# Generating Rooted Triangulations with Minimum Degree Four

David Avis Chiu Ming Kong

**Abstract**

A graph is a triangulation if it is planar and every face is a triangle. A triangulation is rooted if the external triangular face is labelled. Two rooted triangulations with the same external face labels are isomorphic if their internal vertices can be labelled so that both triangulations have identical edge lists.

In this article, we show that in the set of rooted triangulations on $n$ points with minimum degree four, there exists a target triangulation $E_n^*$ such that any other triangulation $E_n \neq E_n^*$ in the set can be transformed to $E_n^*$ via a finite sequence of single and double diagonal transformations. Using this result with the reverse search technique, we present an algorithm for generating all non-isomorphic rooted triangulations on $n$ points with minimum degree four. The triangulations are produced without repetitions in $O(n^2)$ time per triangulation. The algorithm uses $O(n)$ space.

## 1 Background

### 1.1 Maximal Planar Graphs and Diagonal Transformation

A simple planar graph is **maximal planar** if and only if it is triangulated. Hence a **triangulation** of a set of points in the plane is a maximal planar graph. In a **maximal planar graph**(**MPG**) $G = (V, E)$, $|E| = 3|V| - 6$.

An edge in a triangulation is **external** if it is contained in exactly one triangle (not the external face) of the triangulation. The vertices of the external edges are **external vertices**. Vertices that are not external are **internal vertices**. An edge in a triangulation is **internal** if it is contained in exactly two triangles (neither is the external face) of the triangulation. Such an edge **bounds** the two triangles.

The objects we want to generate are **rooted triangulations with minimum degree four** on $n$ points (given $n$), which, hereafter, will be referred to as **rooted maximal planar graphs with minimum degree four** (**rooted MPG4**).

In a maximal planar graph $G$, let $\triangle abc$ and $\triangle abd$ be two adjoining triangular faces. The edges of these faces form a quadrangle $acbd$ with the diagonal $(a, b)$. When this diagonal is replaced by $(c, d)$, one obtains a new maximal planar graph $G'$ with the same vertices and the same number of edges and faces. We say that $G'$ has been obtained from $G$ by a **diagonal transformation** or a **diagonal flip**. This is possible only if $(c, d)$ is not an edge in $G$ already.

Additional details may be found in [4, 6].

### 1.2 Reverse Search Essentials

The reverse search method is developed by Avis and Fukuda [2]. It was used to generate all rooted triangulations (without degree constraint) by Avis [1] and dually to generate all 3-polytopes by Deza et al [3]. It is a technique for generating all vertices of a graph whose edges are given implicitly by an oracle. Let $H = (T, U)$ be such a graph. In our application, each vertex in $T$ corresponds to a rooted MPG4 with given $n$. $U$ is the edge set of adjacent vertices in $T$. Two rooted MPG4 are adjacent if they differ by one edge or two edges. The method works by finding a spanning tree in the graph $H$. To do this we first fix some given rooted MPG4 with the required parameter $n$. Call

