A Genetic Algorithm for Minimum Tetrahedralization of a Convex Polyhedron*

Kiat-Choong Chen† Ian Hsieh† Cao An Wang†

Abstract

A minimum tetrahedralization of a convex polyhedron is a partition of the convex polyhedron with minimum number of tetrahedra. The problem of finding the minimum tetrahedralization of a convex polyhedron is known to be NP-Hard. In this paper, a *genetic algorithm* is presented to find an approximate solution to this problem. Our result always shows improvements to those produced by commonly used *peeling* and *pulling* methods.

1 Introduction

Tetrahedralization is an importatnt research topic in computational geometry. It has many applications in areas such as finite element method, computer graphics, robotics, CAD/CAM and mesh generation. While a simple polyhedron may not be always tetrahedralizable [10], it can always be done for convex polyhedra. Moreover, finding a minimum tetrahedralization even in convex polyhedra has been proved to be NP-C [2]. There are two widely used methods for tetrahedralizing a convex polyhedron, namely, peeling and pulling. The former is the reverse process of repeatedly removing the (tetrahedralized) cap of a vertex in the polyhedron, and the latter (or called *starring*) is to partition the polyhedron by connecting a vertex to all its non-adjacent vertices by edges. The peeling method produces $O(n \log n)$ tetrahedra for a convex polyhedron with n vertices [3]. It is $O(n^2)$ for ordered peeling (ordered peeling will be described later in the paper). The pulling method outputs $2n - deg(v) - 4$ tetrahedra, where v is the tip vertex of the pulling and $deg(v)$ is the degree of v in the surface graph of the polyhedron[4]. In many cases, such as *stacked* polyhedron, both methods may produce very bad results with respect to the optimum one.

In this paper, we propose a genetic algorithm for the problem. We briefly describe how to represent population, selection, crossover, and mutation for tetrahedralization. The algorithm is implemented in JAVA. The results of our algorithm are compared with the results generated from peeling and starring methods. It is shown that the genetic approach can produce a lesser number of tetrahedra than the above mentioned two methods in almost all the cases. Moreover, the genetic approach can obtain the optimal result for certain convex polyhedrons for which the minimum tetrahedralization is known (e.g. 'stacked' polyhedron and 'bi-umbrella' polyhedron).

This paper is organized as follows: section 2 gives a brief description of genetic algorithms, while section 3 shows the results produced by our program, and section 4 concludes our work.

2 Genetic Algorithm

The Genetic Algorithm (GA for short) is a heuristic search model and it performs a multi-directional search, while other methods (e.g. hill climbing) only process a single direction in the search space. A GA goes through a process of evolution: the best fits will survive to the next generation and the bad fits will die off. A fitness function is used to distinguish and measure how good each solution.

The set of possible solutions (called chromosomes) are generated. The set is called a population. Once the population has been initialized, a process that mimics evolution in nature begins, and runs for a designated number of times. This evolutionary process crosses the genetic information of two chromosomes to form two new children or mutates a chromosome's existing genetic information to form a new chromosome. Each iteration, also called a generation, creates a new population of chromosomes based on the previous iteration of chromosomes. The evolution process consists of the following steps:

(1) Evaluating the fitness function f (x) of each chromosome in the population; (2) Selection - selecting two parent chromosomes for crossover, based on their f (x) value; (3) Crossover - exchange the genetic information of the two parents to form two new children and copy them into the new population, based on a percentage. If crossover was not performed then duplicate copies of the parents are copied into the new population; and (4) Mutation - change or alter the genetic information of a chromosome based on percentage.

*This work is based on the undergraduate Honours project by the first two authors. The work of the third author is partially supported by NSERC grant OPG0041629.

†Department of Computer Science, Memorial University of Newfoundland, St. John's, Newfoundland, Canada A1B 3X5 (email: kiat@cs.mun.ca, ihsieh@cs.mun.ca, wang@garfield.cs.mun.ca)

In our problem, the set of chromosomes is the set of possible tetrahedralizations of the convex polyhedron from the point set.

• **Representations**

A tetrahedralization of a convex polyhedron uniquely determines a set of triangles. In our case, we represent each tetrahedralization by a list of triangles as well as a list of (internal) edges. For each triangle, we use its three vertices to represent it. For each vertex, we record its x, y, and z coordinates. To represent the relationship between the edges and the triangles, a 2-dimensional matrix stores the number of triangles that share a certain edge. That is, if the element at row i and collumn j in the matrix is a 4, then it corresponds edge (i, j) is shared by exactly 4 traingles. For example, the left-hand side of Figure 1 is the tetrahedralization of convex polyhedron with 5 vertices, and the middle of the figure is the matrix representing the relationship of the edges and the numbers of their sharing triangles.

