TETRAHEDRALIZATION OF SIMPLE AND NON-SIMPLE POLYHEDRA

Godfried T. Toussaint   *Clark Verbrugge*   *Caoan Wang*[*]   *Binhai Zhu*

School of Computer Science
McGill University

[*]Department of Computer Science
Memorial University of Newfoundland

ABSTRACT

It is known that not all simple polyhedra can be tetrahedralized, i.e., decomposed into a set of tetrahedra without adding new vertices (tetrahedralization). We investigate several classes of simple and non-simple polyhedra that admit such decompositions. In particular, we show that certain classes of rectilinear (isothetic) simple polyhedra can always be tetrahedralized in O(n^2) time where n is the number of vertices in the polyhedron. We also show that polyhedral slabs (even with holes) as well as subdivision slabs can always be tetrahedralized in O($n \log n$) time. Furthermore, for simple polyhedral slabs O(n) time suffices. We show that polyhedra that are the union of three convex polyhedra can always be tetrahedralized in O(n^2) time. Finally we show that if a polyhedron is the union of four or more convex polyhedra it does not necessarily admit a tetrahedralization.

1. Introduction

Decomposing a geometric object into simpler parts is one of the most fundamental problems in computational geometry. It is well known that every simple polygon can be triangulated, decomposed into triangles without adding new vertices (Steiner points) [Le11]. In fact, recently Chazelle showed that a simple polygon can be triangulated in linear time [Ch91]. The problem of tetrahedralizing a simple polyhedron in 3D without adding new vertices is significantly more difficult than its 2D counter part.

Schoenhardt gave a counterexample (the 6-vertex polyhedron illustrated in Fig.1) showing that it is not always possible to tetrahedralize a simple polyhedron [Sch28]. Bagemihl extended Schoenhardt's result by showing that there exist n-vertex simple polyhedron which cannot be tetrahedralized [Bag48] (Fig.2). Seidel gave a counterexample to show that not all simple rectilinear polyhedra can be tetrahedralized (Chapter 10 of [O'R87]). Recently Ruppert and Seidel showed that it is NP-complete to decide whether a simple polyhedron (even if it is known to be star-shaped) can be tetrahedralized [RS92]. Nevertheless Chazelle and Palios showed that if O(r^2) Steiner points are allowed, a simple polyhedron can be tetrahedralized into a linear number of tetrahedra in O($n \log n+r$) time, where r is the number of reflex edges of the given polyhedron [CP92].

Although it seems that tetrahedralizing a general simple polyhedron in 3D is an intractable problem, there are some results known regarding tetrahedralizing special classes of simple and non-simple polyhedra. It is well known for example that a convex polyhedron in 3D can always be tetrahedralized in linear time by simply connecting with line segments any vertex of the polyhedron to all the other vertices [Le11]. Also recently Goodman and Pach [GP88] proved that the class of simple polyhedra defined by $CH(P \cup Q) - P - Q$, where CH denotes convex hull, P, Q are both convex polyhedra and $P \cap Q = \varnothing$, can always be tetrahedralized. They also showed that the class of *non-simple* polyhedra defined by $P - Q$, such that Q, P are both convex polyhedra and $Q \subset P$ (we call such a poly-

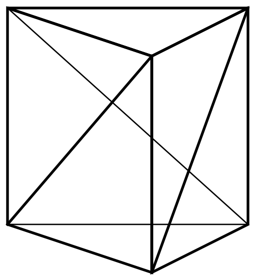


Fig. 1 Schoenhardt's counterexample.

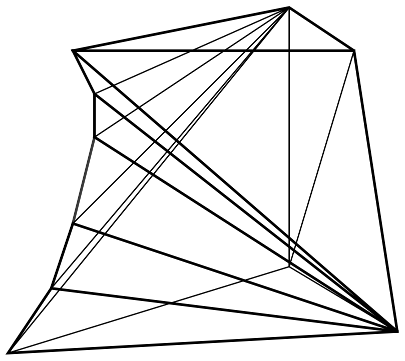


Fig. 2 Bagemihl's counterexample.

hedron a *convex annulus* henceforth), can always be tetrahedralized. Both of these algorithms have a time complexity of O(n^2). Bern [Be93] proved that the first algorithm is optimal and he proposed an O(n log n) time algorithm to improve the second one.

