Generating Geometric Objects I: Basics

David Avis

McGill University

http://cgm.cs.mcgill.ca/ avis

Generating Geometric Objects

Input:

Points, lines, planes, halfspaces, spheres, ...

Output:

Vertices, intersection points, triangulation, Voronoi diagram, cells, faces, ...

Difficulties:

Output size may be huge, degeneracy, numerial errors

Typical Problems

Triangulation:

Generate all triangulations on a given point set

Euclidean Spanning Trees:

Generate all planar spanning trees on a given set of points

Laman frames:

Generate all Laman frames on a given point set

Arrangements:

Generate all the cells or vertices of an arrangement of lines, planes or hyperplanes

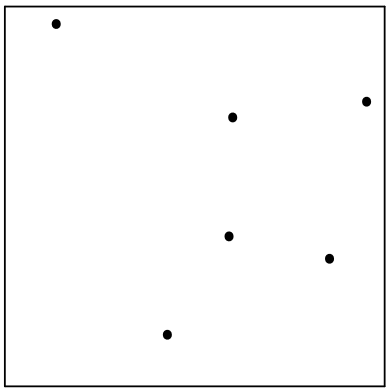
Convex Hull:

Generate the vertices or facets of a convex polyhedron

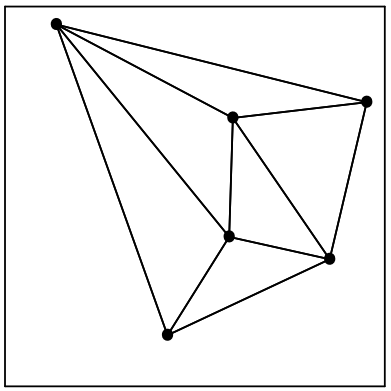
Problem: Triangulations of a point set

Input: A planar set P of n points

Output: The set of all triangulations of P.



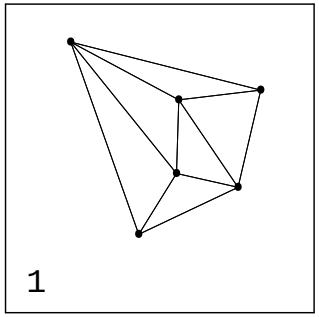
P: 6 points in Q^2

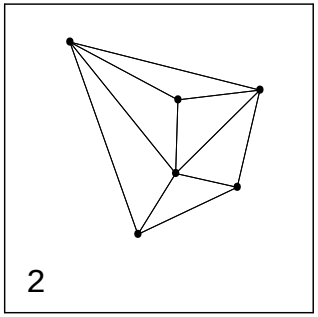


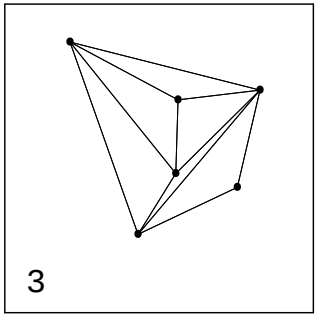
+ 6 Triangulations

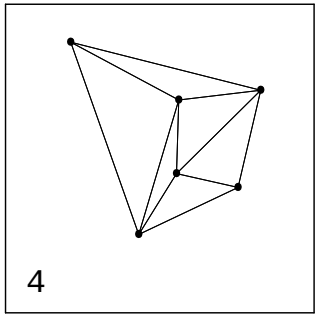
Delaunay Triangulation

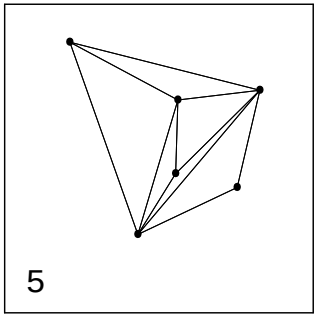
All triangulations on these 6 points:

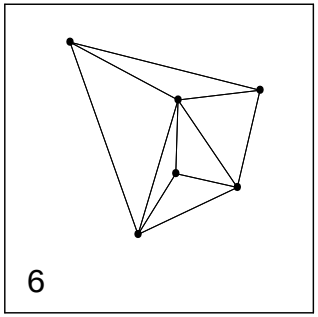


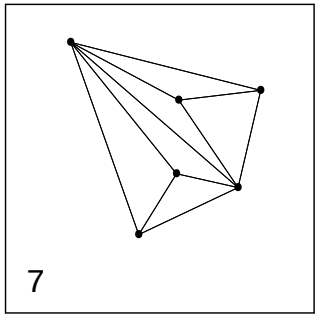




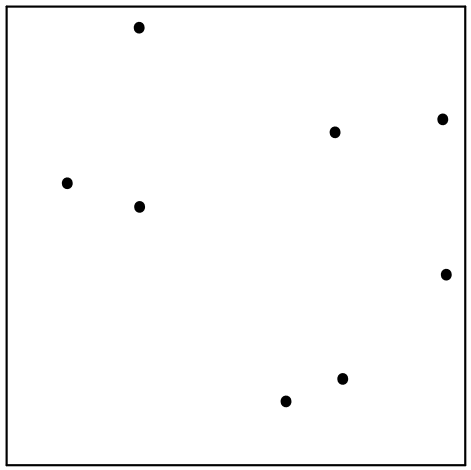




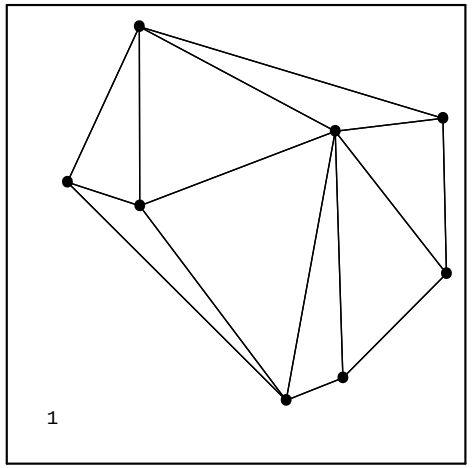




Instance with 8 points:



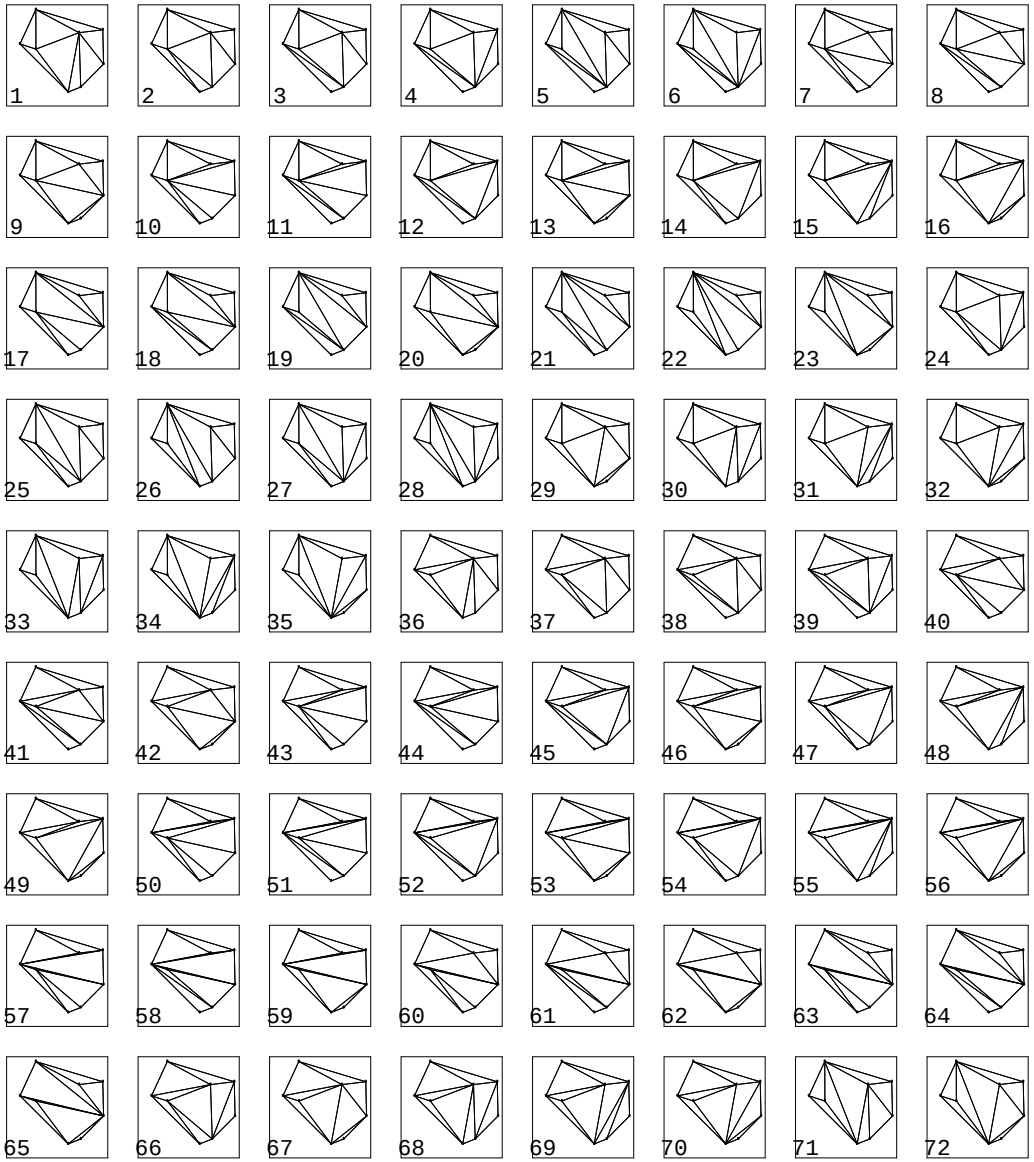
P: 8 points in Q^2



+ 71 Triangulations

Delaunay Triangulation

All triangulations on these 8 points:



Problem: Permutations of { 1,2,...,n }

Input:

Integer $n = 3$

Output:

1 2 3

1 3 2

2 3 1

2 1 3

3 1 2

3 2 1

Output size: n! = 6

Basic Idea: Graph Search

Define a connected graph $G = (V, E)$ where:

- V are the objects to be generated
- E are defined implicitly by an adjacency oracle

Informally, the oracle generates all objects similar to a given object.

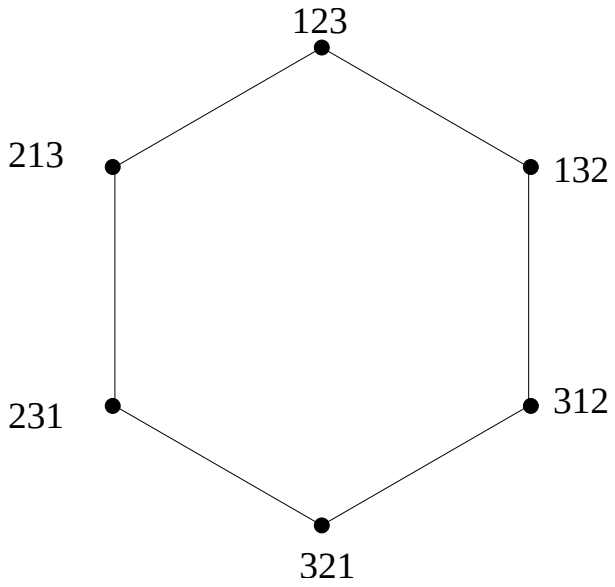
Let Δ denote the maximum degree of G.

Adjacency oracle: Permutations

Let $\pi = (\pi_1, \pi_2, ..., \pi_n)$ where $\{\pi_1, \pi_2, ..., \pi_n\} = \{1, 2, ..., n\}$

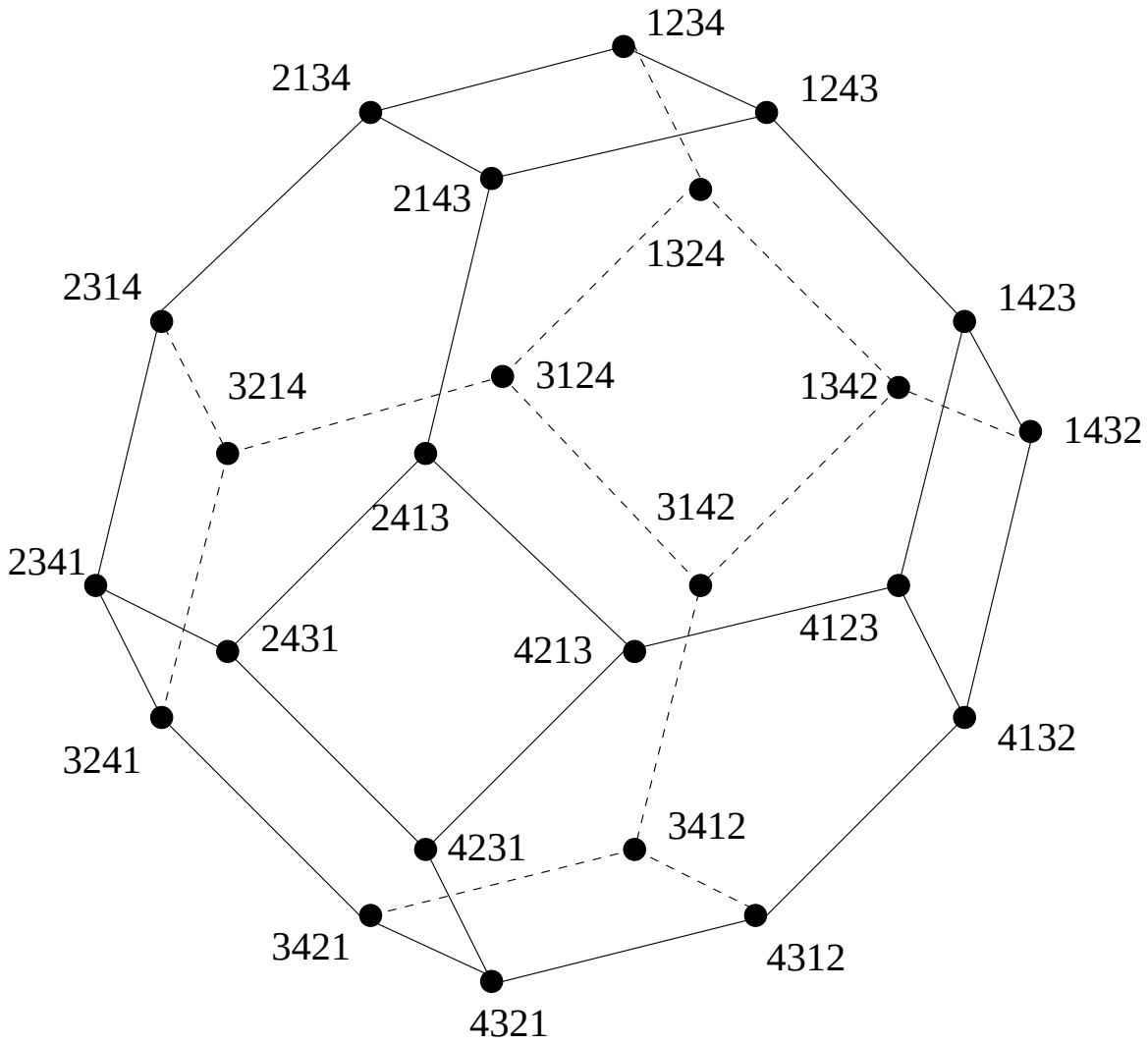
$$Adj(\pi, i) = (\pi_1, \pi_2, ..., \pi_{i-1}, \pi_{i+1}\pi_i, ..., \pi_n) \quad \text{for} \quad i = 1, 2, ..., n-1$$

$$\Delta = n - 1$$



Eg. $n = 3$

Permutahedron ($n = 4$)



ADJACENCY ORACLE

$G = (V, E)$ Δ = maximum degree

For every $x \in V$ $i = 1, 2, \ldots, \Delta$

$$Adj(x, i) = \begin{cases} y & \text{where } xy \in E \\ \phi & \text{otherwise} \end{cases}$$

Key Property:

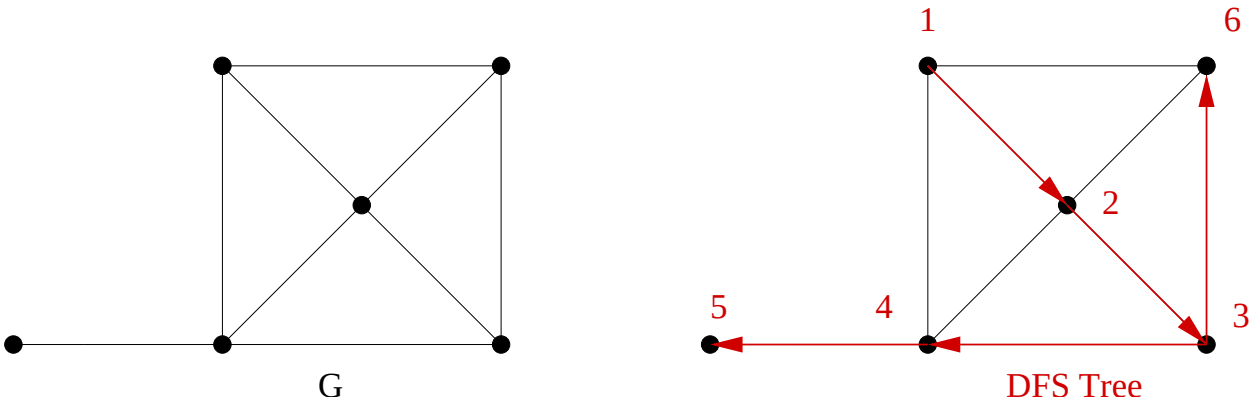
For every edge xy in G there is a unique i s.t.

$$y = Adj(x, i)$$

How to Search the Graph?