• **Initialize Population**

First, all the points forming a convex polyhedron are generated using a JAVA package from [7]. Since these points are randomly generated on a sphere, all these points are the vertices of the convex polyhedron. After the vertices of a convex polyhedron had been created, we use Peeling and Pulling methods to create tetrahedralizations. In peeling method, we have two variations: random and ordering. With random method, a vertex is randomly chosen from the remaining vertices and connect it to the tetrahedralization of a subset of vertices previously chosen to form a new tetrahedralization with one more vertices. With ordering method, the vertices are ordered by their x (or y or z) coordinates and then chosen in the ordered sequence.

• **Evaluation of Fitness**

Evaluating the fitness function of each chromosome in the population is based on Euler's formula $t = e + n - 3$, where t is the number of tetrahedra, e is the number of internal edges and n is the number of vertices of the convex polyhedron. Since the number of tetrahedra is proportional to the number of internal edges, the chromosome's fitness is determined by the number of internal edges it contains. The fitter chromosome will have a lesser number of internal edges. The fitness function is defined as $f(C) = SIZE(e_iList)$, where C is the chromosome and $SIZE$ is a function that returns the number of internal edges in e_iList of C.

• **Selection Process**

The *Roulette Wheel Selection* method is used for the selection of chromosomes that will become the candidates (parents) for the crossover operator [9].

In addition, another parameter, called *Elitism*, is used for selection. Elitism is a selection method where the most-fit chromosomes in the population are automatically copied into the next generation. That is, if the elitism parameter were set to K, then the top K chromosomes in the population are copied to the next generation. This is to ensure that the best chromosome generated so far is passed down to the next generation and to guarantees that our population will not degrade over the evolution process.

• **Crossover**

Crossover is to exchange genetic information between two parent chromosomes (say C1 and C2 which are selected from the selection process) to form two new children. The crossover operator is associated with a crossover rate. For example, for a particular crossover process, a random number between 0 and 1 is generated. If the number is lower than certain value, says CR (crossover rate), then the crossover operator is performed. With regard to our representation of a tetrahedralization, a simple swapping of triangles and internal edges between two tetrahedralizations does not guarantee a legal tetrahedralization as some internal triangles would overlap each other and hence needs correction after crossover. To avoid this problem, a method similar to the polygon crossover as described in [5] is used. In our case, a common subboundary (SUB) is searched for both C1 and C2. A SUB is the set of vertex that forms a boundary which is common in C1 and C2. The algorithm of finding SUB is outlined briefly as below:-

Randomly select an internal edge, iEdge, which is in one chromosome, say C1, but not in the other, say C2. If iEdge intersects any triangles of C2, then (1) record all the vertices of the intersecting triangles to SUB. (2) record all the edges that exist between the vertices of both, the two end points of iEdge and the vertices of the intersecting triangles, and the vertices in SUB. (3) record all the internal edges from both chromosomes such that one of the end points of the internal edge is in SUB, but not the other, and (4) record all the edges that exist between the vertices of the intersecting triangles and the two end points of iEdge. All these edges are recorded in a Stack, say STK. Next, do the intersection checking process using all the recorded edges in STK as iEdge against the triangles of C1. This intersection checking process is repeated until no more intersections are detected in both C1 and C2.

Because SUB is contained and common in both parent chromosomes, both of those regions can be legally swapped from the two parents, producing two new legal tetrahedralizations as children. In the worst case, there is no SUB between C1 and C2 (SUB will be the given convex polyhedron) and hence C1 and C2 will be treated as children of next generation with no crossover.

• **Mutation**

Mutation is to alter a small portion of a chromosome and hence introduces variability into the population of the next generation. In our case, the mutation operator is based on "local transformation" or flip, namely "3-2"

No. of Points	RP	GA_{RP} with	GA_{RP} with	OP	GA_{OP} with	Starring /	$GA_{Starring}$ with
		Random Mutation	Ranking Mutation		Random Mutation	Pulling	Random Mutation
10	4.1	2.5	2.5	7.2	2.5	2.8	2.5
15	12.1	6.8	6.7	18.3	7	7.1	6.8
20	21.7	11.2	10.5	34.1	11.6	11.5	11
30	44	22.4	20.7	66.7	23	20.7	20.4

Figure 1: RP = Random Peeling Tetrahedralization GA_{RP} = Genetic Algorithm that used Random Peeling to initialize its initial population, OP = Ordered Peeling Tetrahedralization, GA_{OP} = Genetic Algorithm that used Ordered Peeling to initialize its initial population, Starring / Pulling = Pulling Tetrahedralization, $GA_{Starring}$ = Genetic Algorithm that used Pulling to initialize its initial population.