In this paper we show that the following classes of simple and non-simple polyhedra can always be tetrahedralized: a simple slab (with or without holes), a subdivision slab, a box with fixed depth rectilinear holes (we call it a type-1 box henceforth) and a box with linearly ordered rectangular holes (we call it a type-2 box henceforth). We also show that the union of three convex polyhedra can always be tetrahedralized while the union of four convex polyhedra need not admit a tetrahedralization.

2. Simple and Non-simple Slabs

We begin by giving some elementary definitions. A *simple slab S* is a simple polyhedron defined by translating a simple polygon F with m vertices by a fixed distance in an arbitrary direction thus defining another identical polygonal face F' such that both F and F' are faces of S and all other faces are parallelograms resulting from the insertion of edges between pairs of corresponding vertices of F and F'. If F is a triangle then S is called a *prism*. If F is a simple polygon with k holes we call the resulting non-simple polyhedron a *slab with k holes.*We show that an n-vertex slab with k holes can always be tetrahedralized. In a prism we say that two diagonals are *coincident* if they meet at a vertex. Note that if a prism has two coincident diagonals then we can tetrahedralize the prism in two different ways: we can triangulate the un-triangulated face (we call this face a *free* face henceforth) by inserting one of the diagonals arbitrarily. The following algorithm tetrahedralizes a slab with k holes.

Algorithm 1

begin

Step 1: Triangulate the polygon F with holes.

Step 2: Construct a graph G with k+1 vertices such that each node of the graph corresponds to a hole or the outermost face of the polygon. For every diagonal which connects two holes or connects a hole with the outermost face of F we insert an edge between the corresponding nodes of the graph. This gives us a connected planar graph for G. Compute a spanning tree of G. Label the k diagonals which correspond to the k edges of this spanning tree *cutting diagonals* and double these cutting diagonals. This results in a triangulated simple polygon (without holes).

Step 3: Visit every prism whose top has one cutting diagonal and insert *coincident* diagonals on the faces of the prism opposite to the face whose top edge is the cutting diagonal (this makes the face, incident to the cutting diagonal, a free face). These prisms are labelled *constrained* prisms because if one of the inserted diagonals is later flipped (if necessary) then so is its neighbor in order to preserve coincidence and the resulting freedom for inserting diagonals on its cutting diagonal face.

Step 4: Visit every prism whose top has two cutting diagonals and insert *coincident* diagonals on the faces of the prism containing the cutting diagonals. These prisms are labelled *frozen* prisms because these diagonals will never be changed again. If more than one triangle each containing two cutting diagonals have these cutting diagonals anchored on the same vertex then all inserted diagonals are made coincident on this vertex.

Step 5: Compute the dual tree of the triangulation on *F*.

Step 6: Starting with any leaf of this dual tree perform a depth-first search to visit all prisms. Triangulate all prisms as they are visited except for the *fixed* ones. If a prism is ordinary it can be triangulated arbitrarily. If the prism is *constrained* then coincidence should be respected.

end

Sketch of Proof of Correctness:

The only possible conflicts arise at prisms corresponding to leaves of the dual tree where we must make sure that the prisms on both sides of the cutting diagonals contain matching diagonals. First there is no face in the triangulation of the upper face with three cutting diagonals. This means that the spanning tree we obtain in *Step* (2) has a cycle. We call a prism corresponding to a leaf in the dual tree whose upper face contains one cutting diagonal, a *type*-1 prism; a prism corresponding to a degree-two node in the dual tree whose upper face contains one cutting diagonal a *type*-2 prism and a prism corresponding to a leaf in the dual tree whose upper face contains two cutting diagonals a *type*-3 prism.

Now we consider the following three cases:

(I) A type -1 prism shares the free face with another type-1 prism or a type-2 prism. According the *Step* (3), the top edge of the free face is a cutting diagonal and the free face can be triangulated arbitrarily. We only need to choose one triangulation to make the tetrahedralization of the two prisms consistent.

(II) A type-1 prism or a type-2 prism shares its free face with a type-3 prism. According to Step (4), this face is a face of the type-3 prism and is already triangulated by inserting a coincident diagonal. This gives us a triangulation of the free face (hence a tetrahedralization of the two prisms).