Depth First Search (DFS) explores a graph by:

- Labelling a given starting vertex
- Finding and labelling a vertex adjacent to last labelled vertex
- Backtracking when there are no more edges from a given vertex



Disadvantages:

- All labelled vertices need to be stored
- A stack is needed for backtracking

Space Efficient Graph Searching - I

Gray Code Method: (Wilf, Read, ...)

- Traverse a Hamiltonian cycle in the graph
- Can be initiated from any vertex
- *Next* function gives a unique successor to each vertex
- Useful for highly structured problems

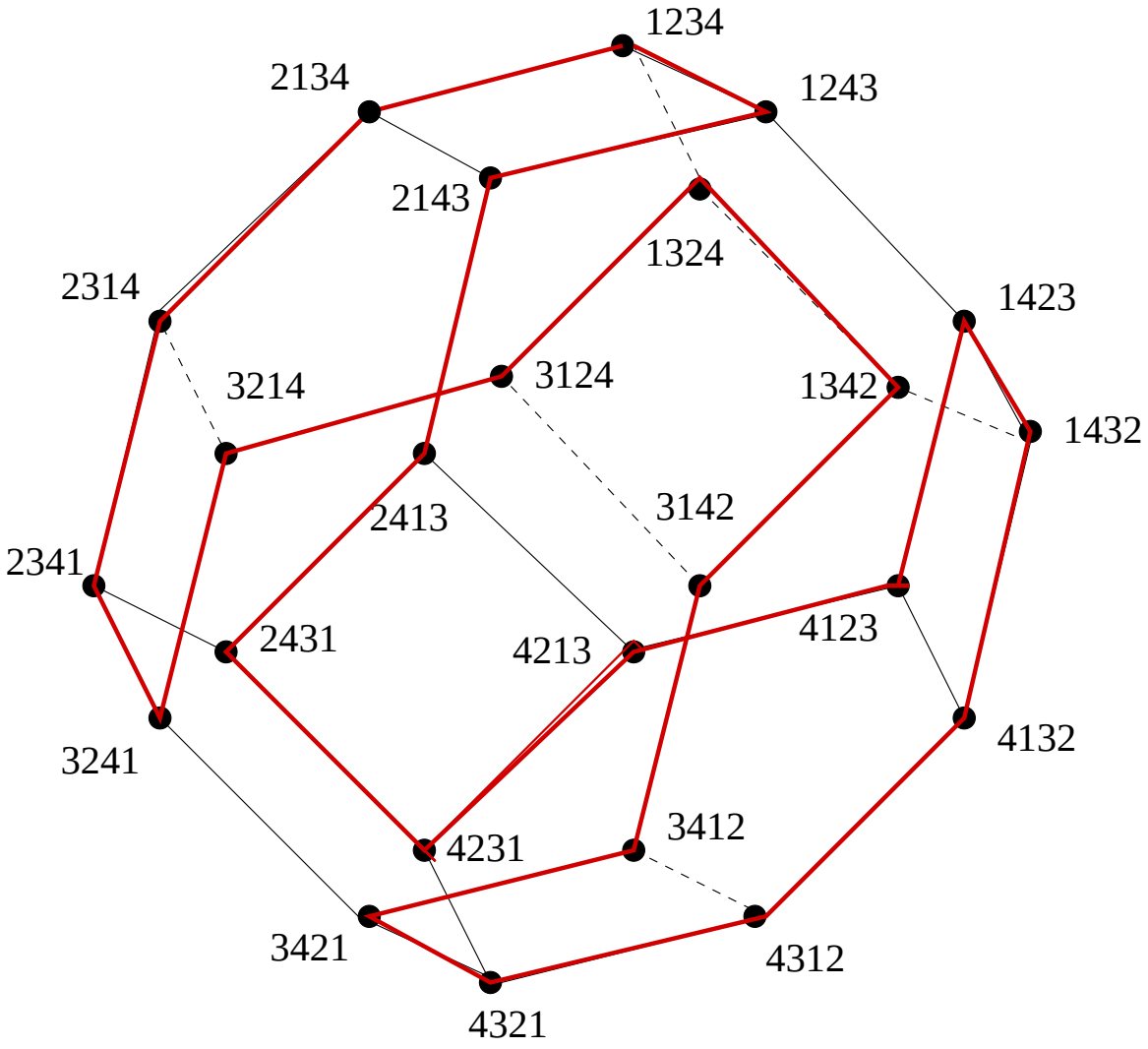
Disadvantages:

- Underlying graph may not be Hamiltonian
- Finding a cycle may be hard

Examples:

- subsets of a set, permutations, binary trees, ...

Hamilton Cycle on the Pemutahedron



Space Efficient Graph Searching - II

Reverse Search Method: (Avis-Fukuda(91), ...)

- Traverse a spanning tree in the graph
- Initiated from the root (target)
- Local search gives a unique parent for each non-root
- Useful for unstructured problems, especially geometric
- Little or no extra storage required

Disadvantages:

- Parent and/or Adjacency may be expensive to compute
- Wasteful if graph is dense

Examples:

- triangulations, arrangements, vertex enumeration, matchings, Euclidean trees, matroid bases, Laman frames, ...

LOCAL SEARCH

$G = (V, E)$ $\quad\quad$ $S \subseteq V$ set of targets

$f : V - S \rightarrow V$ local search function

Key Properties:

(a) $(v,\ f(v)) \in E$

(b) The following loop terminates for every $v \in V$:

while $v \notin S$
{ output $(v,\ f(v))$;
$v = f(v)$; }

Note:

The above code defines a path in G
from v to some vertex in S.