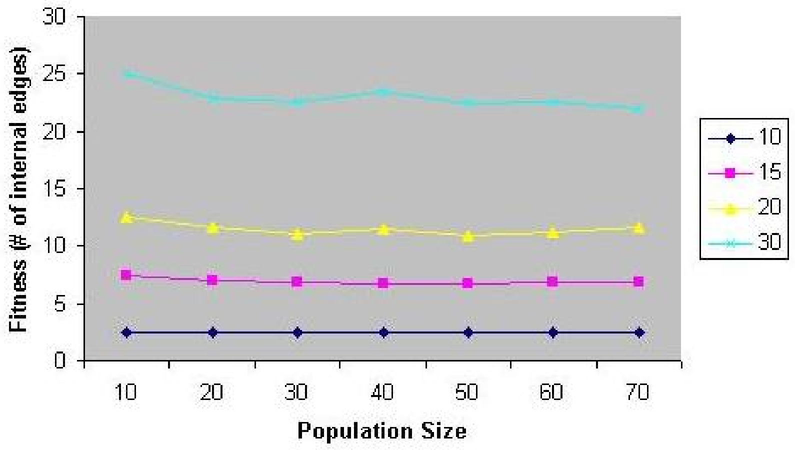


Figure 2: Above figure shows the performance of GA for different number of points (n) and population sizes. For n = 10 and 15, population size between 40 - 50 gives lesser number of internal edges compared other population sizes, whereas when n = 20 and 30, a 'good' population size is in about the range 50 - 70.

flip and "2-3" flip as described in [1]. A "3-2" flip is to remove an internal edge whereas a "2-3" flip is to add an internal edge. Therefore, they can be used to alter the chromosome so that the number of tetrahedra can possibly be increased or decreased. However, not all internal edges are flippable. A "3-2" flippable edge must satisfy two conditions: (1) it is shared by exactly three tetrahedra and (2) it is not on the CH. For "2-3" flip, the edge to be added must also satisfy two coditions: (1) it intersects exactly one triangle and (2) the vertices of the intersecting triangle must have edges connecting to both end points of the new edge. In the left-hand side of figure 3, the internal edge (2,3) can be removed with the "3-2" flip operator and can be added with the "2-3" flip operator. As with the crossover, the mutation operator is also associated with a mutation rate (MR) to determine whether or not the mutation operator is to be applied to the chromosome. To determine which mutation operator is to be applied to the chromosome, a random number between 0 and 1 is generated. If the number is lower than a certain value, say FR (flip rate for 3-2 flip), then "3-2" flip is chosen, otherwise, the "2-3" flip would be chosen. There are two ways to select the flippable edge, by random selection (Random Mutation) or ranking selection (Ranking Mutation). Random selection is to select randomly the flippable edge from a set of all possible flippable edges, whereas ranking selection is to rank the set of flippable edges according to the vertex degree before the selection. For "2-3" flip, the ranking is in descending order, else ascending order. By using the ranking mutation, the chromosome is possibly mutated towards a similar structure as the chromosomes generated by pulling method that gives linear number of tetrahedra in the hopes that this will direct GA to produce a better minimum than the pulling method.

3 Results

The GA contains three parts; tetrahedralization, visualization and evolution. The first two parts are implemented by modifying a CH applet package from [7]. GA is tested on point sets of sizes $10, 15, 20$ and 30 with 10 sets of each size. The point sets were randomly generated on the sphere. The size of the generations was set to $20(n)$ where n is the number of points or the iteration will terminate when there is no more internal edges in the tetrahedralization. Population size was set to 40 for $n = 10$ and 15 and 50 for $n = 20$ and 30 (see figure

2). The crossover rate was set to 0.05, elitism was set to 1 and mutation rate was 0.7. Flip rate was set to 0.5 for "3-2" flip and 0.5 for "2-3" flip. The results of GA are compared with three different methods of tetrahedralization, namely random peeling, ordered peeling and pulling.

Results with Randomly Generated Point Sets On Sphere

The results in the table were obtained by taking the average number of tetrahedra on 10 trials of each number of points.