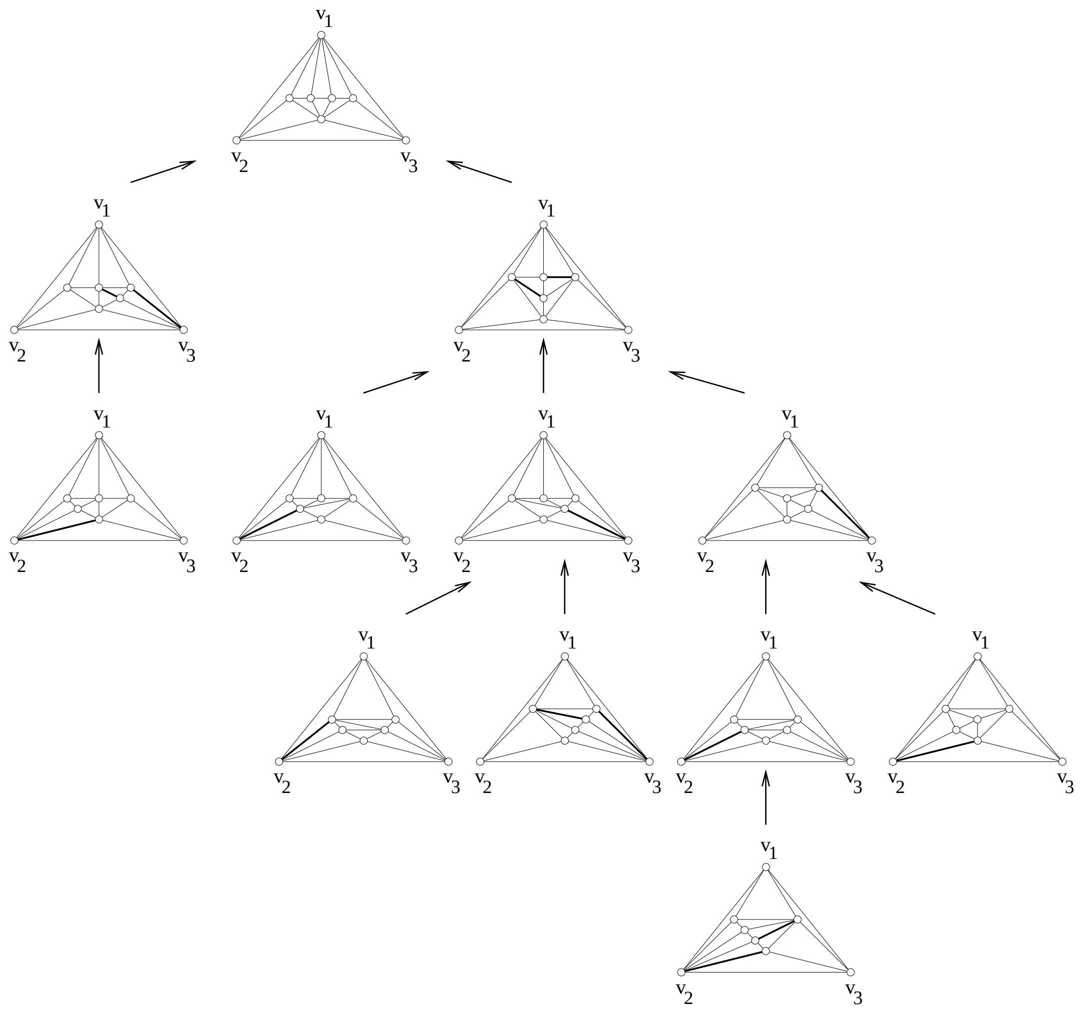


Figure 1: Reverse search tree of rooted MPG4 for $n = 8$

the vertex in $H$ corresponding to this triangulation the **target** $E_n^*$. We then define a **local search** function $L$, that given any vertex $E_n$ of $H$ defines a unique adjacent vertex in $H$, with the property that repeated application of $L$ defines a path in $H$ to the target. In other words, $(E_n, L(E_n))$ is an edge in $U$ and $L^k(E_n) = E_n^*$ for some finite integer $k$. The path generated consists of a sequence of rooted MPG4 (for given $n$) that differ in one edge or two edges and ends with the target MPG4. The set of all such paths form a spanning tree called the **reverse search tree** in $H$. Figure 1 shows the reverse search tree of rooted MPG4 for $n = 8$. The local search function is described in section 3. The arrows show the traces of applications of the local search function. The bold-faced edges are those on which diagonal flips are performed.

The reverse search procedure is essentially a depth first search. It is initiated at the target $E_n^*$ and traverses the reverse search tree of $H$ by *reversing* the local search function. To do this we generate all neighbours in $H$ of any given vertex $E_n$ in $T$, in some given order. This is done by an **adjacency oracle**. Using the adjacency oracle at $E_n^*$ we consider neighbours of $E_n^*$ until we find a neighbour $E_n$ such that $L(E_n) = E_n^*$. We then replace $E_n^*$ by $E_n$ and use the adjacency oracle to find (if possible) a neighbour $E_n'$ of $E_n$ such that $L(E_n') = E_n$. If such a vertex exists we move to $E_n'$ and continue. If we reach a node $E_n$ for which no such neighbour exists, we backtrack by computing the parent $E_n^0 = L(E_n)$ of $E_n$. We then continue from $E_n^0$ using the adjacency oracle to

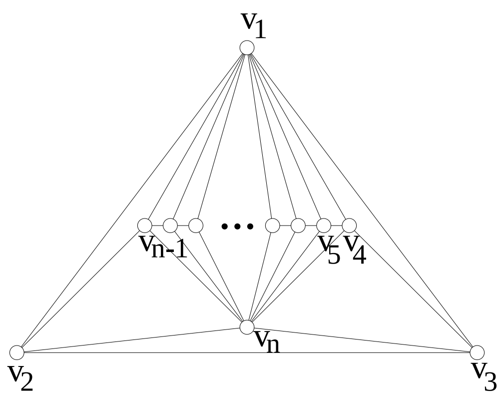

Figure 2: Target Triangulation

give the next neighbour of $E_n^0$ in the order after $E_n$.

Hence it is necessary to specify:

- the target MPG4,
- the local search function, and
- the adjacency oracle.

## 2 Target Triangulation

Let $E_n$ be a rooted MPG4 with vertex set $\{v_1, v_2, ..., v_n\}$. All rooted MPG4 mentioned hereafter are assumed to be rooted at $\{v_1, v_2, v_3\}$. To avoid the trivial case we assume $n > 6$. Let $d(v)$ denote the degree of vertex $v$.

The target triangulation $E_n^*$ for $n > 6$ is a *stand* holding a *gem* as shown in figure 2. It is the unique rooted triangulation with degree sequence:

$$d(v_1) = n - 2, d(v_2) = d(v_3) = d(v_4) = \cdots = d(v_{n-1}) = 4, d(v_n) = n - 2$$

## 3 Local Search

In general, given a rooted MPG4 $E_n \neq E_n^*$, we first build the stand by reducing $d(v_2)$ and $d(v_3)$ to 4, and then reduce the degrees of the vertices in the horizontal "chain" in the gem to 4, working from right to left along the "chain".