(III) A type-3 prism shares a cutting diagonal face with another type-3 prism. Again according to *Step* (4), the face of a type-3 prism whose top edge is not a cutting diagonal is a free face. If two (or more) prisms have their cutting diagonals anchored on the same vertex then all inserted diagonals are incident to this vertex and therefore the face separating the two prisms has a consistent triangulation. Otherwise we insert coincident diagonals on the non-free faces of one prism. Then according to the triangulation of the face separating the two prisms we insert coincident diagonals on the non-free faces of the other prism. For both of these two sub-cases any triangulation of the free faces gives us a tetrahedralization of the two prisms.

A complexity analysis establishes the following theorem.

Theorem 1: An n-vertex polyhedral slab with holes can be tetrahedralized in optimal O(n log n) time and linear space and a simple n-vertex slab can be tetrahedralized in optimal O(n) time.

Now we generalize Algorithm 1 to obtain an algorithm for tetrahedralizing a (bounded) subdivision slab. A (*bounded*) *subdivision slab* is built from two identical planar subdivisions lying on two parallel planes (without loss of generality, assume the two planes are parallel to the X-Y plane) such that every face which is not parallel to the X-Y plane is a parallelogram.

The proof of correctness is similar to that for Algorithm 1 and is therefore omitted. We show instead how to transform the problem so that we can apply Algorithm 1 directly.

Algorithm 2:

begin

Step 1: Triangulate the upper and lower faces of the subdivision slab.

Step 2: Compute a spanning tree of the dual graph of the triangulation. If an edge of the dual graph is not an edge of the spanning tree, then double the edge in the triangulation which corresponds to that dual graph edge. This gives us a triangulated polygon. To make the terminology consistent with what we used in Algorithm 1, we call these edges which are doubled *cutting diagonals.*

Step 3: Run Steps (3)-(6) of Algorithm 1.

end

Analysis of Algorithm 2:

Step (1) has a time complexity of O(n log n) and Step (2) takes O(n) time. The remaining steps take a total of O(n) time. Therefore the algorithm runs in O(n log n) time and linear space. We thus have the following theorem.

Theorem 2: An n-vertex subdivision slab can be tetrahedralized in O(n log n) time.

3. Rectilinear polyhedra

We mentioned in the introduction that it is not always possible to tetrahedralize a simple rectilinear polyhedron. Nevertheless we show in this section that certain special classes of rectilinear polyhedra can always be tetrahedralized.

We first give a definition of the two classes of simple rectilinear polyhedra to be investigated. A *type*-1 box is defined as follows: given a 3D rectilinear box with height H, "dig" a set of pairwise non-intersecting rectilinear polygonal holes at the top of the box such that all these holes have a unique depth h, such that $h < H$. We call the resulting simple rectilinear polyhedron a *type*-1 box (see Fig.3). A *type-2* box is defined similarly: given a 3D rectilinear box with height H, dig a set of non-intersecting rectangular holes (i.e. their orthogonal projections on the top of the box are a set of disjoint rectangles) on the top of the surface of the box such that the heights of the holes $h_i < H$ (for all i), and the holes are linearly ordered along a direction l on F, i.e. the orthogonal projections of the rectangles on the top of the box along l are a set of disjoint line segments on l (see Fig. 4). Since such an ordering of the holes (if it exists) can be computed in O(n log n) time by sorting, we assume that such an order is already known. We call such a simple rectilinear polyhedron a type-2 box. We give below two algorithms, Algorithms 3 and 4, to tetrahedralize a type-1 box and a type-2 box, respectively.

Algorithm 3

Step 1: Compute the convex hull of the rectilinear holes. Since the holes have the same depth, we obtain a slab with holes of height *h*.

Step 2: Run Algorithm 1 to tetrahedralize the above slab. We are left with a box with a hole *E* such that the bottom of the hole is a planar triangulation.

Step 3: Run the algorithm by Bern to tetrahedralize the polyhedron obtained in *Step* (2).

Note that after Steps (1) and (2) we have a rectilinear box with a hole such that there are coplanar edges on the hole. We can think of this as a convex annulus such that the outer polyhedron is a 6-vertex rectilinear polyhedron and the inner polyhedron is a convex a polyhedron (with coplanar edges). It can be shown that Bern's algorithm, which essentially cuts off convex vertices of the outer polyhedron one by one, also works for the above situation, i.e., even if there are coplanar edges on the inner convex polyhedron. We therefore have the following:

Theorem 3: An n-vertex type-1 box can be tetrahedralized in $O(n \log n)$ time and linear space.

