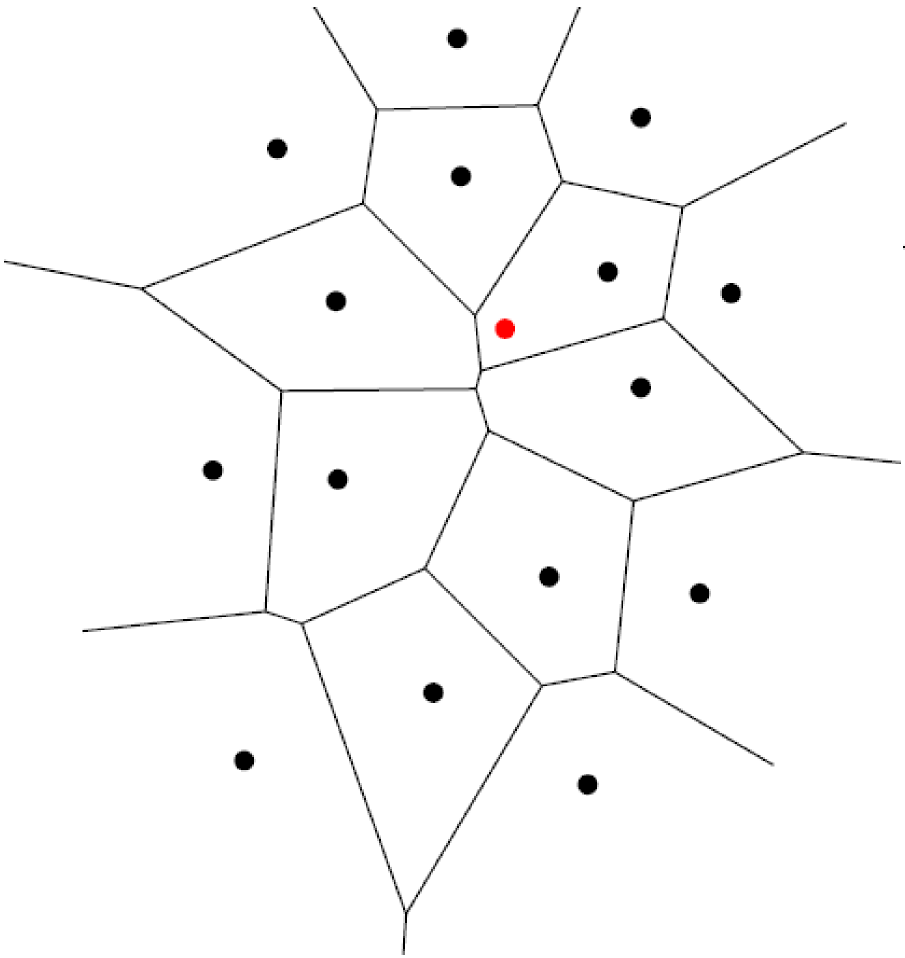
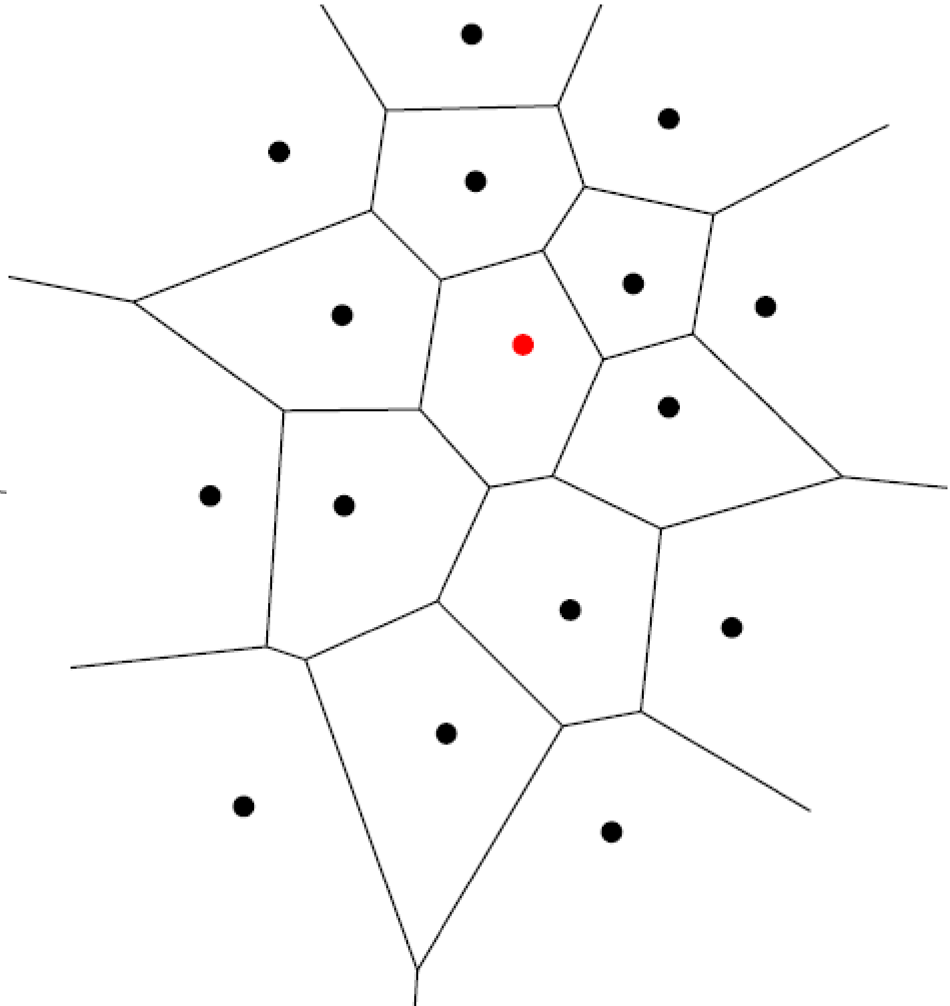
Algorithms for Voronoi Construction

Incremental Algorithm

- Initial diagram with the point and the VD after the algorithm





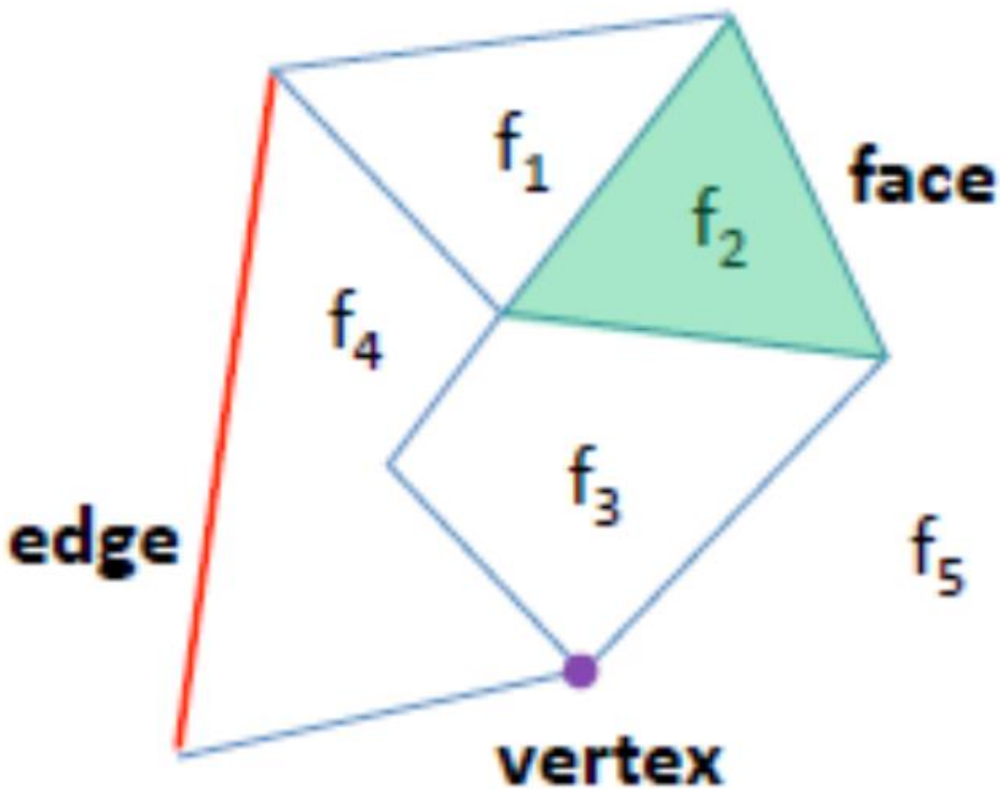
Incremental Algorithm –(contd.)