We say that a vertex $v$ has **consecutive neighbours** $u_1, u_2, \ldots, u_m$ if these neighbours occur in consecutive counter-clockwise order in the unique planar embedding of the triangulation. An edge is said to be **1-flippable** if and only if it is internal and a diagonal transformation can be performed on it without destroying the minimum-degree-4 property of the triangulation. Two edges are said to be **2-flippable** if and only if they are internal and a diagonal transformation can be performed on each of them simultaneously without destroying the minimum-degree-4 property of the triangulation. The following lemmas describe some properties of MPG4, which motivate our Local Search function.

**Lemma 3.1** *Let $s$ be a vertex in a MPG4 such that $d(s) \geq 5$. Suppose $s$ has five consecutive neighbours $u_1, u_2, u_3, u_4, u_5$ such that $u_1$ is external, and $u_2, u_3, u_4$ are internal. Then either $d(\{u_2, u_3, u_4\}) \geq 5$, or there exists a vertex $t \in N^2(s) \bigcap_{j=1}^{5} N(u_j)$.*

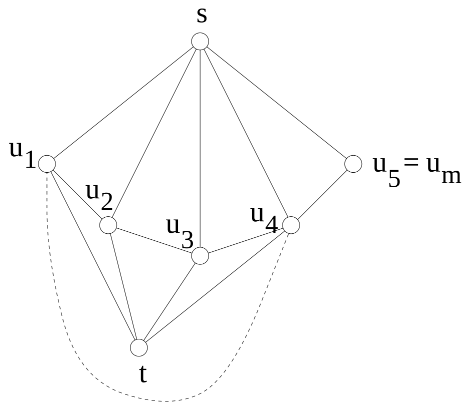


Figure 3: Illustrating Lemma 3.2

**Proof**: Suppose $d(u_2) = d(u_3) = d(u_4) = 4$. Edges $(u_1, u_2), (u_2!(u_3, u_4), (u_4, u_5)$ are internal. Let $(u_1, u_2)$ bound $\triangle u_1u_2s$ and $\triangle u_1u_2t$ for some vertex $t$. Let $(u_2, u_3)$ bound $\triangle u_2u_3s$ and $\triangle u_2u_3t'$ for some vertex $t'$. Then for $d(u_2) = 4$, it must be that $t = t'$. Likewise, $(u_3, u_4)$ bounds $\triangle u_3u_4s$ and $\triangle u_3u_4t$; $(u_4, u_5)$ bounds $\triangle u_4u_5s$ and $\triangle u_4u_5t$. Hence $t \in N^2(s) \bigcap_{j=1}^{5} N(u_j)$. □

**Lemma 3.2** *Let $s$ be a vertex in a MPG4 such that $d(s) = m \geq 5$. Suppose $s$ has consecutive neighbours $u_1, u_2, u_3, \ldots, u_{m-1}, u_m$ such that $u_1$ is external, and $u_2, u_3, \ldots, u_{m-1}$ are internal. If $d(u_2) = d(u_3) = \cdots = d(u_{m-2}) = 4$ and $d(u_{m-1}) \geq 5$, then either there exists a vertex $t \in N^2(s) \bigcap_{j=1}^{m-1} N(u_j)$ such that $d(t) \geq 5$, or $(u_1, u_{m-1})$ is 1-flippable. Further, $(s, u_{m-1})$ is 1-flippable.*

**Proof**: If $m > 5$: By the argument in Lemma 3.1, there exists a vertex $t \in N^2(s) \bigcap_{j=1}^{m-1} N(u_j)$. Since $(m-1) \geq 5$, $d(t) \geq 5$.
If $m = 5$ (*cf* figure 3): By the argument in Lemma 3.1, there exists a vertex $t \in N^2(s) \bigcap_{j=1}^{4} N(u_j)$. If $d(t) = 4$, it must be that $t, u_1, u_4$ form a triangle in the triangulation. Since $(u_1, u_4)$ is internal, it bounds $\triangle u_1u_4t$ and $\triangle u_1u_4x$ for some $x \neq t$ and $(x, t)$ is not an edge in the triangulation. Now since $d(u_1) \geq 5$ and $d(u_4) \geq 5$, $(u_1, u_4)$ is 1-flippable.

In both cases, since $(u_{m-2}, u_m)$ cannot be an edge in the triangulation, $(s, u_{m-1})$ is 1-flippable. □

**Lemma 3.3** *Let $s$ be a vertex in a MPG4 such that $d(s) \geq 5$. Suppose $s$ has four consecutive neighbours $u_1, u_2, u_3, u_4$ such that $u_2, u_3$ are internal, and $d(u_2) \geq 5$. Then either $(s, u_2)$ is 1-flippable, or $(s, u_3)$ is 1-flippable. Further, if $(s, u_2)$ is not 1-flippable, then $d(u_1) \geq 5$.*