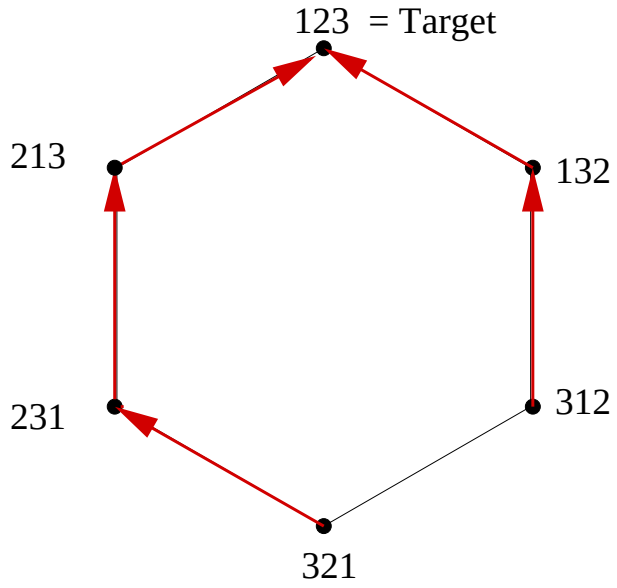
Local Search: Permutations

Let $\pi = (\pi_1, \pi_2, ..., \pi_n)$ Target: $(1, 2, ..., n)$

$$f(\pi) = (\pi_1, \pi_2, ..., \pi_{i-1}, \pi_{i+1}\pi_i, ..., \pi_n)$$

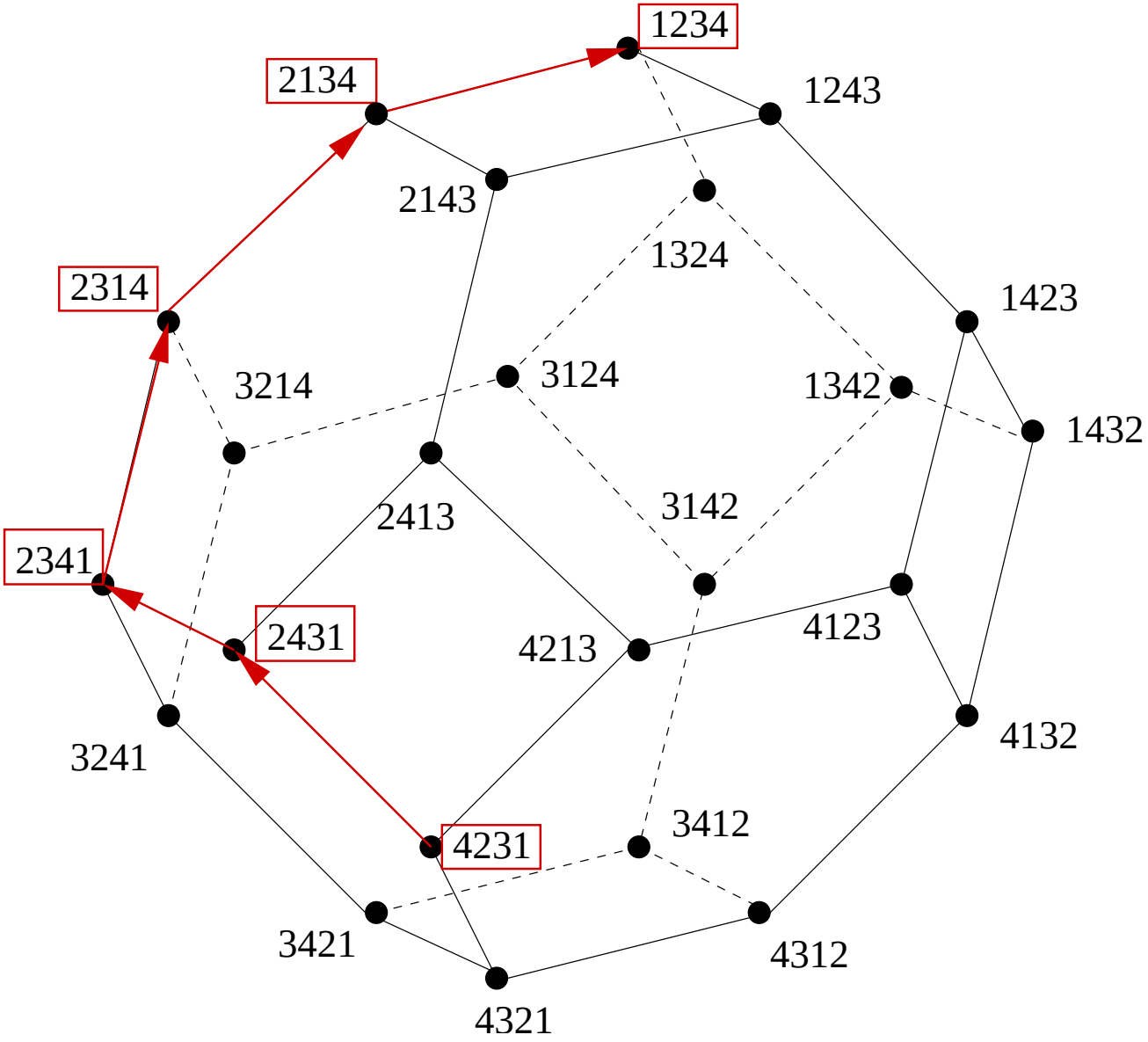
where i is the smallest index for which

$$\pi_i > \pi_{i+1}$$



Eg. $n = 3$

Local search: Permutations ($n = 4$)

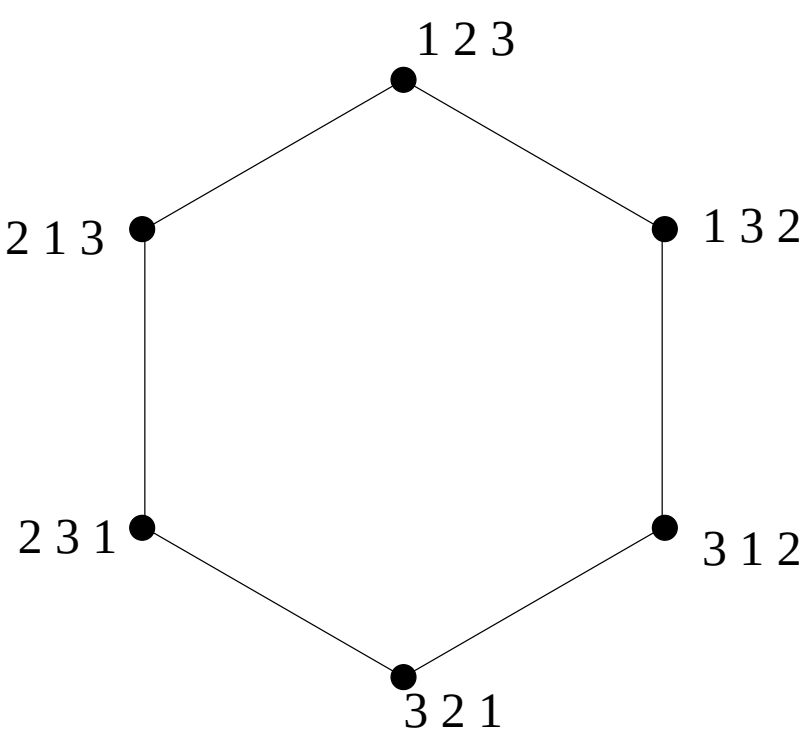


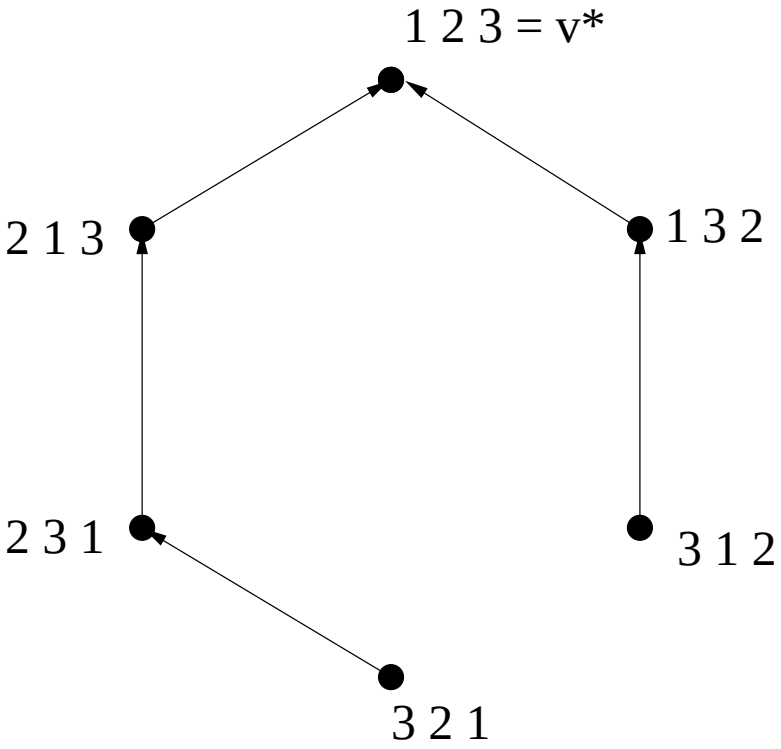
Reverse Search Tree

- The set of all local search paths forms the reverse search tree
- The tree is explored depth first from the root (target)
- For $i = 1, ..., \Delta$, $Adj(x, i)$ is a child of x iff $x = f(Adj(x, i))$
- Backtracking is performed by applying f
- No need to label vertices, as we only explore a tree

