometric design, and computer graphics. Coping with non-convex boundaries and handling hexahedral mesh elements are also difficult but significant issues.

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] CRPC News Archive. Record traveling salesman solution – new heuristic approach to classic linear programming problem yields big dividend, 1998. http://www.crpc.rice.edu/CRPC/newsArchive/techweb_6_29_98.html.

[4] I. Bárány and Z. Füredi. Empty simplices in Euclidian spaces. *Canad. Math. Bull.*, 30:436–445, 1987.

[5] R. E. Barnhill. Surfaces in computer aided geometric design: A survey with new results. *Computer Aided Geometric Design*, 2:1–17, 1985.

[6] R. E. Barnhill and F. F. Little. Three- and four dimensional surfaces. *Rocky Mountain J. Math.*, 14:77–102, 1984.

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

[8] 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.

[9] 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.

[10] L. J. Billera and A. Björner. Face numbers of polytopes and complexes. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 15. CRC Press, 1997.

[11] J. R. Brauer. *What Every Engineer Should Know about Finite Element Analysis*. Marcel Dekker Inc., 2nd edition, 1993.

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

[13] S. W. Cheng, M. J. Golin, and J. C. F. Tsang. Expected case analysis of β-skeletons with applications to the construction of minimum-weight triangulation. In *Proc. 7th Canadian Conference of Computational Geometry*, pages 279–284, 1995.

[14] S. W. Cheng, N. Katoh, and M. Sugai. A study of the *lmt*-skeleton. In *Proc. 7th International Symposium on Algorithms and Computation*, volume 1178 of *Lecture Notes in Computer Science*, pages 256–265. Springer Verlag, 1996.

[15] S. W. Cheng and Y. F. Xu. Approaching the largest β-skeleton within a minimum weight triangulation. In *Proc. 12th Annual ACM Symposium on Computational Geometry*, pages 196–203, 1996.

[16] V. Chvatal, 1998. Private communication.

[17] H. S. M. Coxeter. *Regular Polytopes*. Dover Publications, INC., 3rd edition, 1973.

[18] J. A. De Loera. Computing minimal and maximal triangulations of convex polytopes. *working paper*, 1998.

[19] 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.

[20] J. A. De Loera, F. Santos, and F. Takeuchi. Maximal dissections of convex polytopes. *in preparation*, 1999.

[21] J. A. De Loera and F. Takeuchi, 1999. Private communication.

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

[23] 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.

[24] DIMACS Challenge 1. ftp://ftp.rutgers.edu/pub/netflow.

[25] J. Dompierre, P. Labbe, F. Guibault, and R. Camerero. Proposal of benchmarks for 3d unstructured tetrahedral mesh optimization. In *Proc. of 7th International Meshing Roundtable*, pages 459–478, 1998.

[26] N. Dyn, D. Levin, and S. Rippa. Data dependent triangulations for piecewise linear interpolation. *IMA Journal of Numerical Analysis*, 10:137–154, 1990.

[27] H. Edelsbrunner, F. P. Preparata, and D. B. West. Tetrahedrizing point sets in three dimensions. *Journal of Symbolic Computation*, 10(3–4):335–347, September–October 1990.

[28] P. Fleischmann and S. Selberherr. Three-dimensional Delaunay mesh generation using a modified advancing front approach. In *Proc. of 6th International Meshing Roundtable*, pages 267–278, 1997.

[29] S. Fortune, 1999. Private communication.

[30] L. A. Freitag and P. M. Knupp. Tetrahedral element shape optimization via the jacobian determinant and condition number. In *Proc. of 8th International Meshing Roundtable*, pages 247–258, 1999.

[31] H. N. Gabow. *Implementation of Algorithms for Maximum Matching on Nonbipartite Graphs*. PhD thesis, Dept. of Electrical Eng., Stanford Univ., 1973.

[32] Z. Galil, S. Micali, and H. Gabow. An $O(EV \log V)$ algorithm for finding a maximal weighted matching in general graphs. *SIAM Journal on Computing*, 15(1):120–130, February 1986.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[34] I. M. Gelfand, M. M. Kapranov, and A. V. Zelevinsky. *Discriminants, Resultants, and Multidimensional Determinants*. Birkhäuser, Boston, 1994.

[35] P.-L. George and H. Borouchaki. *Delaunay Triangulation and Meshing*. Hermes, 1998.

[36] P. Gritzmann and V. Klee. Computational convexity. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 26. CRC Press, 1997.

[37] M. Halpern. Industrial requirements and practices in finite element meshing: A survey of trends. In *Proc. of 6th International Meshing Roundtable*, pages 399–411, 1997.

[38] M. Henk, J. Richter-Gebert, and G. M. Ziegler. Basic properties of convex polytopes. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 13. CRC Press, 1997.

[39] K. Ho-Le. Finite element mesh generation method: a review and classification. *Computer Aided Design*, 20(1):27–38, 1988.