**Proof**: Suppose $(s, u_2)$ is not 1-flippable. It must be that $(u_1, u_3)$ is an edge in the triangulation. If $u_4$ lies outside $\triangle su_1u_3$(*cf* figure 4): Since $d(u_2) \geq 5$, $\triangle u_2u_1u_3$ is not empty. Hence there exists a vertex $a$ inside $\triangle u_2u_1u_3$ and $a, u_1, u_3$ form a triangle in the triangulation. Hence $d(u_3) \geq 5$. Since $(u_2, u_4)$ cannot be an edge in the triangulation, $(s, u_3)$ is 1-flippable. Since $(u_1, u_3)$ is internal, it bounds $\triangle bu_1u_3$ for some vertex $b$ outside $\triangle su_1u_3$. Hence $d(u_1) \geq 5$. If $u_4$ lies inside $\triangle su_1u_3$(*cf* figure 5): Since$(u_1, u_3)$ is internal, it bounds $\triangle au_1u_3$ for some vertex $a$ inside $\triangle su_1u_3$ ($a$ may or may not be $u_4$), and $\triangle bu_1u_3$ for some vertex $b$ outside $\triangle u_1u_3u_2$. Hence $d(u_1) \geq 5$ and $d(u_3) \geq 5$. Since $(u_2, u_4)$ cannot be an edge in the triangulation, $(s, u_3)$ is 1-flippable. □

**Lemma 3.4** *Let $s$ be a vertex in a MPG4 such that $d(s) \geq 5$. Suppose $s$ has four consecutive neighbours $u_1, u_2, u_3, u_4$ such that $u_1$ is external, $u_2, u_3, u_4$ are internal, and $d(u_2) = d(u_3) = 4$. Let $t \neq s$ be the common neighbour of $u_1, u_2, u_3, u_4$, and $d(t) \geq 5$. Then $(s, u_2)$ and $(t, u_3)$ are 2-flippable.*

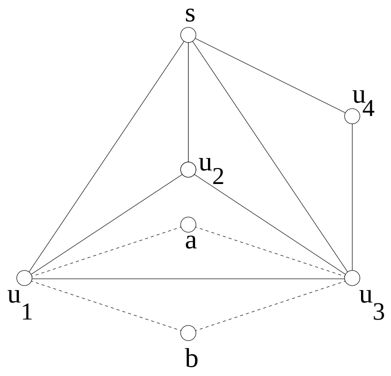


Figure 4: Illustrating Lemma 3.3

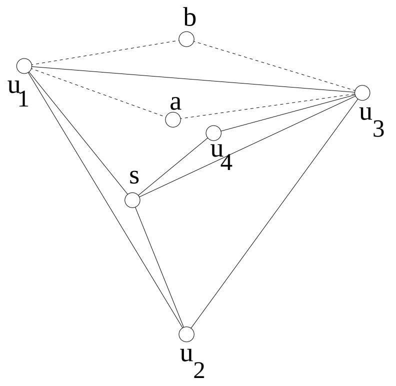


Figure 5: Illustrating Lemma 3.3

**Proof**: $(u_1, u_3)$ and $(u_2, u_4)$ cannot be edges in the triangulation. If we flip $(s, u_2)$ and $(t, u_3)$, the new edges are $(u_1, u_3)$ and $(u_2, u_4)$. It can be verified that only $d(s)$ and $d(t)$ are decreased by 1. The minimum-degree-4 property is preserved and so $(s, u_2)$ and $(t, u_3)$ are 2-flippable. □

Before presenting the Local Search function, let us illustrate it with an example. The idea is to compare first, the degrees of vertices $v_2$ and $v_3$ to their corresponding degrees in the target, and then the degrees of vertices along the horizontal "chain" in the gem to their corresponding degrees in the target, starting with the vertex adjacent to $v_1$ and $v_3$, and examine the neighbours of $v_1$ in clockwise order.

Figure 7 shows an example with $n = 11$. $E^1$ has $d(v_3) = 5$. Since $d(t) < 5$, we apply Lemma