- To build the Voronoi polygon/ region of p_{i+1} , we use a data structure called Doubly Connected Edge List (DCEL)
- DCEL is proposed by Muller and Preparata
- DCEL is also known as half edge data structure

DCEL

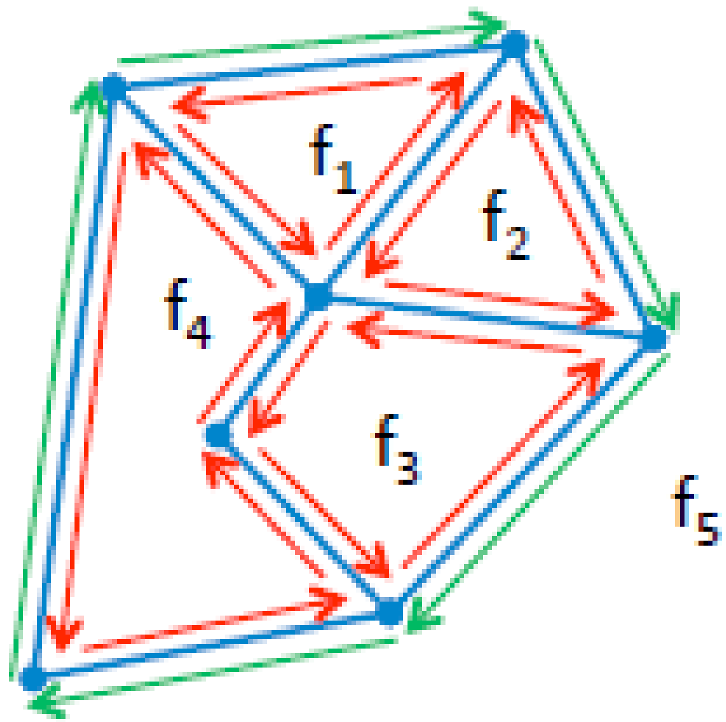
- DCEL is one of the most commonly used representations for planar subdivisions such as Voronoi diagrams.
- It is an edge-based structure which links together three sets of records:
  - **Vertex**
  - **Edge**
  - **Face**
- It facilitates traversing the faces of planar subdivision, visiting all the edges around a given vertex

DCEL



- **Record for each face, edge and vertex**

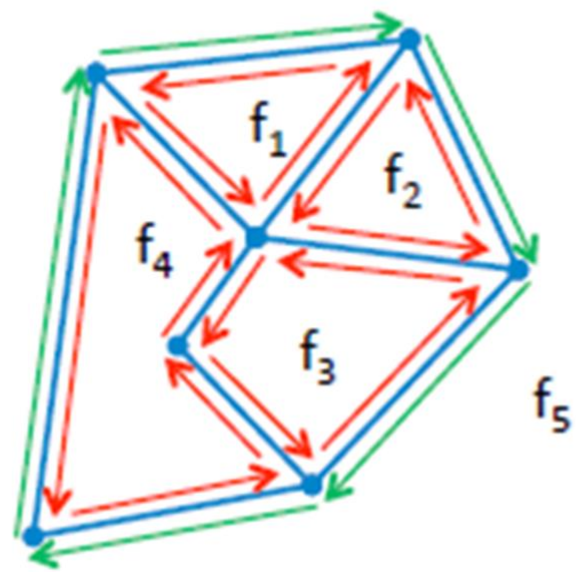
DCEL



- Edges are oriented counterclockwise inside each face
- Since each edge is shared by two faces, each edge is replaced by two half edges, one for each face

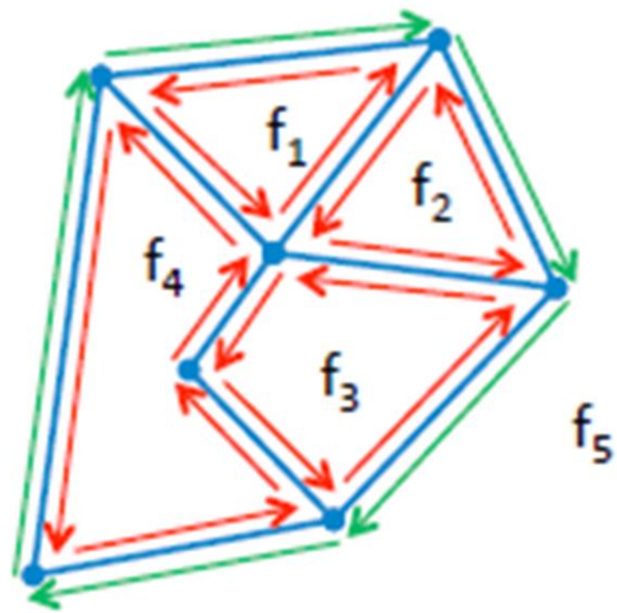
Vertex record

- **The vertex record of a vertex v stores:**
- Coordinates of v
- A pointer IncidentEdge(v)
  - To an arbitrary half edge that has v as its origin



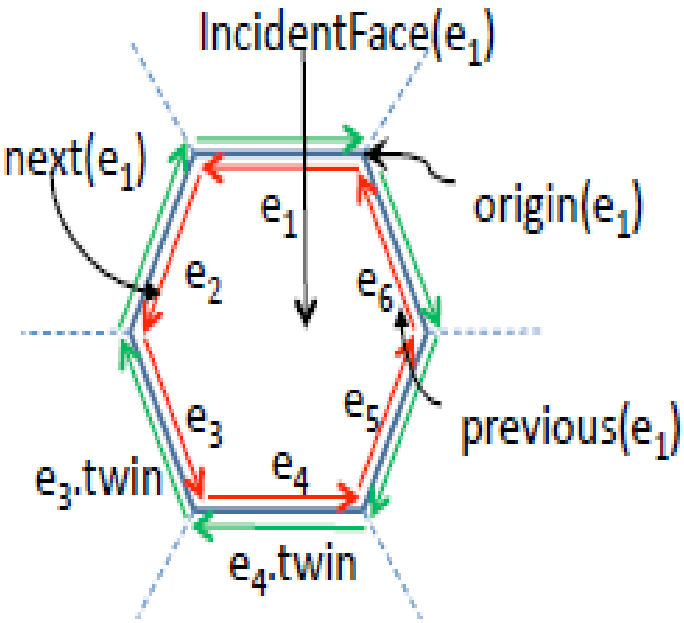
Face record

- **Face record of a face f stores:**
- A pointer to some half edge on its boundary
  - Which can be used as a starting point to traverse f in a counterclockwise order