Now we propose a quadratic time algorithm to tetrahedralize an n-vertex type-2 box with k holes: $p(1), p(2),...,p(k)$ (without loss of generality assume k is a power of 2). Though the holes have a linear order along l, if we tetrahedralize them incrementally we obtain an $O(n^3)$ time algorithm. Instead, we apply a divide-and-conquer strategy described as Algorithm 4.

Algorithm 4

Step 1: Tetrahedralize the region bounded by the holes using a divide-and-conquer strategy; i.e. tetrahedralize the regions formed by the first $k/2$ holes and by the last $k/2$ holes, then combine the results by applying Goodman and Pach's algorithm [GP88].

Step 2: Run the algorithm by Bern to tetrahedralize the convex annulus obtained in *Step* (1).

The only thing we need to show is that Step (1) always works. The crucial point is that the holes are linearly ordered. The first iteration will tetrahedralize the regions formed by $p(1), p(2),..., p(k\text{-}1), p(k)$. After the first iteration we have $k/2$ convex holes which are still linearly ordered along l. This property holds after every iteration. Therefore we can simply continue this recursive procedure until only one convex hole remains.

The analysis of Algorithm 4 is as follows: Step (2) runs in $O(n \log n)$ time. Step (1) can be measured by the following recurrence relation:

$$T(n) = 2\,T(n/2) + O(n^2) = O(n^2).$$

Therefore the time complexity of Algorithm 4 is $O(n^2)$.

Theorem 4: An n-vertex type-2 box can be tetrahedralized in $O(n^2)$ time.

4. Union of Convex Polyhedra

In this section we show the tetrahedralization of a class of polyhedra which is the union of convex polyhedra. A polyhedron is called a $U(k)$ polyhedron if it is the union of k convex polyhedra. Therefore trivially, $U(1)$ can always be tetrahedralized and it is straightforward to show that $U(2)$ can always be tetrahedralized. We will show that a $U(3)$ polyhedron can always be tetrahedralized while a $U(4)$ polyhedron can not always be tetrahedralized.

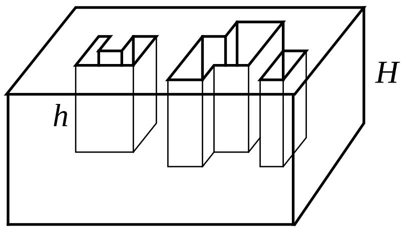


Fig. 3 A type-1 box.

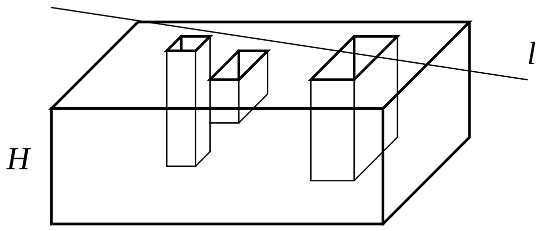
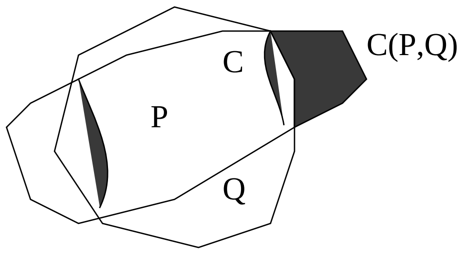


Fig. 4 A type-2 box.

Let P and Q be two convex polyhedra such that $P \cap Q$ is not empty. For simplicity, we will not consider the cases when P and Q do not have proper intersections, i.e., when $P \cap Q$ is only a point, a line segment or a polygonal face. Let C, which will be called a *crown* henceforth, be the simple closed polygonal chain on $P \cap Q$ as well as on the surfaces of both P and Q. Therefore $CH(C)$, the convex hull of the vertices of C, separates P into two parts. We call any of the two parts of P-$CH(C)$ a *convex cap* (with respect to Q) and denote it by $C(P,Q)$ (Fig.5).