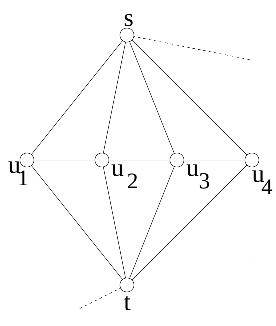


Figure 6: Illustrating Lemma 3.4

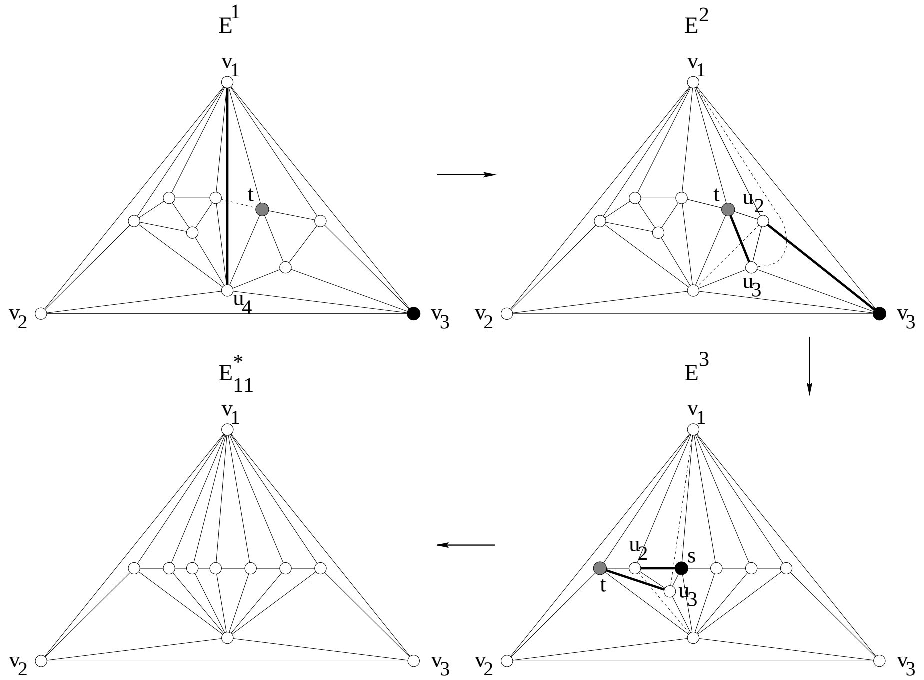


Figure 7: Illustrating the local search function

3.2 and flip $(v_1, u_4)$ to get $E^2$. In $E^2$, $d(v_3)$ is still $> 4$. We can apply Lemmas 3.2 and 3.4, flip $(v_3, u_2)$ and $(t, u_3)$ to get $E^3$. We see that $d(s) = 5$ in $E^3$. Again we can apply Lemmas 3.2 and 3.4, flip $(s, u_2)$ and $(t, u_3)$ and then reach our target $E_{11}^*$.

Figure 8 shows the pseudo code for the Local Search function `LocalSearch`. In our notation, `deg(v)` denotes $d(v)$, and `v[i]`, `u[j]` denote vertices $v_i$ and $u_j$ respectively. Hereafter, "application" of the function implies the diagonal transformation follows on the returned edge(s).

The next theorem shows that repeated application of the function leads any rooted MPG4 to the target.

**Theorem 3.1**

*i. Edges returned by* `LocalSearch` *are either 1-flippable or 2-flippable.*

*ii. Successive application of* `LocalSearch` *transforms any rooted MPG4 $E_n (n \geq 6$ and $E_n \neq E_n^*)$ to the target MPG4 $E_n^*$.*

**Proof**: For part $i$, we use the lemmas. Whenever two edges are returned, we apply Lemmas 3.2 and 3.4. Whenever $(v_i, u_j)$, $(v_i, u_{j+1})$, $(s, u_j)$ or $(s, u_{j+1})$ is returned, we apply Lemma 3.3. Whenever $(v_i, u_{m-1})$ or $(u_1, u_{m-1})$ is returned, we apply Lemma 3.2. In all cases, $s$ in the lemmas will be the vertex $v_i$ or $s$ in `LocalSearch`. By the lemmas, the returned edges are either 1-flippable or 2-flippable.

Hence, we have shown that application of `LocalSearch` on any rooted MPG4 always results in another rooted MPG4 (for given $n$).

For part $ii$: for $2 \leq i \leq 3$, the algorithm reduces $d(v_i)$ by one, by returning a 1-flippable edge incident to $v_i$; or it returns some edge, flipping which leads to another MPG4, on which subsequent application of `LocalSearch` returns a 1-flippable edge incident to $v_i$, hence reducing $d(v_i)$ by one.

To show that the process of reducing $d(v_3)$ does not increase $d(v_2)(= d(u_m))$, we will go through all the cases.

```
LocalSearch(E,n) // returns one(or two) flippable edge(s) or (-1) if E==target

// reduce deg(v[2]) and deg(v[3]) to 4
i=2; while (deg(v[i])==4  &&  i<=3) i++;
if (i<=3){ Let u[1],u[2],...,u[m] be the m consecutive neighbours of v[i].
           u[1] is external, u[2],...,u[m-1] are internal.  // deg(v[i])==m>=5
           if (deg(u[j])>=5 for some minimal j where 2<=j<(m-1))
             { if (u[j-1],u[j+1]) is not an edge return((v[i],u[j]));
               else return((v[i],u[j+1]));
             }
           if (deg(u[j])<5 for all 2<=j<(m-1)  &&  deg(u[m-1])>=5)
             { case (i) of
               2: return((v[i],u[m-1]));
                  // cannot return (v[i],u[m-1]) if i==3, otherwise deg(v[2])
                  // will go up
               3: Let t!=v[i] be the common neighbour of u[1],u[2],...,
                  u[m-1].
                  if (deg(t)<5) return((u[1],u[m-1]));
                  else return((v[i],u[2]) and (t,u[3]));
             }
           if (deg(u[j])<5 for all 2<=j<=(m-1))
             { Let t!=v[i] be the common neighbour of u[1],u[2],...,u[m].
               return((v[i],u[2]) and (t,u[3]));
             }
         }
// build the gem
Let s be the vertex adjacent to v[1] and v[3], lasts be v[3].
while (deg(s)==4  &&  s!=the vertex adjacent to v[1] and v[2])
     { lasts=s; s=the neighbour of v[1] previous to s in counter-clock-
       wise order;
     }
if (s==the vertex adjacent to v[1] and v[2]) return (-1); // E==target

Let u[1]=v[1], u[2],...,u[m]=lasts be the m consecutive neighbours
of s.  // deg(s)==m>=5
if (deg(u[j])>=5 for some minimal j where 2<=j<(m-1))
  { if (u[j-1],u[j+1]) is not an edge return((s,u[j]));
    else return((s,u[j+1]));
  }
if (deg(u[j])<5 for all 2<=j<(m-1)  &&  deg(u[m-1])>=5)
  { // cannot return (s,u[m-1]), otherwise deg(lasts) will go up
    Let t!=s be the common neighbour of u[1],u[2],...,u[m-1].
    if (deg(t)<5) return((u[1],u[m-1]));
    else return((s,u[2]) and (t,u[3]));
  }
if (deg(u[j])<5 for all 2<=j<=(m-1))
  { Let t!=s be the common neighbour of u[1],u[2],...,u[m].
    return((s,u[2]) and (t,u[3]));
  }
```