Adjacency and Reverse Search Tree for Permutations

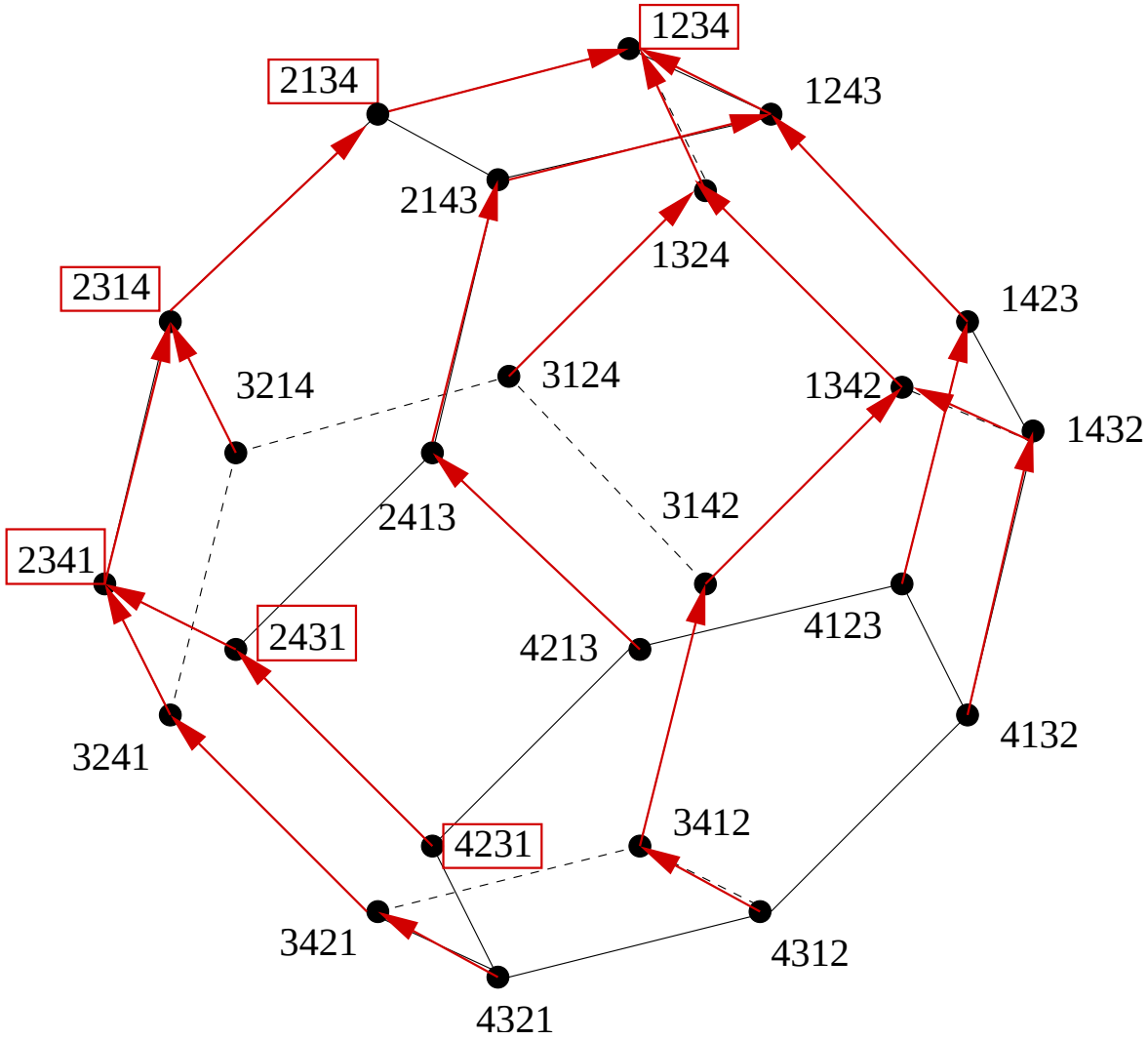
$(n = 3)$





Adjacency and Reverse Search Tree for Permutations

$(n = 4)$



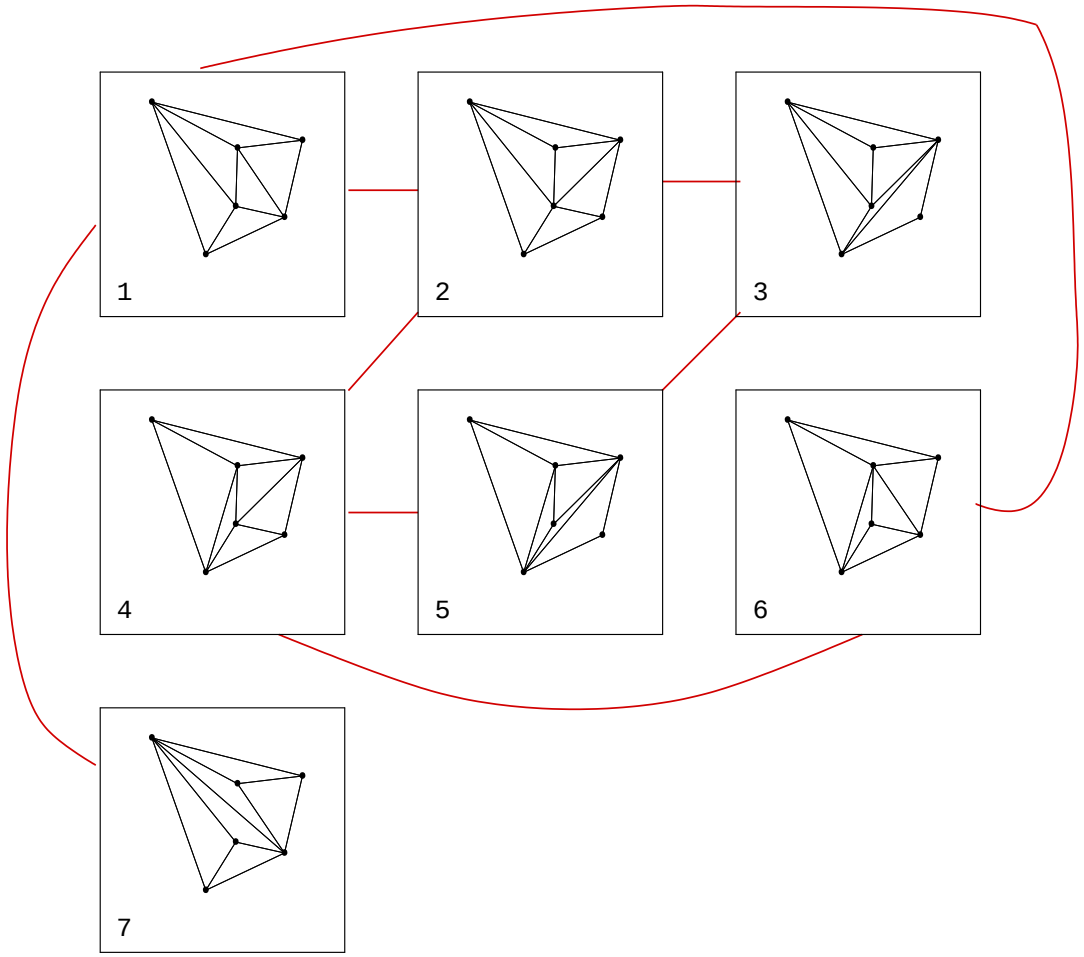
REVERSE SEARCH

```
for s∈S do                          // S is target set
    v = s;      j = 0;
    repeat
        while j < Δ do              // Δ is max. degree
        { j++;
          if ( f(Adj(v, j)) == v )
                {v = Adj(v, j);     // forward step
                  output v;
                  j = 0;}
        }                                   // end while

          if v ≠ s                  // backtrack step
                {u = v;   v = f(v);
                 find j s.t. Adj(v, j) = u}

    until v == s and j == Δ
```

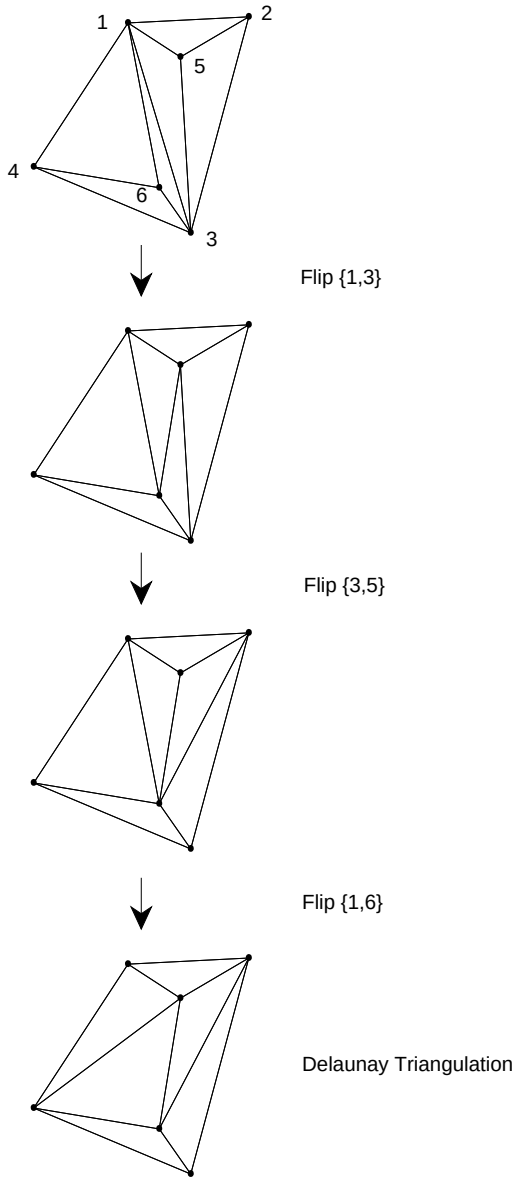
Problem: Generating all Triangulations of a point set

Adjacency:	Diagonal flip on convex quadrilateral
Local search:	Lex-min Delaunay edge flip
Target:	Delaunay triangulation