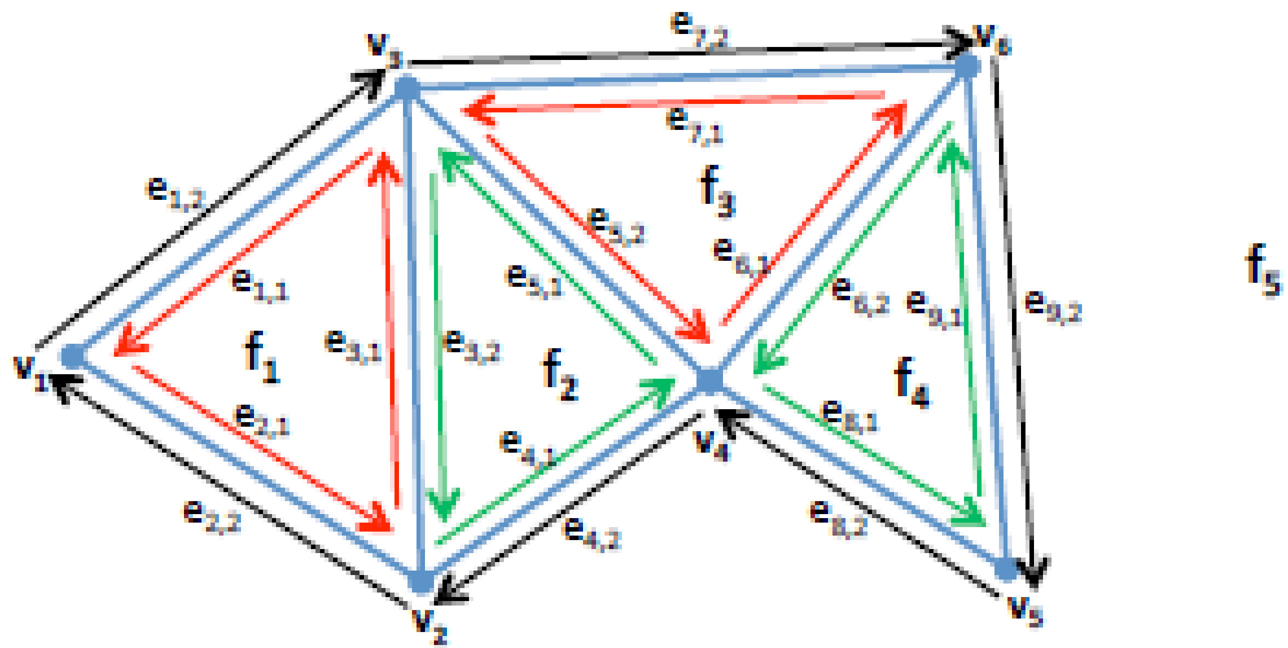


Half-Edge Record

- **The half-edge record of a half-edge e stores pointer to** :
- Origin(e)
- Twin of e, e.twin or twin(e)
- The face to its left, IncidentFace(e)
- Next half edge on the boundary of IncidentFace(e), Next(e)
- Previous half-edge, Previous(e)

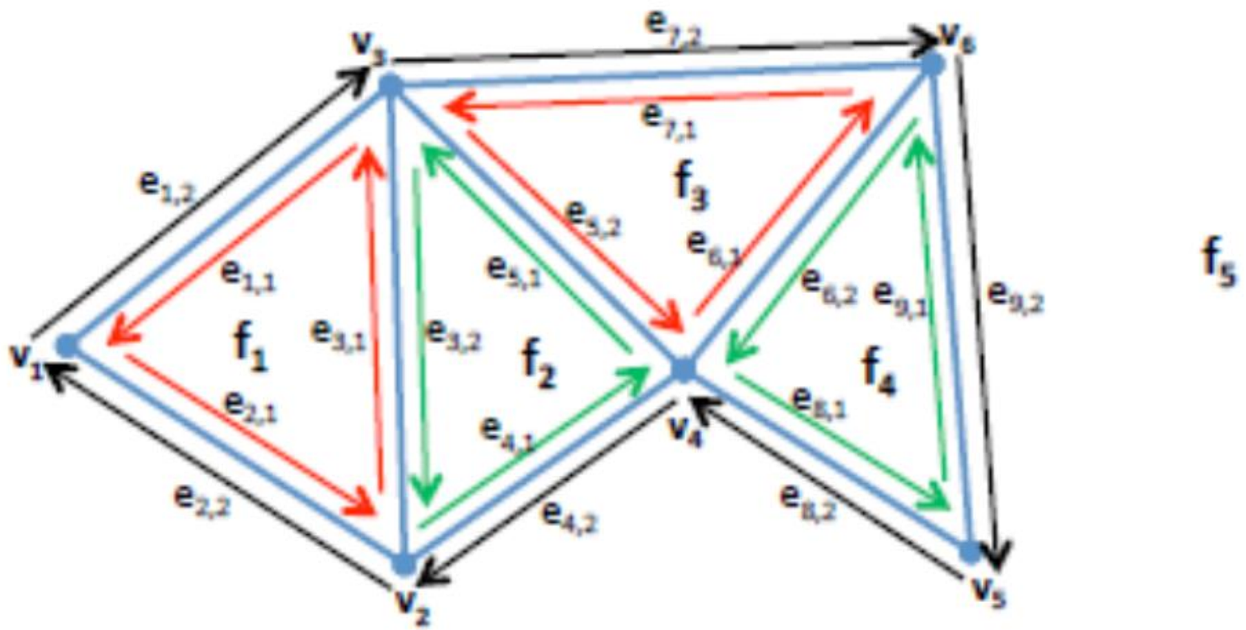


DCEL



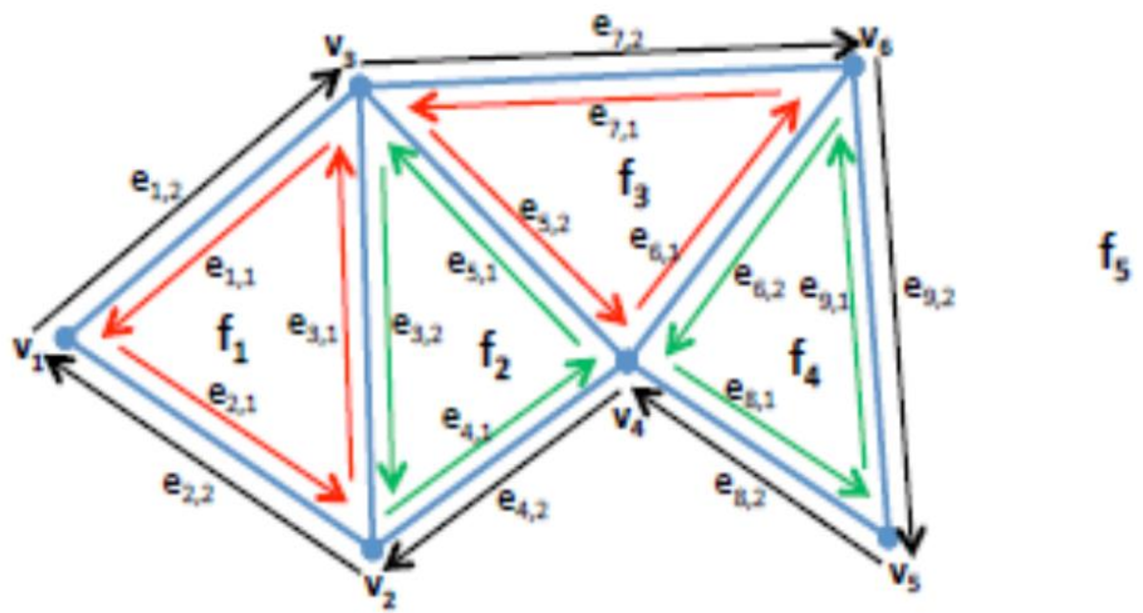
Vertex	Coordinates	IncidentEdge
v_1	(x_1, y_1)	$e_{2,1}$
v_2	(x_2, y_2)	$e_{4,1}$
v_3	(x_3, y_3)	$e_{3,2}$
v_4	(x_4, y_4)	$e_{6,1}$
v_5	(x_5, y_5)	$e_{9,1}$
v_6	(x_6, y_6)	$e_{7,1}$

DCEL



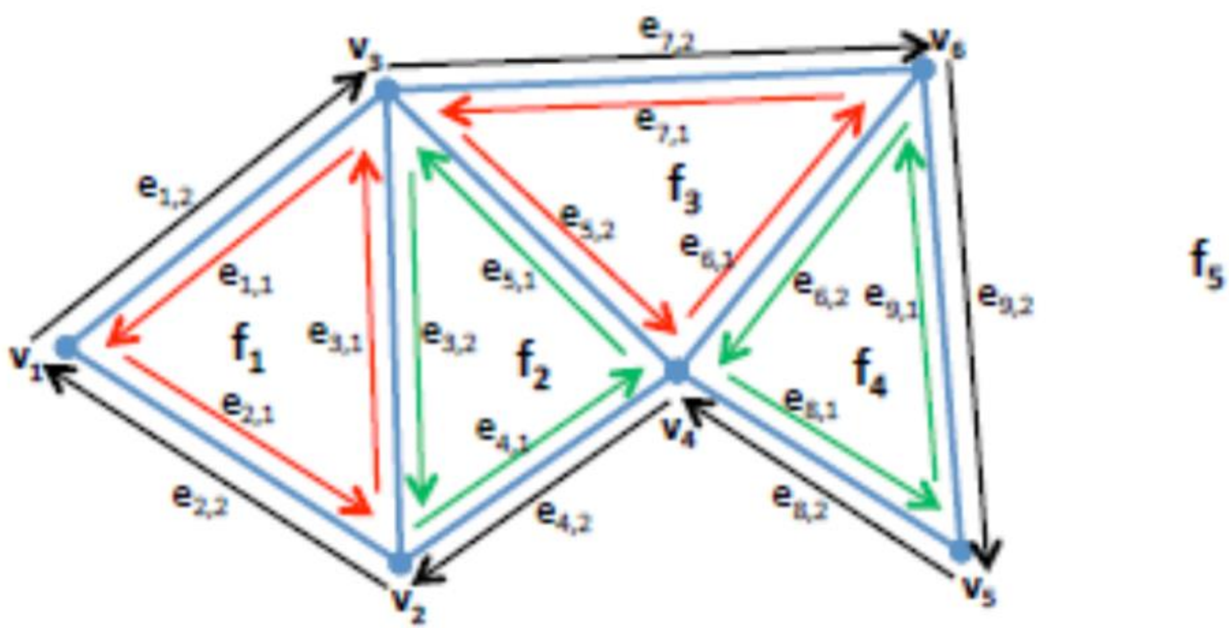
Face	Edge
f_1	$e_{1,1}$
f_2	$e_{5,1}$
f_3	$e_{5,2}$
f_4	$e_{8,1}$
f_5	$e_{9,2}$