Figure 8: Pseudo code for the Local Search function

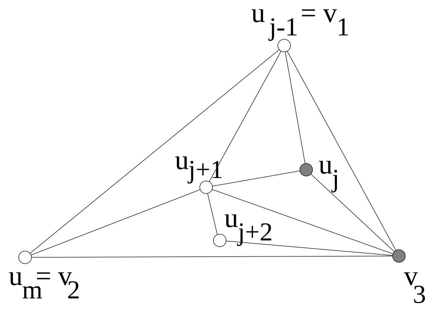


Figure 9: Illustrating Theorem 3.1

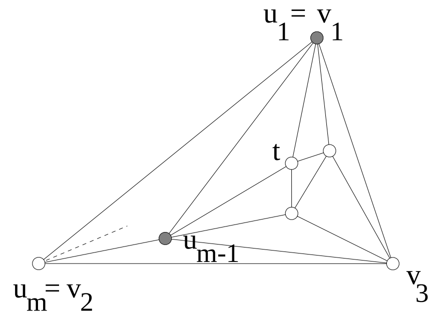


Figure 10: Illustrating Theorem 3.1

If ($d(u_j) \geq 5$ for some minimal $j$ where $2 \leq j < (m-1)$):

If $(v_i, u_j)$ is returned, the new edge is $(u_{j-1}, u_{j+1})$ where $m \neq j-1$ or $j+1$; else, $(u_{j-1}, u_{j+1})$ is an edge in the triangulation. By Lemma 3.3 (with $v_3, u_j$ as $s, u_2$ respectively in the lemma), $d(u_{j-1}) \geq 5$, and so $u_{j-1}$ must be $v_1$ (*cf* figure 9). Otherwise, it contradicts the fact that $j$ is minimal. Now since $d(v_3) > 4$, we know that $u_{j+2} \neq v_2$ exists. After flipping $(v_3, u_{j+1})$, the new edge is $(u_j, u_{j+2})$.

If ($d(u_j) < 5$ for all $2 \leq j < (m-1)$ and $d(u_{m-1}) \geq 5$):

If $(v_i, u_2)$ and $(t, u_3)$ are returned, the new edges are $(u_1, u_3)$ and $(u_2, u_4)$. If $(u_1, u_{m-1})$ is returned, the new edge is $(t, b)$ where $b \neq v_2$ (*cf* figure 10): since $\triangle v_3 u_{m-1} v_2$ is a triangle in the triangulation, in order that $d(v_2) = 4$, there exists at least one vertex in $\triangle u_1 v_2 u_{m-1}$, and so there is some vertex $b \neq v_2$ which forms a triangle with $u_1$ and $u_{m-1}$ in the triangulation.

If ($d(u_j) < 5$ for all $2 \leq j \leq (m-1)$):

The new edges are $(u_1, u_3)$ and $(u_2, u_4)$.

When $d(v_2)$ and $d(v_3)$ reach 4, the algorithm works on the vertices that are internal neighbours of $v_1$ in clockwise order. Next we will show that, when $s=$ the vertex adjacent to $v_1$ and $v_3$, and $lasts = v_3$, the process of reducing $d(s)$ does not increase $d(lasts)$ or $d(v_2)$.

If ($d(u_j) \geq 5$ for some minimal $j$ where $2 \leq j < (m-1)$):

If $(s, u_j)$ is returned, the new edge is $(u_{j-1}, u_{j+1})$ where both $u_{j-1}$ and $u_{j+1}$ are internal; else $(u_{j-1}, u_{j+1})$ is an edge in the triangulation. By Lemma 3.3 (with $s, u_j$ as $s, u_2$ respectively in the lemma), $d(u_{j-1}) \geq 5$, and so $u_{j-1}$ must be $v_1$ (*cf* figure 11). Otherwise, it contradicts the fact that $j$ is minimal. Now since $d(s) > 4$, we know that $u_{j+2} \neq v_3$ exists. $u_{j+2}$ cannot be $v_2$: otherwise, either $d(v_3)$ would be equal to three, or there exists at least one vertex inside $\triangle s v_2 v_3$ so that $d(v_2)$ would be greater than four. After flipping $(s, u_{j+1})$, the new edge is $(u_j, u_{j+2})$.