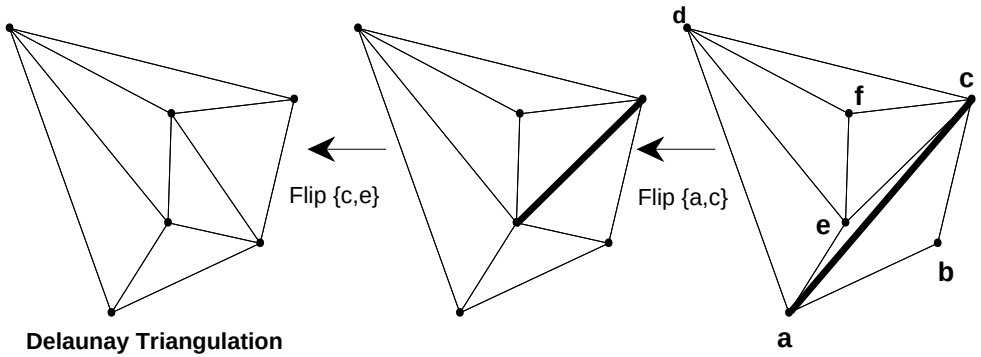
Triangulations: Adjacency by edge flip



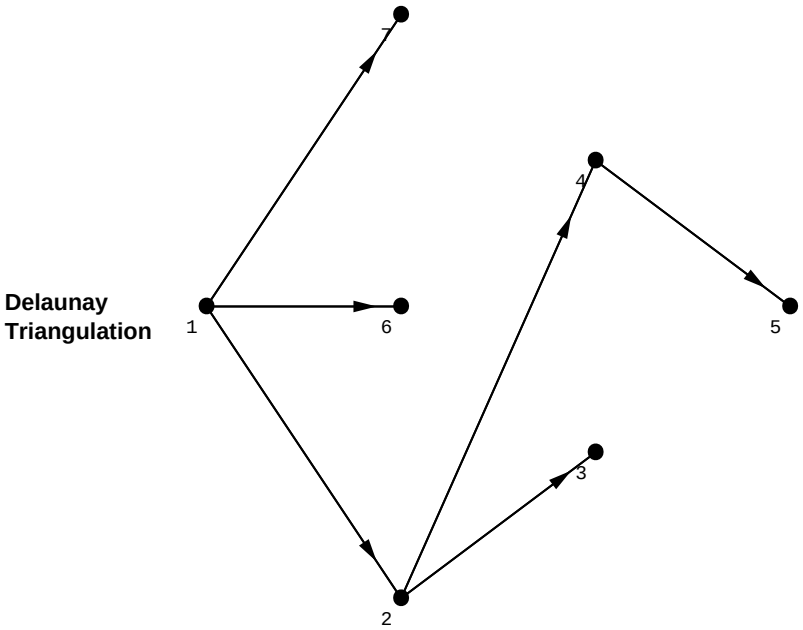
Triangulations: Local search



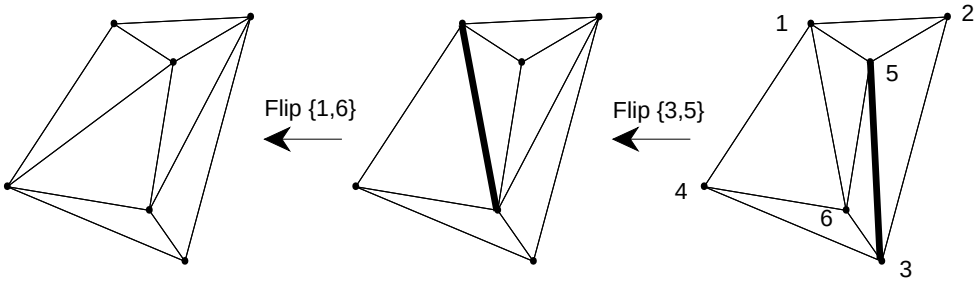
The Flip Algorithm



The Reverse-Flip Search Tree for the 6 Points Problem

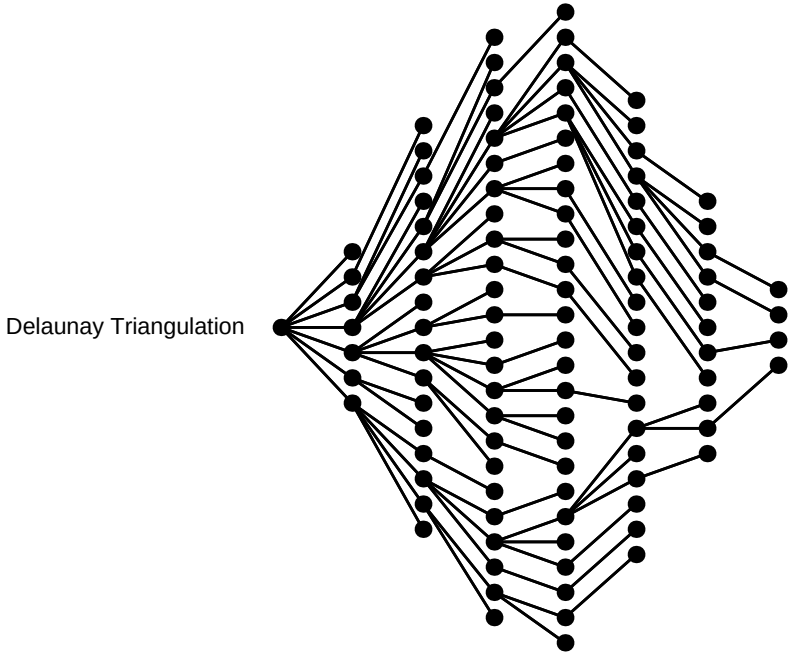


The Flip Algorithm



Delaunay Triangulation

The Reverse-Flip Search Tree for the 8 Points Problem



Delaunay Triangulation

Problem: Constrained Triangulations

Input: A planar set $P = \{1, 2, ..., n\}$

A set F of edges on P

Output: The set of all triangulations of P that contain F

Solution: Target: Constrained Delanay triangluation

Adjacency: Constrained edge flip

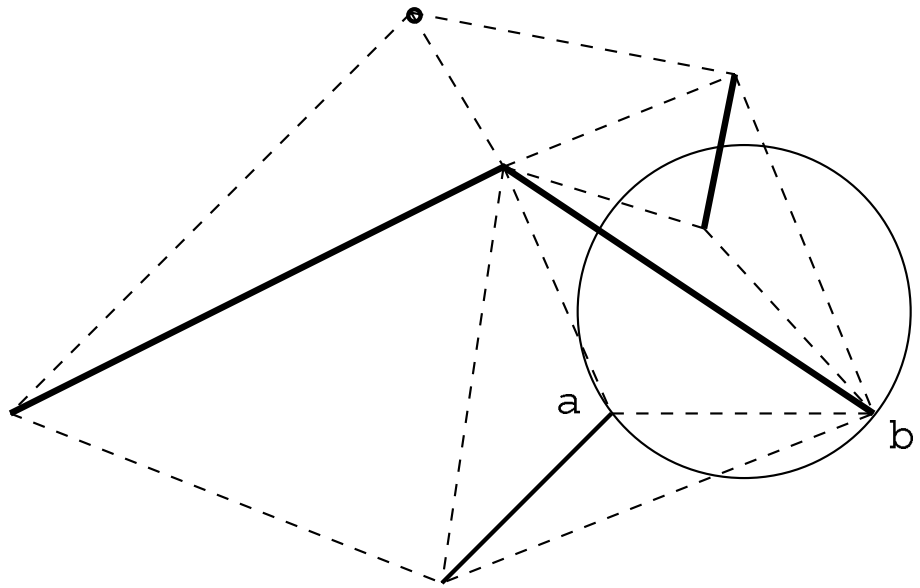


Figure 4. The CDT of a planar straight line graph.

Open Problems: Triangulations

Input: A planar set $P = \{1, 2, ..., n\}$

A set F of edges on P

Output: All triangulations of P that do not contain F

Input: A planar set $P = \{1, 2, ..., n\}$

Integer k

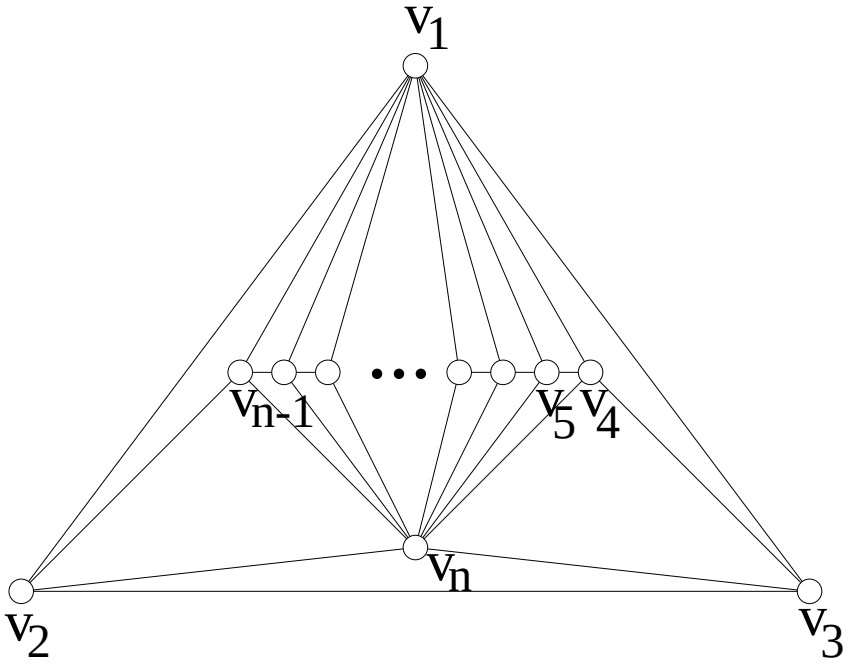
Output: All triangulations of P with maximum (or minimum) degree k