Storage space requirement



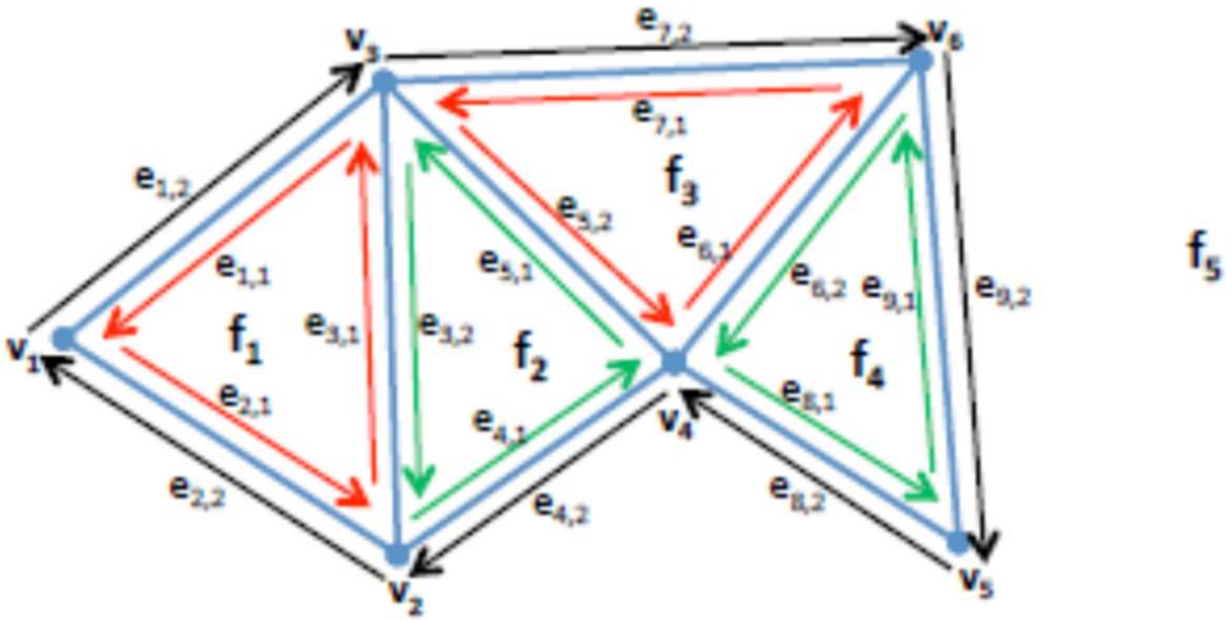
Half-edge	Origin	Twin	IncidentFace	Next	Previous
$e_{3,1}$	v_2	$e_{3,2}$	f_1	$e_{1,1}$	$e_{2,1}$
$e_{3,2}$	v_3	$e_{3,1}$	f_2	$e_{4,1}$	$e_{5,1}$
$e_{4,1}$	v_2	$e_{4,2}$	f_2	$e_{5,1}$	$e_{3,2}$
$e_{4,2}$	v_4	$e_{4,1}$	f_5	$e_{2,2}$	$e_{8,2}$
…	…	…	…	…	…

Storage space requirement



- Linear in the number of vertices, faces and edges

Operations on DCEL

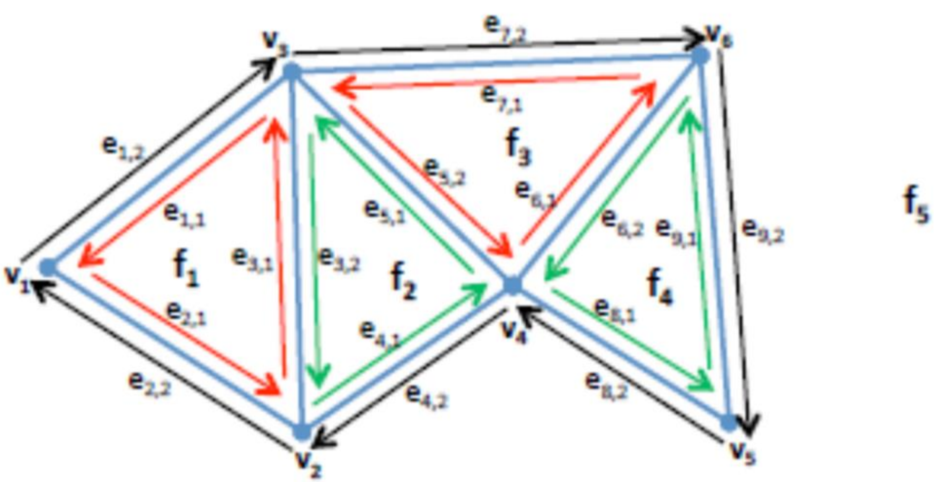


- Walk around a given face in CCW order
- Access a face from an adjacent one
- Visit all the edges around a given vertex

Traversing a face f

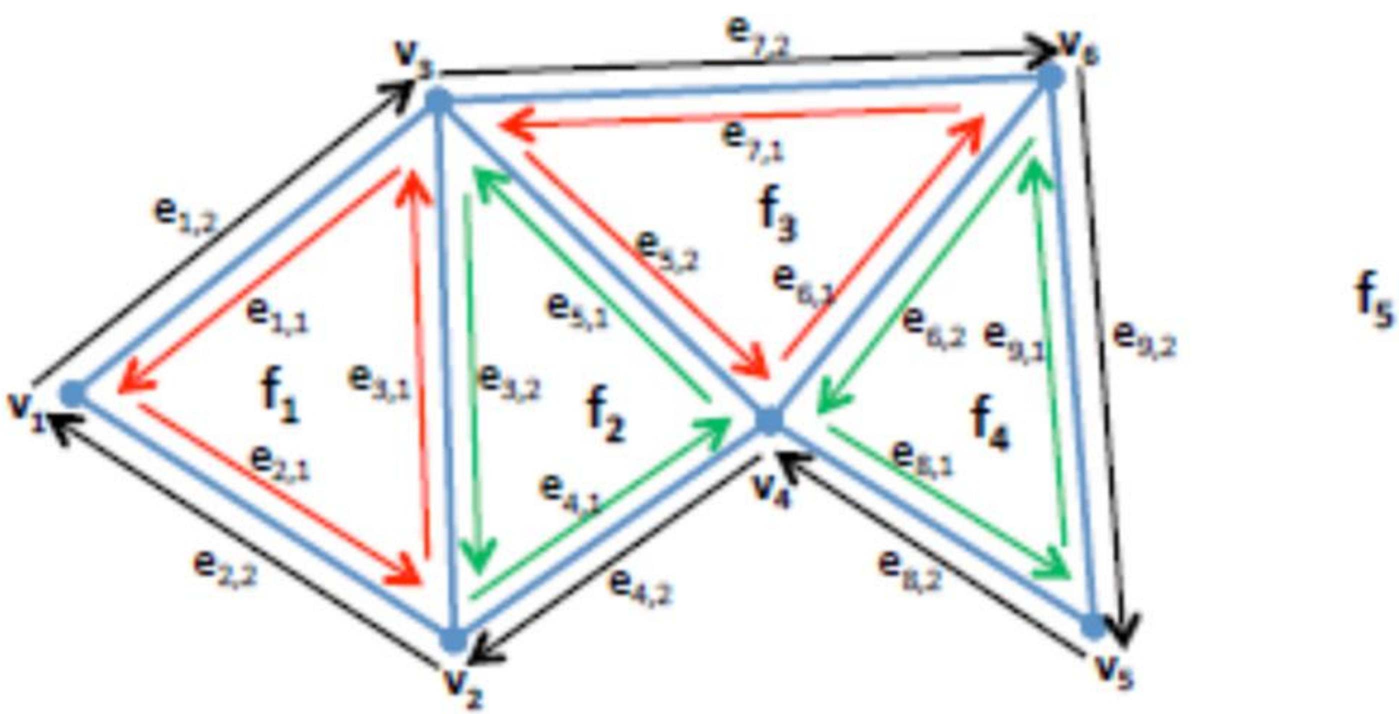
- Given an edge of f
- Determine the half-edge e incident on f
- Start_edge = e
- While next(e) ≠ Start_edge then

  e=next(e)