If ($d(u_j) < 5$ for all $2 \leq j < (m-1)$ and $d(u_{m-1}) \geq 5$):

If $(s, u_2)$ and $(t, u_3)$ are returned, the new edges are $(u_1, u_3)$ and $(u_2, u_4)$. If $(u_1, u_{m-1})$ is

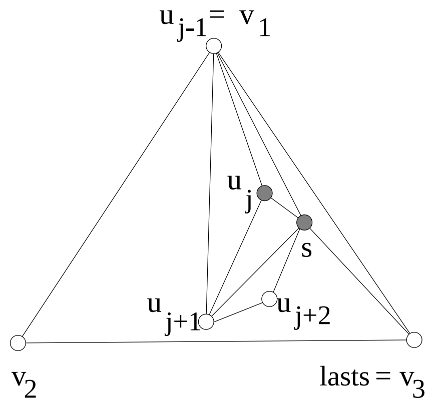


Figure 11: Illustrating Theorem 3.1

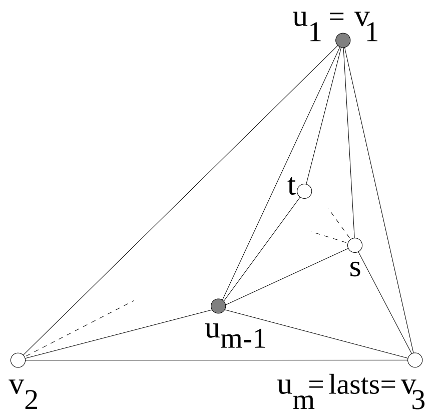


Figure 12: Illustrating Theorem 3.1

returned(*cf* figure 12), the new edge is $(t, b)$ where $b$ cannot be $v_3$: since $d(v_3) = 4$, $u_{m-1}, v_2, v_3$ must form a triangle in the triangulation. $b$ cannot be $v_2$ either, because for $d(v_2)$ to be equal to 4, there exists at least one vertex inside $\triangle v_1 v_2 u_{m-1}$.
If $(d(u_j) < 5$ for all $2 \leq j \leq (m-1))$:

The new edges are $(u_1, u_3)$ and $(u_2, u_4)$.

By the structure of the target triangulation we defined, the above argument applies to subsequent $s$'s for $lasts \neq v_3$ as well, since we can assume that previous $lasts$'s are erased from the graph and this part will not be affected by the algorithm anymore.

The process stops once the degrees of the vertices match their counterparts in the target, and so the algorithm is finite. $\square$

## 4 Adjacency Oracle

Suppose $E_n$ has its edges indexed from 1 to $3n-6$, and let $e_1, e_2, e_3$ be the 3 external edges. The edges are stored in some ordered edge list $E_n = \{e_1, e_2, \ldots, e_{3n-6}\}$.

Suppose a single diagonal flip is performed on some internal edge $e$. The new edge $e'$ is well defined and need not be specified explicitly. The operation is: $\text{flip}_1(E_n, e) = E_n - e + e'$.

We know that not only a single diagonal flip, but two simultaneous diagonal flips can also define

| $n$ | #rooted MPG4 | #rooted MPG5 | time taken/sec. | #rooted MPG | time taken/sec. |
|---|---|---|---|---|---|
| 4 | - | - | - | 1 | 0 |
| 5 | - | - | - | 3 | .002 |
| 6 | 1 | - | 0 | 13 | .003 |
| 7 | 3 | - | 0 | 68 | .010 |
| 8 | 12 | - | .007 | 399 | .060 |
| 9 | 59 | - | .037 | 2,530 | .506 |
| 10 | 313 | - | .255 | 16,965 | 4.355 |
| 11 | 1,713 | - | 1.850 | 118,668 | 37.762 |
| 12 | 9,559 | 1 | 13.224 | 857,956 | 330.423 |
| 13 | 54,189 | 0 | 92.120 | 6,369,883 | 2911.029 |
| 14 | 311,460 | 6 | 638.280 | ? | ? |
| 15 | 1,812,281 | 13 | 4459.693 | ? | ? |
| 16 | 10,661,303 | 55 | 30424.496 | ? | ? |
| 17 | 63,336,873 | 189 | 207669.911 | ? | ? |