Results with Known Optimum Solution (point sets are generated manually)

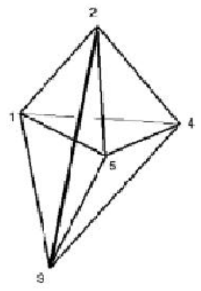
Below are the results obtained by the GA on point sets for which the optimum solution is known. (See figure 4, the left-hand side shows the initial tetrahedralization with many internal edges and the right-hand side shows the result obtained by GA with no internal edges.)

4 Analysis

From the results, it can be seen that the GA can get a better result than the pulling and peeling methods in most instances and other instances it matches the results of the two methods. This shows that the genetic approach can be considered a viable solution to this problem, especially GARP with Ranking Mutation which shows a better result than using the GA with Random Mutation. Nevertheless, further study and research needs to be done to see how the parameters such as the rate of the different genetic operator, the size of the population and the number generations affect the result. The results also suggest that different parameters should be applied to GAs with different tetrahedralization methods because different tetrahedralization produces different structure of tetrahedralization and different number of tetrahedra. For instance, GA_{OP} might need a larger number of iterations (generations) to allow the GA to converge to the optimum because OP produces more internal edges than RP and pulling. At present, it is known that choosing a high percent for the elitism and crossover parameters will saturate the population with current "good" solutions. However, the current good solution may not be the optimum one, and the current "bad" solution may lead to the optimum solution but was pushed out by the elitism policy.

5 Conclusion

We have presented a genetic approach to find the minimum tetrahedralization of a convex polyhedron. The results show that genetic approach is always better than or equal to the pulling or peeling method. GA can even obtain the optimum solution for point sets for which the optimum is known.



$$\begin{array}{c|ccccc} & \multicolumn{5}{c}{j} \\ i & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 3 & 3 & 2 & 2 \\ 2 & 3 & 0 & 3 & 3 & 3 \\ 3 & 3 & 3 & 0 & 3 & 3 \\ 4 & 2 & 3 & 3 & 0 & 2 \\ 5 & 2 & 3 & 3 & 2 & 0 \end{array}$$

Figure 3:

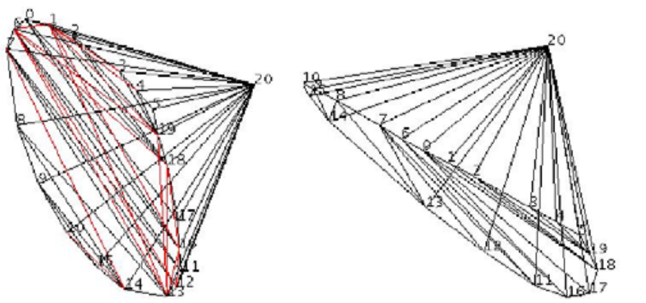


Figure 4: Stacked point set.

References

[1] B. Joe, *Three-Dimensional Triangulations From Local Transformations* , **SIAM J. SCI. STAT. COMPUT.** Vol 10, No. 4, pp 718-741 1989

[2] Below A., Brehm U., De Lorea J., and Richter-Gebert J., *Minimal Simplicial Dissections and Triangulations of Convex 3-Polytopes, Discrete and Computational Geometry 24*, 2000, pp.35-48.

[3] Bern M., *Compatible tetrahedralizations*, Proc. 9th Annu. ACM Sympos. Comput. Geom. pp. 281-288.

[4] Chin F., Feng S., and Wang C.A. (2001) *Approximation for Minimum Triangulations of Simplicial Convex 3-polytopes*, **Discrete & Computational Geometry** Vol.26, No.4, pp.499-511. (Oct. 12, 2001)

[5] Kaihuai Qin, Wenping Wang, Minglun Gong, *A Genetic Algorithm for the Minimum Weight Triangulation*, IEEE, 1997.

[6] Kolingerova I. *Genetic Approach to the Minimum Weight Triangulation* , **WSCG'98 Conference Proceedings** Volume II, Pilsen, Czech Republic, 1998, pp.184-191

[7] Lambert T., *Convex Hull Algorithms*, http://www.cse.unsw.edu.au/ lambert/java/3d/hull.html, 1998.

[8] Marek Obitko, *Introduction to Genetic Algorithms with Java Applets*, http://cs.felk.cvut.cz/ xobitko/ga/, 1998.

[9] Michalewicz, Zbigniew, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer, 1996.

[10] Rupert J. and Seidel R., *On the Difficulty of Tetrahedralizing 3-Dimensional Non-convex Polyhedra, Proceedings of the 5th ACM Symposium on Computational Geometry* (1989), pp. 380-392.