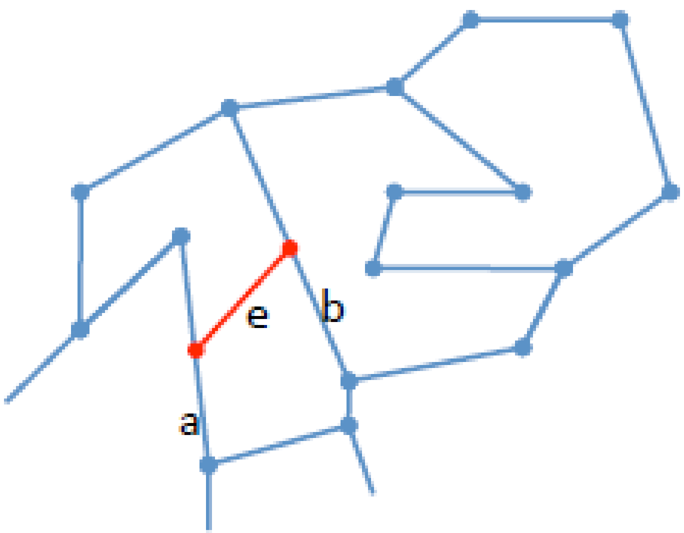


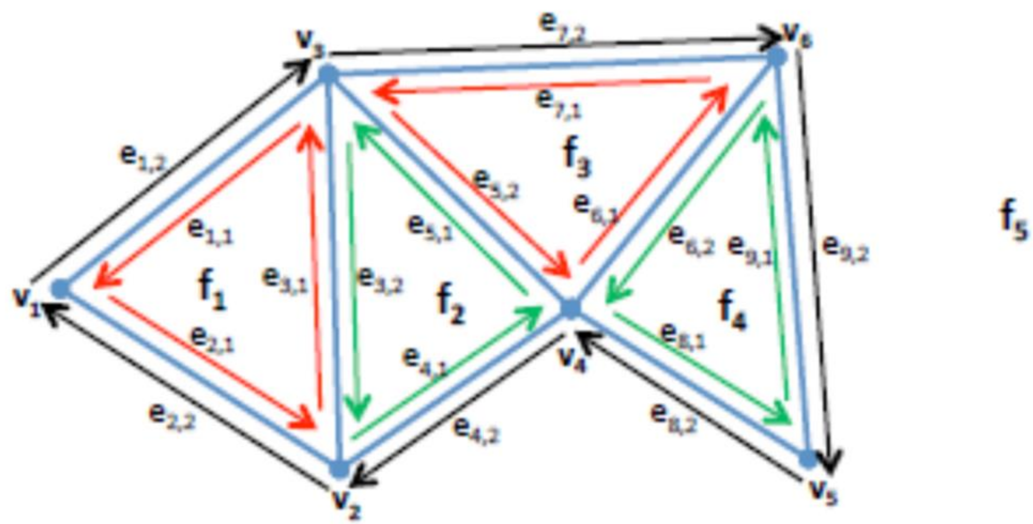
Some other operations on DCEL

- Traversing all edges incident on a vertex v
- Adding a vertex



Add an edge e





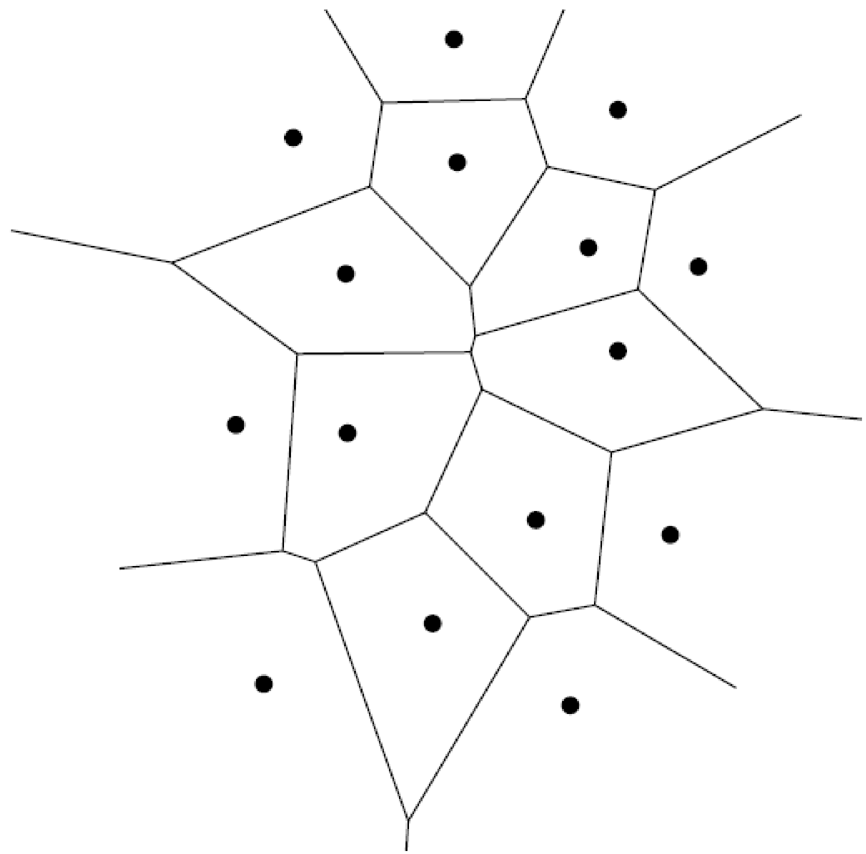
- DCEL can be updated in constant time once the edges a and b are known