Table 1: Results available

adjacency. Suppose a double diagonal flip is performed. The two edges flipped must be the two pairs of opposite sides of some quadrangle in the triangulation. Each quadrangle is defined by one internal edge in the triangulation. Hence for each internal edge in the triangulation, we also consider two pairs of edges. We can define the adjacency oracle which gives all adjacent rooted MPG4 to a given rooted MPG4 $E_n$ as follows:
For $j = 4$ to $(3n-6)$, let $e_j = (a,b)$ bound $\triangle abc$ and $\triangle abd$. Define

$$\mathrm{Adj}_1(E_n, j) = \begin{cases} \mathrm{flip}_1(E_n, e_j) \text{ if } e_j \text{ is 1-flippable} \\ \emptyset \text{ otherwise} \end{cases}$$

$$\mathrm{Adj}_2(E_n, j) = \begin{cases} \mathrm{flip}_1(E_n, (a,c)) \text{ and } \mathrm{flip}_1(E_n, (b,d)) \text{ if } (a,c) \text{ and } (b,d) \text{ are 2-flippable} \\ \emptyset \text{ otherwise} \end{cases}$$

$$\mathrm{Adj}_3(E_n, j) = \begin{cases} \mathrm{flip}_1(E_n, (a,d)) \text{ and } \mathrm{flip}_1(E_n, (b,c)) \text{ if } (a,d) \text{ and } (b,c) \text{ are 2-flippable} \\ \emptyset \text{ otherwise} \end{cases}$$

## 5 Results

The Reverse Search algorithm for generating rooted MPG4 for a given $n (n \geq 6)$ is shown in figure 13. In our notation, `E^*` denotes the target $E_n^*$; `e_i` denotes edge $e_i$; `Adj_k` denotes the adjacency oracle $\mathrm{Adj}_k$; `flip_k` denotes the flip function $\mathrm{flip}_k$; and {} denotes $\emptyset$. Note that here, `LocalSearch` is modified to return a pair of integers instead of edge(s). It functions exactly the same as it was defined in section 3.

`LocalSearch` returns $(i,k)$ where $i$ stands for edge $e_i$ and $k$ indicates the kind of flip. When $k = 1$, $\mathrm{flip}_k(E, e_i)$ corresponds to the single diagonal flip on $e_i$. When $k = 2$ or 3, $\mathrm{flip}_k(E, e_i)$ corresponds to the double diagonal flip on the 2 pairs of opposite edges respectively in the quadrangle defined by $e_i$ in the triangulation.

We have implemented the algorithm in C using the same data structure adopted from the program in [1]. Complexity analysis is straightforward as in [1]: the algorithm uses $O(n)$ space and $O(n^2 \cdot f(n))$ time where $f(n)$ is the number of rooted MPG4 on $n$ points. It was tested on

```
ReverseSearch(n);

int match=0;  // flag to indicate if a neighbour down the reverse search
              // tree is found
E=E^*;  // start at target
output(E);
i=4;  // no flip on e_1, e_2 or e_3
k=1;  // adjacency oracle counter
depth=0;  // depth counter for the reverse search tree; target at depth 0
while (depth>-1)
     { while ((i<=(3n-6)) && (!match))
            { E´= Adj_k(E,i);
              if (E´!={}) { (i´,k´)=LocalSearch(E´,n);
                             if (flip_k´(E´,e_i´)==E) match=1;
                             else { if (k==3) {i++;k=1;}
                                    else k++;
                                  }
                          }
              else { if (k==3) {i++;k=1;}
                     else k++;
                   }
             }
       if (i<=(3n-6))  // traverse down the tree
         { flip_k(E,i); output(E);
           depth++; match=0;
           i=4; k=1;
         }
       else  // backtrack
         { depth--;
           if (depth>-1) // not target
             { (i,k)=Localsearch(E,n);
               flip_k(E,e_i);
               if (k==3){i++;k=1;}
               else k++;
             }
         }
     }
```

Figure 13: Pseudo code for Reverse Search for generating rooted MPG4

a DEC3000/500 Alpha for $6 \leq n \leq 17$. For $6 \leq n \leq 13$, the total number of rooted MPG4 were verified with the program in [1]. The largest problem solved with $n = 17$ and $f(n) = 63,336,873$ took 3461 minutes. We have also added a constraint to our program to produce rooted MPG5. These results and timings are shown in table 1.

# 6 Conclusion and Future Work

We have shown that the graph of rooted MPG4 (given $n$) is connected under single and double diagonal transformations. Hence we designed and implemented an efficient algorithm for generating non-isomorphic rooted MPG4 using reverse search. It is possible to parallelize our program for better run time and we may want to do this to obtain results for larger $n$. Hurtado and Noy [5] showed the graph of rooted triangulations is hamiltonian. It would be interesting to see if the same is true for the graph of rooted MPG4. Finally, we found that the graph of rooted MPG5 (given

$n$) is not connected under single and double diagonal transformations alone. However, it might be connected under single, double and some triple diagonal transformations (in fact the graph of rooted MPG5 for $n = 14$ verifies this) and this deserves further investigation.

## References


[1] D. Avis. Generating rooted triangulations without repetitions. *Algorithmica*, 16:618–632, 1996.

[2] D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 6:21–46, 1996.

[3] A. Deza, K. Fukuda, and V. Rosta. Wagner's theorem and combinatorial enumeration of 3-polytopes. Combinatorial Geometry B-271, Department of Information Sciences, Tokyo Institute of Technology, October 1993.

[4] S. Even. *Graph Algorithms*. Computer Science Press, 1979.

[5] F. Hurtado and M. Noy. The graph of triangulations of a convex polygon. Technical Report MA2-IR-94-13, Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, May 1994.

[6] O. Ore. *The Four-Color Problem*. Academic Press, 1967.