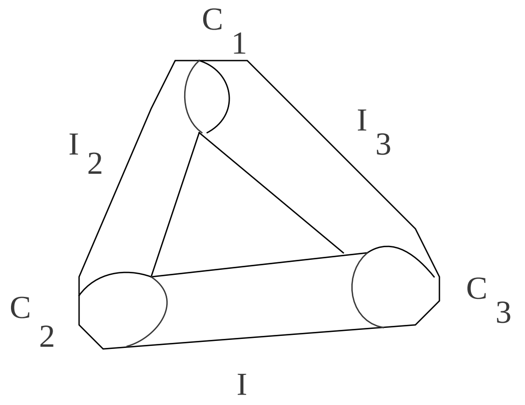


Fig. 5 A U(2) polyhedron and the illustration for a convex cap.

Fig. 6 An interlocked U(3) polyhedron.

Lemma 5: An n-vertex convex cap can be tetrahedralized in O($n$log$n$) time.

Proof: A convex cap can be seen as a special case of a convex annulus, which can always be tetrahedralized in O($n$log$n$) time [Be93].

We say a U(3) polyhedron with no convex cap is an *interlocked U(3)* polyhedron (Fig. 6). By definition, we know that an interlocked U(3) polyhedron must nest each other.

Lemma 6: An n-vertex interlocked U(3) polyhedron can be tetrahedralized in O(n^2) time.

Proof: (Refer to Fig.6) Let I denote an interlocked $U(3)$ polyhedron, and let C_1, C_2 and C_3 be its three crowns. Then $CH(C_1)$, $CH(C_2)$ and $CH(C_3)$ separate I into three parts, I_1,I_2 and I_3. $I_1 \cup I_2 \cup CH(C_1) \cup CH(C_2) \cup CH(C_3)$ is a U(2) polyhedron, which can be tetrahedralized in linear time. I_3 is a convex polyhedron with two disjoint convex holes, which can be tetrahedralized by first running Goodman and Pach's algorithm [GP88] to tetrahedralize the part between the two convex holes, then tetrahedralize the remaining polyhedron (which is a convex annulus) by running Bern's algorithm [Be93].

Theorem 7: An *n*-vertex $U(3)$ polyhedron can be tetrahedralized in $O(n^2)$ time.

Proof: (Refer to Fig.7) If there exists a vertex of the $U(3)$ polyhedron U_3 which is the union of P_1,P_2 and P_3, then it is trivial that U_3 can be tetrahedralized. Otherwise, every vertex of U_3 belongs to at most two original polyhedra. In this case the crowns of P_1 and P_2 do not intersect those of P_2 and P_3 and those of P_1 and P_3. Now we remove and tetrahedralize all the convex caps (if any), the remainder is either a convex polyhedron (case (a)) or an interlocked $U(3)$ polyhedron (case (b)), which can all be tetrahedralized. The $O(n^2)$ time complexity of this algorithm (will be given in the following paragraph) follows from Lemma 6.

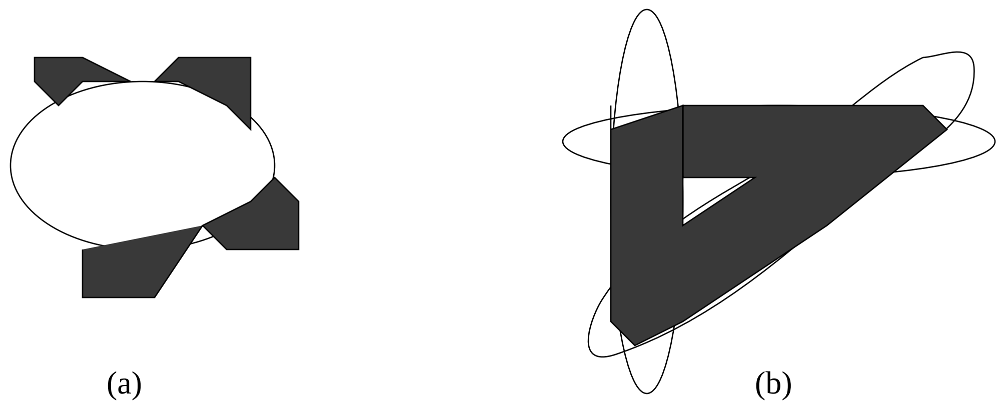


Fig. 7 The illustration for the proof of Theorem 7.

Algorithm 5

Step 1: Identify all the crowns by traversing the refles edges of U_3. Compute the convex hull of each crown, consequently we have all the convex caps.