Recall

Incremental Algorithm

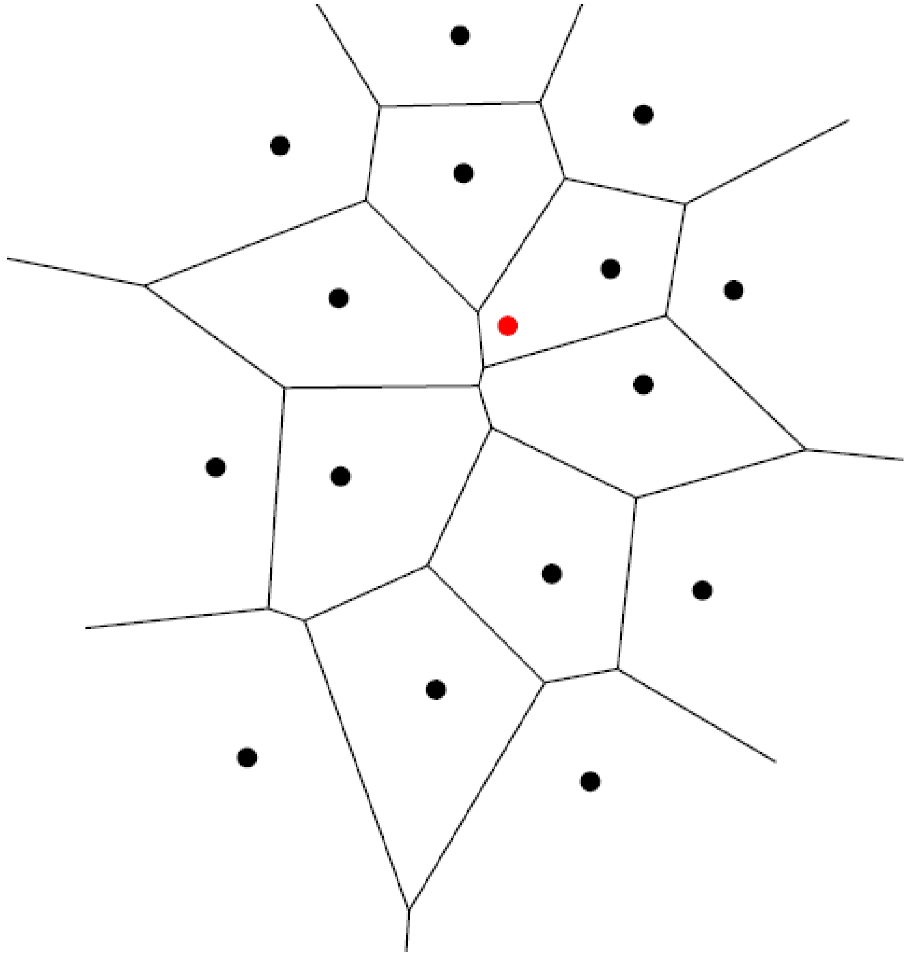
Incremental Algorithm

- Starts with a Voronoi diagram of $\{p_1, p_2, p_3, \ldots, p_i\}$



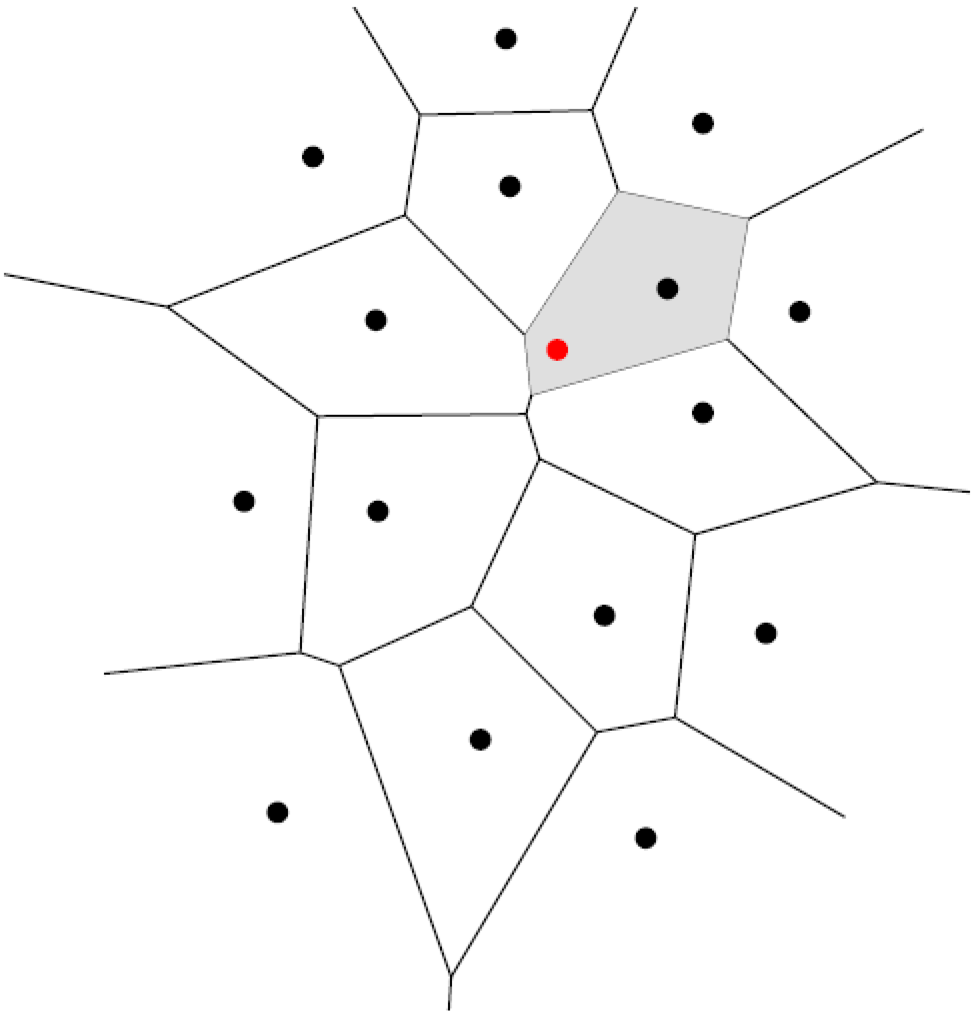
Incremental Algorithm –(contd.)

- Add a point p_{i+1}



Incremental Algorithm –(contd.)

- Explore all possibilities to find the point p_j closest to p_{i+1}

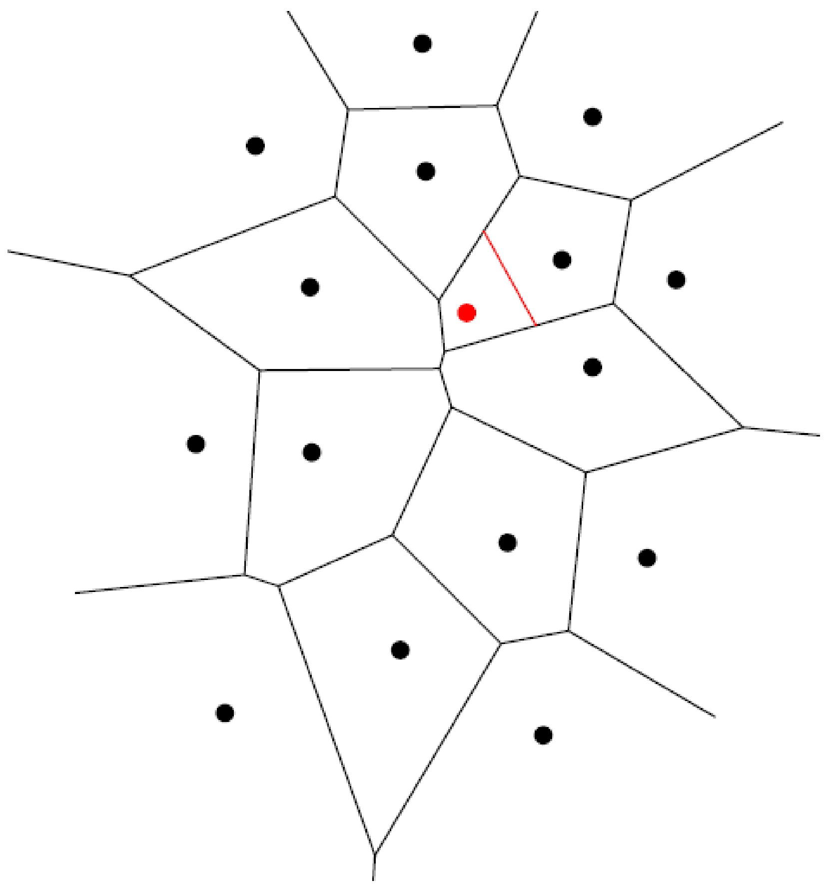


Incremental Algorithm –(contd.)

- Compute p_{i+1}'s Voronoi polygon/ region

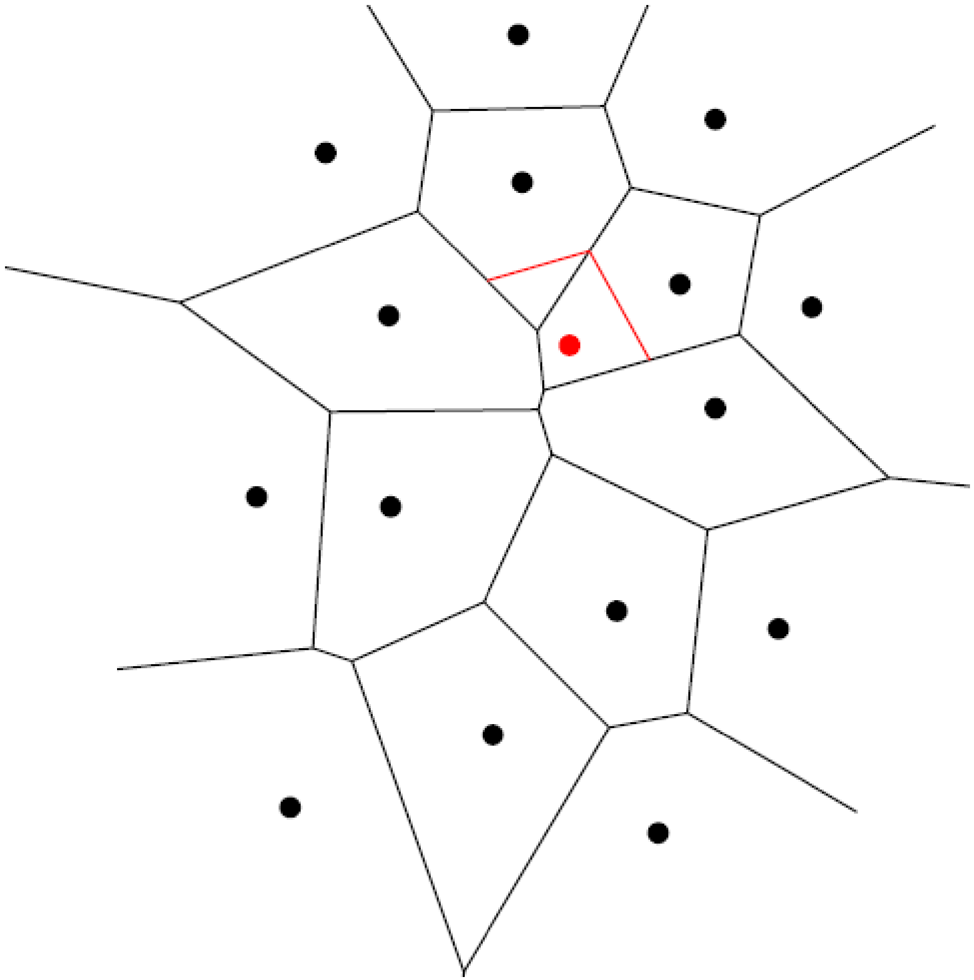
Incremental Algorithm –(contd.)