Problem: Max Planar Graphs with Min Degree k (MPG_k)

Input: Integers n, k

Output: All rooted MPG_k

Rooted: Outer face labels fixed



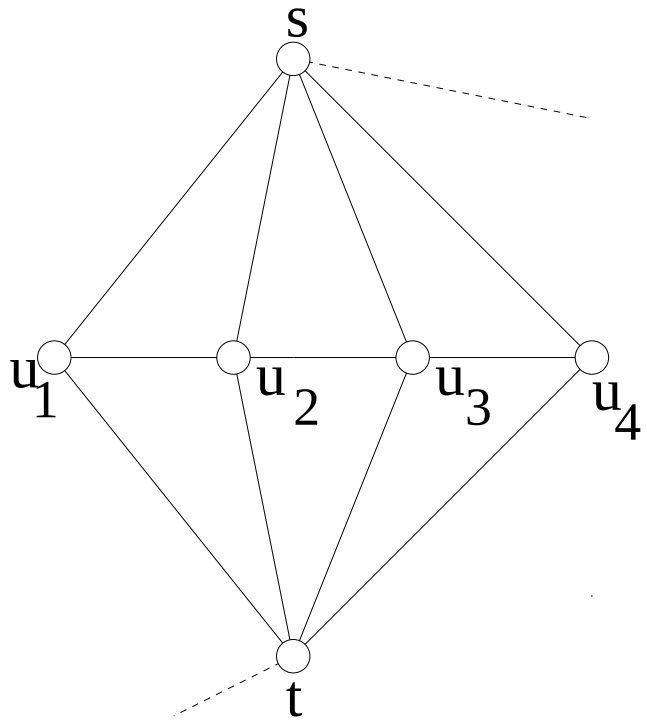
k=3 All $MPGs$
k=4 D.A. and M. Kong, 1996
k=5 Open

The Number of Maximal Planar Graphs

n	**#rooted MPG4**	**#rooted MPG5**	time taken/sec.	**#rooted MPG**	time taken/sec.
4	-	-	-	1	0
5	-	-	-	3	0
6	1	-	0	13	0
7	3	-	0	68	0
8	12	-	0	399	0
9	59	-	0	2,530	0
10	313	-	0	16,965	0.4
11	1,713	-	0.2	118,668	3.8
12	9,559	1	1.3	857,956	33.0
13	54,189	0	9.2	6,369,883	291.1
14	311,460	6	63.8	?	?
15	1,812,281	13	444.6	?	?
16	10,661,303	55	3042.4	?	?
17	63,336,873	189	20766.9	?	?

Double Flips

$flip(s, u_2)$ leaves $d(u_2) = 3$



After $flip(s, u_2)$ **and** $flip(t, u_3)$ all vertices have min deg 4.

Adjacency Oracle

E is an MPG_4, and edge $e_j = (a, b)$ bounds $\triangle abc$, $\triangle abd$.

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

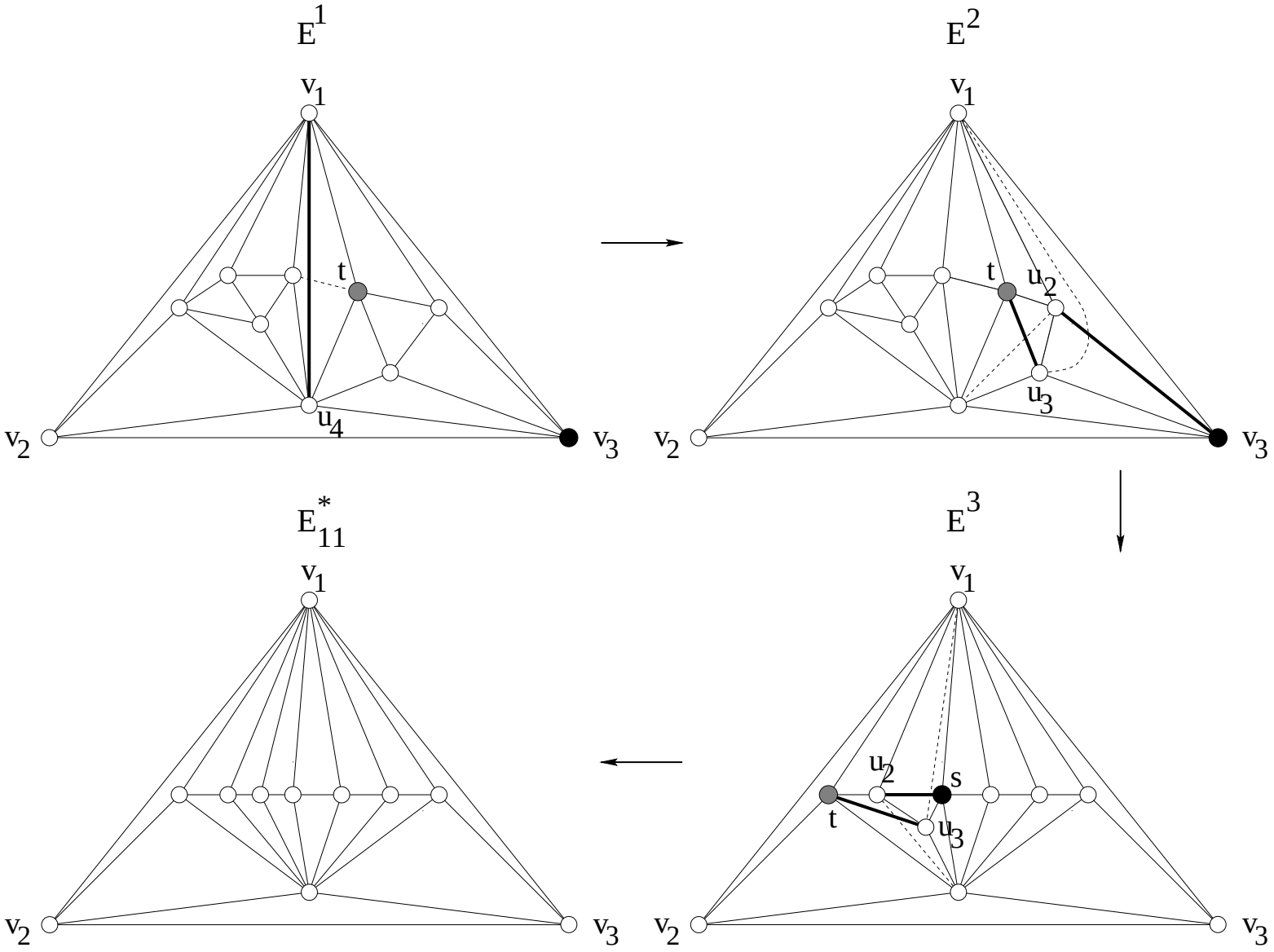
$$\mathrm{Adj}_2(E, j) = \begin{cases} flip(a, c)) \ \& \ flip(b, d)) \text{ if 2-flippable} \\ \emptyset \text{ otherwise} \end{cases}$$