Step 2: Remove and tetrahedralize all the convex caps.

Step 3: If the remainder is a convex polyhedron, tetrahedralize it directly.

Otherwise, the remainder is an interlocked $U(3)$ polyhedron. Use the method in

Lemma 6 to tetrahedralize it in quadratic time.

We have shown that a $U(3)$ polyhedron can always be tetrahedralized. Now we will give a counterexample to show that a $U(4)$ polyhedron is not always tetrahedralizable. The example is essentially based on the Shoenhardt polyhedron, which can be decomposed into four disjoint convex polyhedra (no two convex polyhedra intersect except on their surface) (Fig.8). Consequently we have shown that the Shoenhardt polyhedron is the union of four convex polyhedra.

Theorem 8: A $U(4)$ polyhedron can not always be tetrahedralized.

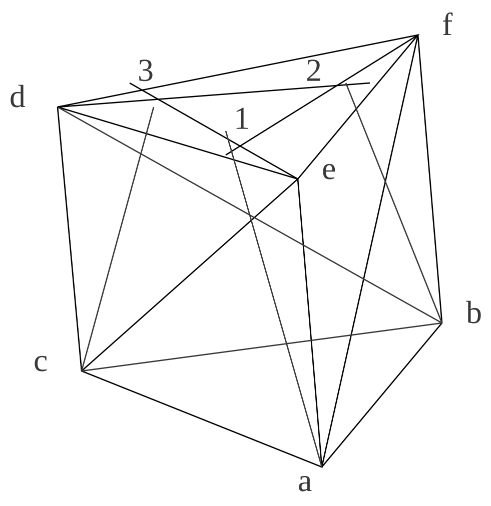


Fig.8 The shoenhardt polyhedron can be decomposed into 4 disjoint convex polyhedron: P(aef1), P(bdf2), P(cde3) and P(abc123).

5. Concluding Remarks

We follow in the footsteps of Goodman and Pach[GP88] as well as Bern [Be93] by showing in this paper that certain special classes of simple and non-simple polyhedra can always be tetrahedralized. We also provide efficient algorithms for obtaining tetrahedralizations of polyhedra belonging to these classes. One direction for future research is to find other nontrivial classes of polyhedra which can be tetrahedralized. For example, if we modify the definition of a type-2 box such that the holes do not have a linear order and call the resulting polyhedron a *type*-3 box, then it is not always possible to tetrahedralize a type-3 box by simply running Goodman and Pach's algorithm.

6. Acknowledgments

The authors thank Jit Bose, Subhash Suri and Marc van Kreveld for discussions.

7. References

[AAP86] Asano,T., Asano, T. and Pinter, R., "Polygon triangulation: efficiency and minimality," *J. Algorithms*, vol. 7, 1986, pp. 221-231.

[Ba48] Bagemihl, F., "On indecomposable polyhedra," *American Mathematical Monthly,* September 1948, pp. 411-413.

[Be93] Bern, M., "Compatible tetrahedralizations," *9th ACM Computational Geometry Conf.* San Diego, CA, May, 1993.

[Ch91] Chazelle, B., "Triangulating a simple polygon in linear time," *Discrete & Computational Ge-*

ometry, vol. 6, No. 5, 1991, pp. 485-524.

[CP90] Chazelle, B. and Palios, L., “Triangulating a nonconvex polytope,” *Discrete & Computational Geometry*, vol. 5, 1990, pp. 505-526.

[GP88] Goodman, J. and Pach, J., “Cell decomposition of polytopes by bending,” *Israel J. Mathematics*, vol. 64, No. 2,1988, pp. 129-138.

[Le11] Lennes, N. J., “Theorems on the simple finite polygon and polyhedron,” *American Journal of Mathematics*, vol. 33, 1911, pp.37-62.

[O’R87] O’Rourke, J., *Art Gallery Theorems and Algorithms*, Oxford University Press, 1987.

[RS92] Ruppert, J. and Seidel, R., “On the difficulty of triangulating three-dimensional nonconvex polyhedra,” *Discrete and Computational Geometry*, vol. 7, 1992, pp. 227-253.

[Sc28] Schoenhardt, E., “Uber die Zerlegung von Dreieckspolyedern in Tetraeder,” *Mathematische Annalen*, vol. 98, 1928, pp. 309-312.