- Build its boundary starting from $b_{i+1,j}$



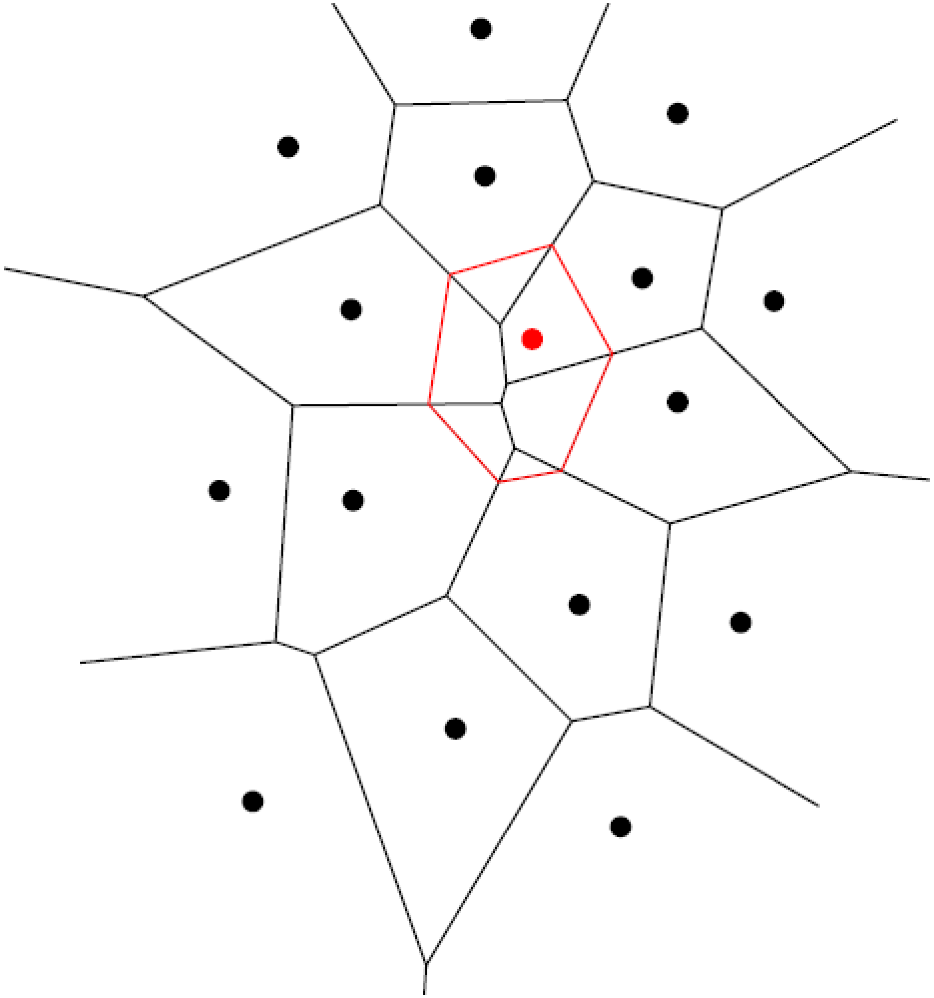
Incremental Algorithm –(contd.)

- Build its other boundaries



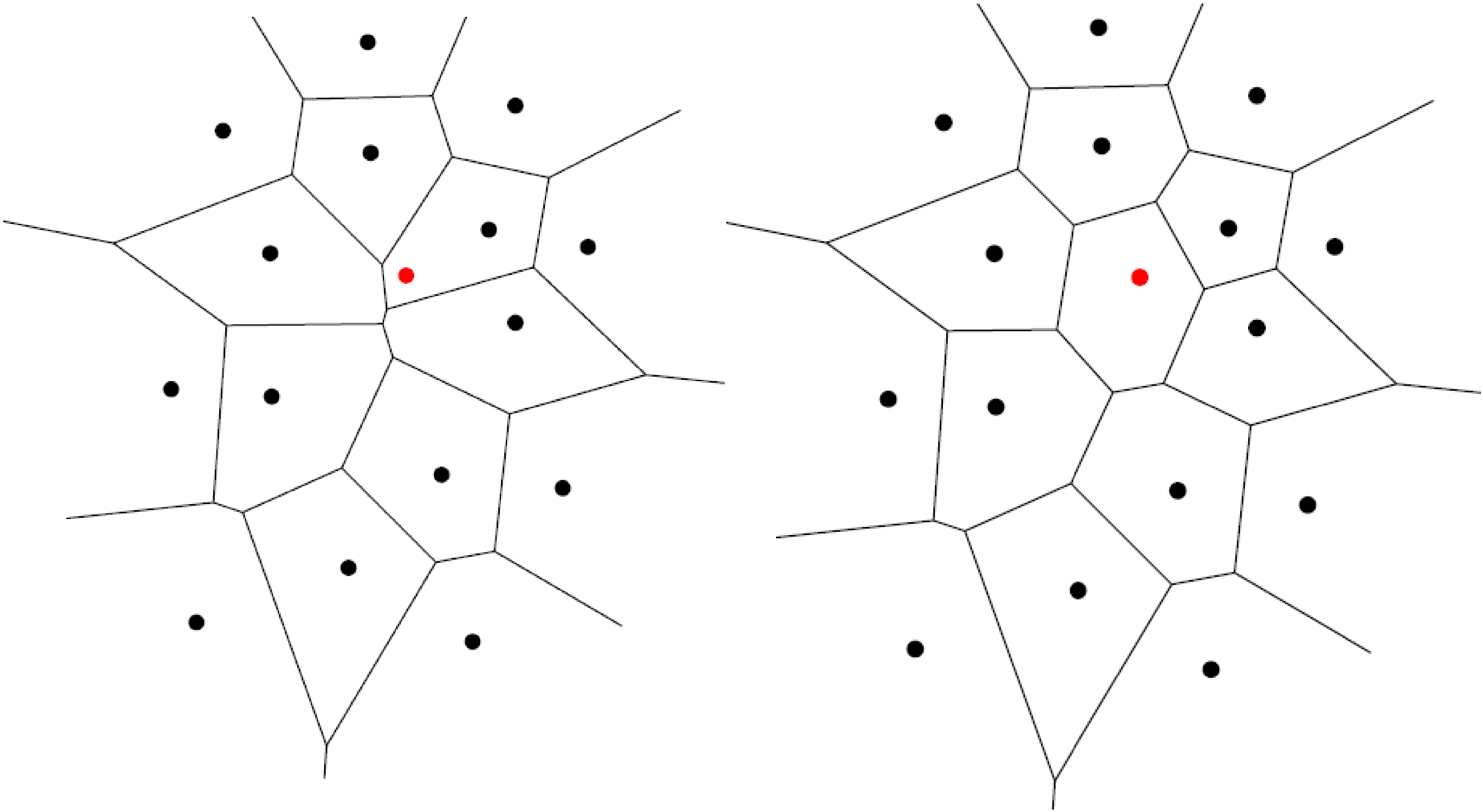
Incremental Algorithm –(contd.)