[40] T. Itoh, K. Shimada, K. Inoue, A. Yamada, and Tomotake Furuhata. Automated conversion of 2d triangular mesh into quadrilateral mesh with directionality control. In *Proc. of 7th International Meshing Roundtable*, pages 77–86, 1998.

[41] kaleido. http://www.math.technion.ac.il/kaleido/.

[42] P. M. Knupp. Matrix norms & the condition number. In *Proc. of 8th International Meshing Roundtable*, pages 13–22, 1999.

[43] K. Koyamada, S. Uno, A. Doi, and T. Miyazawa. Fast volume rendering by polygonal approximation. *Journal of Information Processing*, 15(4):535–544, 1992.

[44] 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.

[45] LEDA. http://www.mpi-sb.mpg.de/LEDA.

[46] A. Lingas. The greedy and Delaunay triangulations are not bad in the average case. *Information Processing Letters*, 22:25–31, 1986.

[47] 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.

[48] K. Mehlhorn and S. Näher. *LEDA: A Platform for Combinatorial and Geometric Computing*. Cambridge University Press, 1999.

[49] 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.

[50] K. Nakahashi and K. Fujii. *Grid Generation and Computer Graphics*, volume 6 of *Computational Fruid Dynamics Series*. University of Tokyo Press, 1995. (in Japanese).

[51] G. L. Nemhauser and G. Sigismondi. A strong cutting plane/branch-and-bound algorithm for node packing. *Journal of Operations Research Society*, 43(5):443–457, 1992.

[52] G. L. Nemhauser and L. E. Trotter. Properties of vertex packing and independent system polyhedra. *Mathematical programming*, 6:48–61, 1974.

[53] G. L. Nemhauser and L. E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical programming*, 8:232–248, 1975.

[54] OSL. http://www6.software.ibm.com/es/oslv2/features/welcome.htm.

[55] S. Owen. A survey of unstructured mesh generation technology. In *Proc. of 7th International Meshing Roundtable*, pages 239–267, 1998.

[56] PORTA. http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/PORTA/readme.html.

[57] PUNTOS. http://www.geom.umn.edu/about/people/home/deloera.html.

[58] V. T. Rajan. Optimality of the Delaunay triangulation in R^d. In *Proc. 7th ACM Symp. Computational Geometry*, pages 357–363, 1991.

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

[60] J. Rambau, 1999. Private communication.

[61] J. Richiter-Gebert. The universality theorems for oriented matroids and polytopes. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 233 of *Contemporary Mathematics*, pages 269–292. American Mathematical Society, 1999.

[62] S. Rippa. *Piecewise linear interpolation and approximation schemes over data dependent triangulations*. PhD thesis, Tel Aviv, 1989.

[63] J. Ruppert and R. Seidel. On the difficulty of triangulating three-dimensional nonconvex polyhedra. *Discrete & Computational Geometry*, 7:227–253, 1992.

[64] F. Santos. A point configuration whose space of triangulations is disconnected, 1999. Preprint. Available at http://matsun1.matesco.unican.es/ santos/Articulos/.

[65] L. L. Schumaker. Computing optimal triangulations using simulated annealing. *Computer Aided Geometric Design*, 10:329–345, 1993.

[66] J. Sekiguchi. *Mathematical Study on Polyhedra and Graphics*, volume 13 of *Mathemacical and Information Science*. Makino Books, 1996. (in Japanese).

[67] K. Shimada. Physically-based automatic mesh generation. *Simulation*, 12(1):11–20, 1993. (in Japanese).

[68] K. Shimada, JH. Liao, and T.Itoh. Quadrilateral meshing with directionality control through the packing of square cells. In *Proc. of 7th International Meshing Roundtable*, pages 61–76, 1998.

[69] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, 1990.

[70] P. Shor. Stretchability of pseudoline arrangements is NP-hard. In P. Gritzman and B. Strumfels, editors, *Geometry and Disrete Mathematics – The Victor Klee Festschrift*, volume 4 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 531–554. American Mathematical Society, 1991.

[71] 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)*, volume 1533 of *Lecture Notes in Computer Science*, pages 377–386. Springer Verlag, 1998.

[72] Takeuchi and Imai. Enumerating triangulations for products of two simplices and for arbitrary configurations of points. In *COCOON: Annual International Conference on Computing and Combinatorics*, pages 470–481, 1997.

[73] TOPCOM. http://www.zib.de/rambau/TOPCOM.html.

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

[75] C. A. Wang, F. Chin, and Y. F. Xu. A new subgraph of minimum weight triangulation. In *Proc. 7th International Symposium on Algorithms and Computation*, volume 1178 of *Lecture Notes in Computer Science*, pages 266–274. Springer Verlag, 1996.

[76] L. A. Wolsey. *Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1998.

[77] J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, and J. Wu. A new approach to the development of automatic quadrilateral mesh generation. *International Journal for Numerical Methods in Engineering*, 32:849–866, 1991.

[78] G. M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer-Verlag, revised first edition edition, 1995.