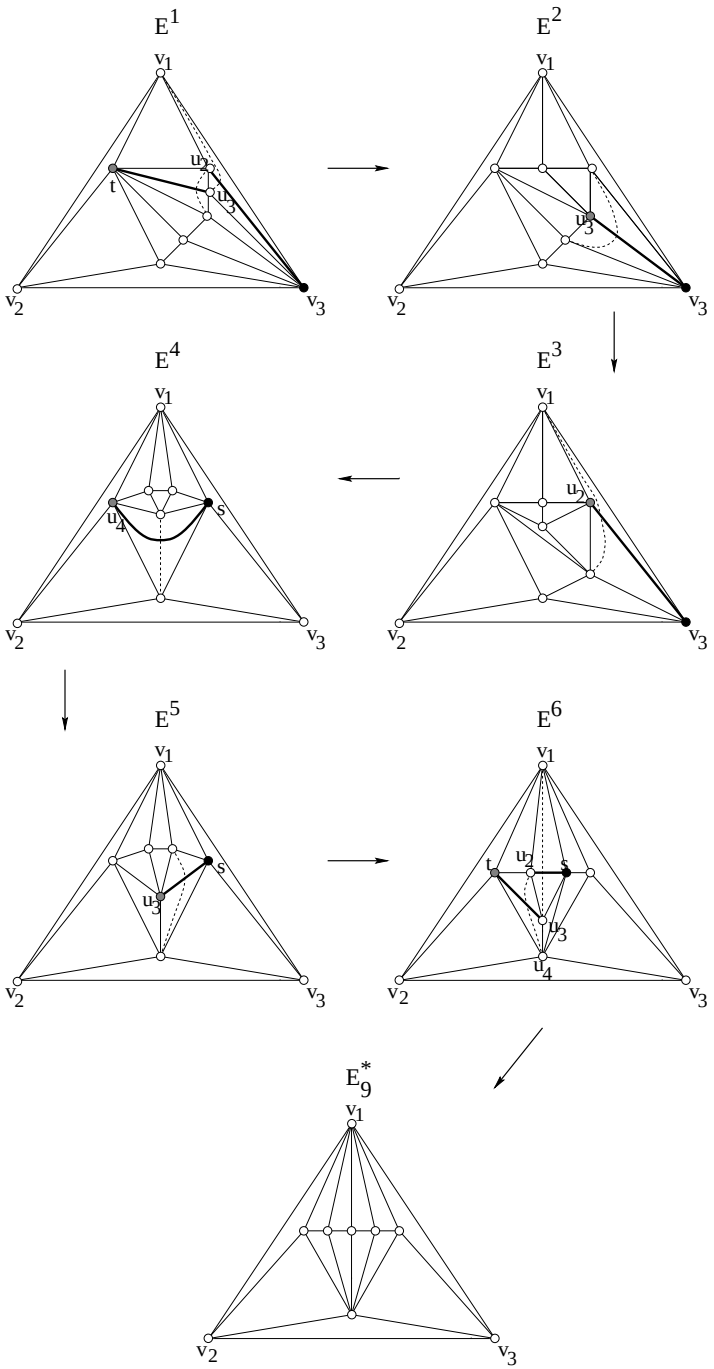
$$\mathrm{Adj}_3(E, j) = \begin{cases} flip(a, d)) \ \& \ flip(b, c)) \text{ if 2-flippable} \\ \emptyset \text{ otherwise} \end{cases}$$

Adjacency Lemma:

The graph defined by Adj_1, Adj_2, Adj_3 is connected.

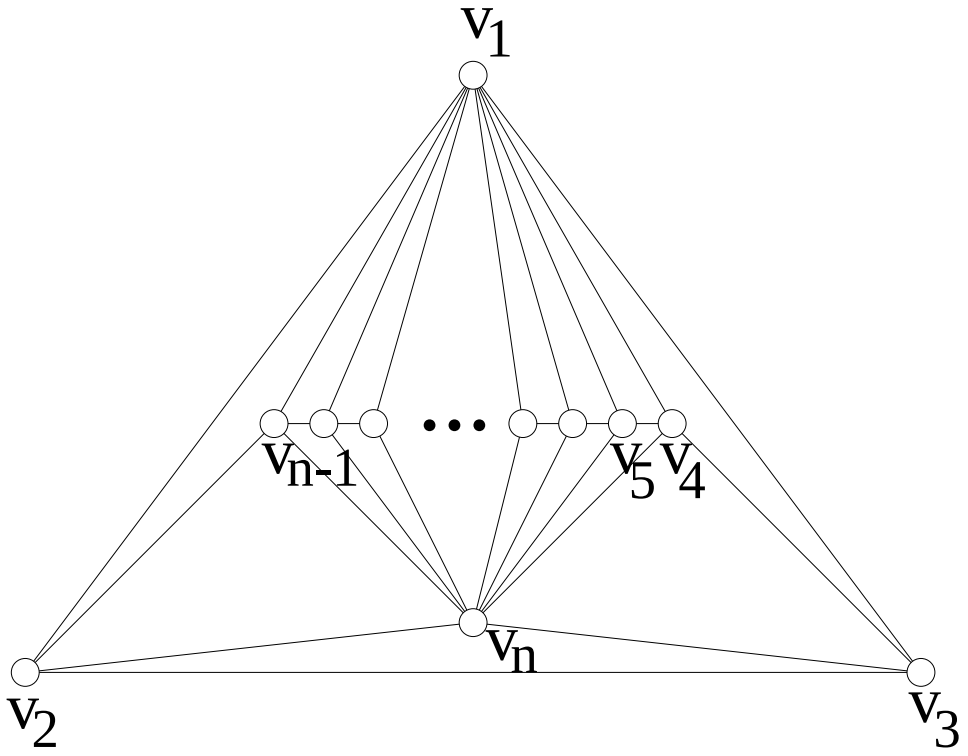
Local Search Function (dotted edge: added; bold edge: deleted)

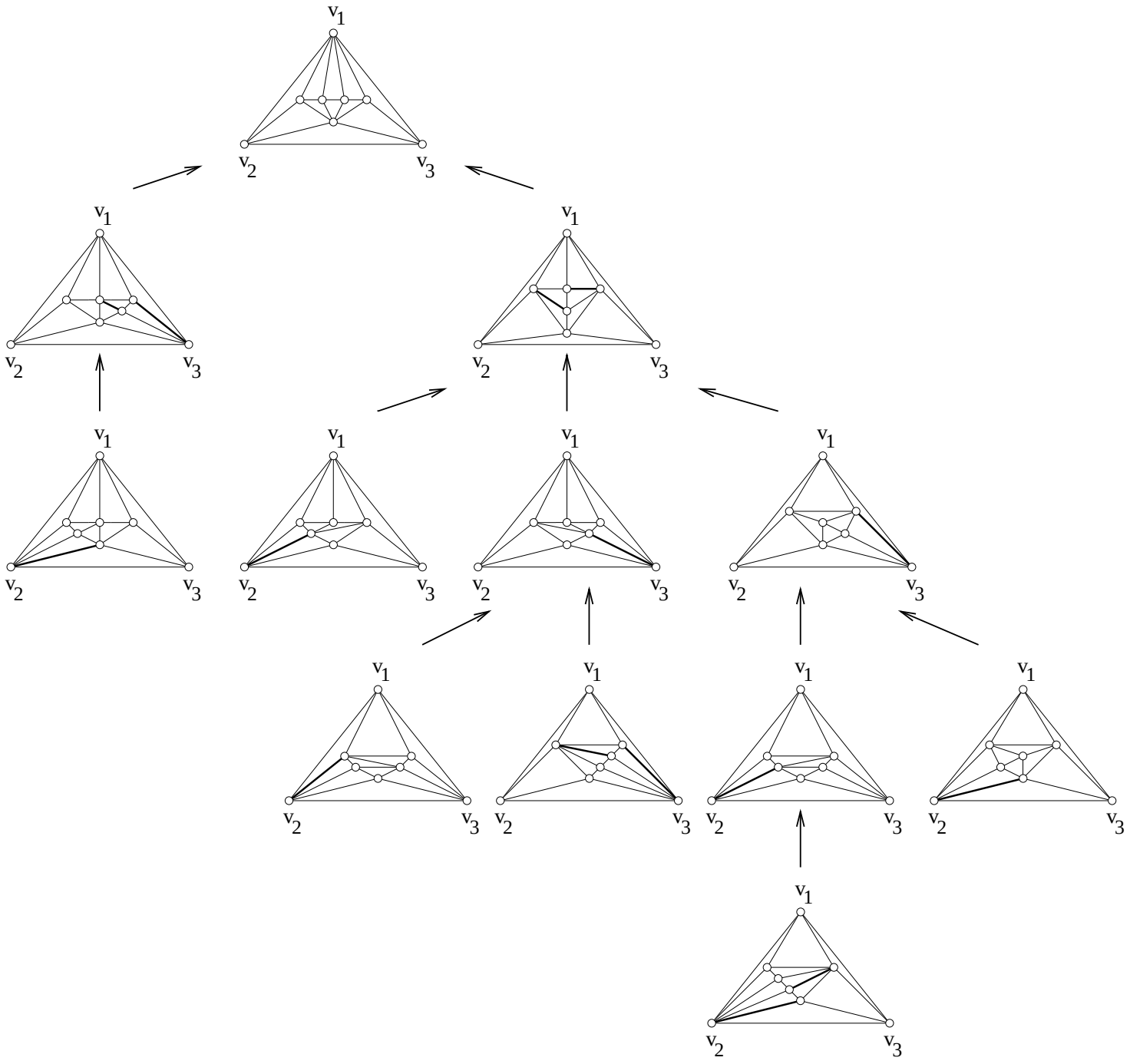



E1
v1
u2
t
u3
v2
v3
E2
v1
u3
v2
v3
E3
v1
u2
v2
v3
E4
v1
u4
s
v2
v3
E5
v1
s
u5
v2
v3
E6
v1
t
u2
s
u3
u4
v2
v3
E*9
v1
v2
v3

Local Search Lemma

- Edges returned by `LocalSearch` are either 1-flippable or 2-flippable.
- Successive application of `LocalSearch` transforms any rooted MPG_4 to the target E^*.




v1
v2
v3