- Build its other boundaries



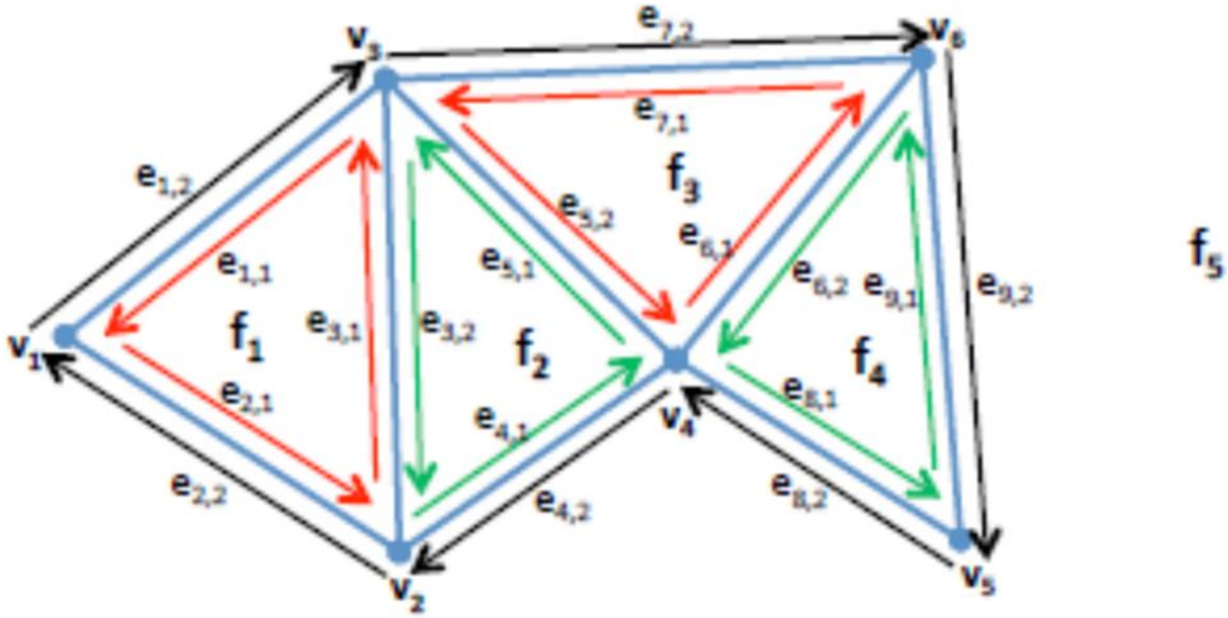
Incremental Algorithm –(contd.)

- Prune the initial diagram



Important operations

- Add a new edge
- Prune the existing edges
- Use DCEL to do these operations in constant time



Incremental algorithm

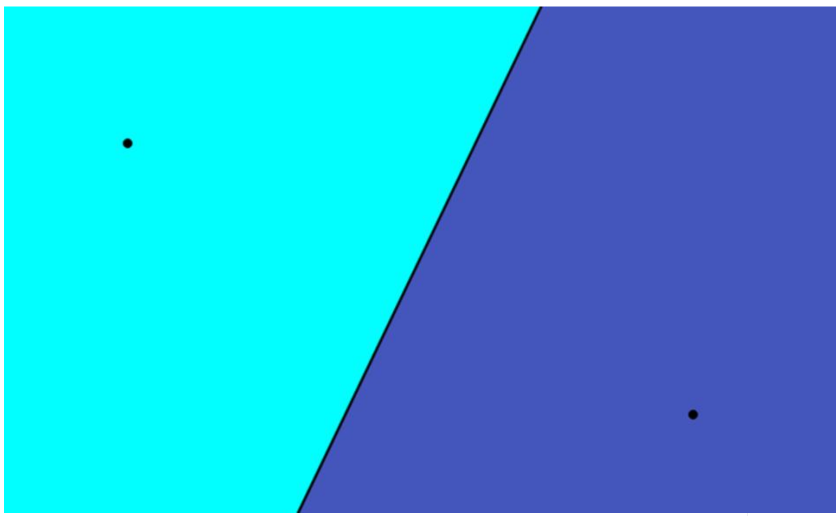
- Starts with a Voronoi diagram of $\{p_1, p_2, p_3, \ldots, p_i\}$
- Add a point p_{i+1}
  - Explore all possibilities to find the point p_j closest to p_{i+1}
- Compute p_{i+1}'s Voronoi polygon/ region
  - Build its boundary starting from $b_{i+1,j}$
- Prune the initial diagram
- While building the Voronoi region of $p_{i+1,}$ update the DCEL
- **Each step runs in O(i) time**
- Total running time is $O(n^2)$

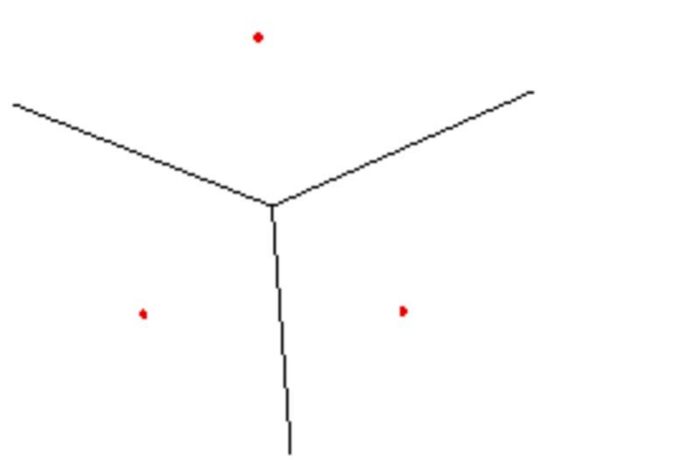
References

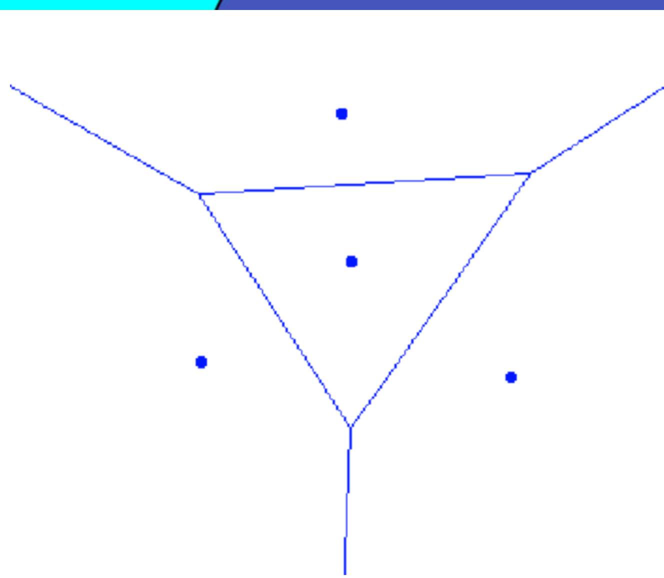
- https://dccg.upc.edu/people/vera/wp-content/uploads/2013/06/GeoC-Voronoi-algorithms.pdf by Professor Vera Sacristan
- de Berg, Van Krevald, Overmars, and Schwarzkpf, *Computational Geometry Algorithms and Applications,* Springer Third Edition, 1998
- F.P. Preparata & M.I. Shamos, *Computational Geometry An Introduction,* Springer International Edition, 1985

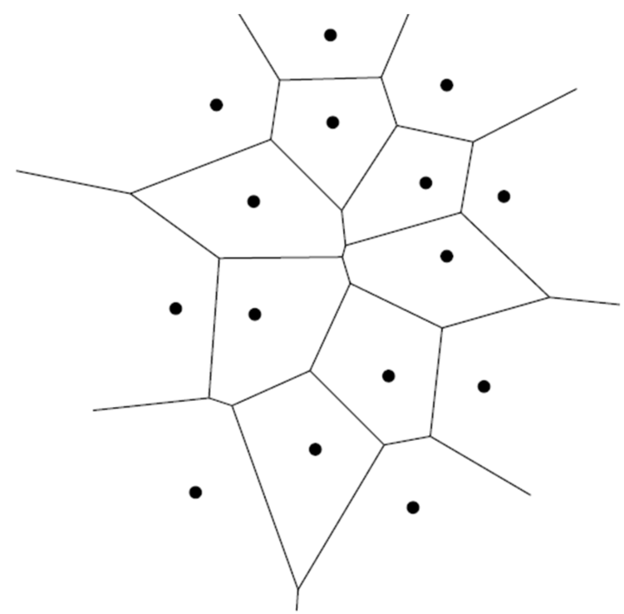
Farthest Voronoi diagram

Closest Voronoi Diagrams



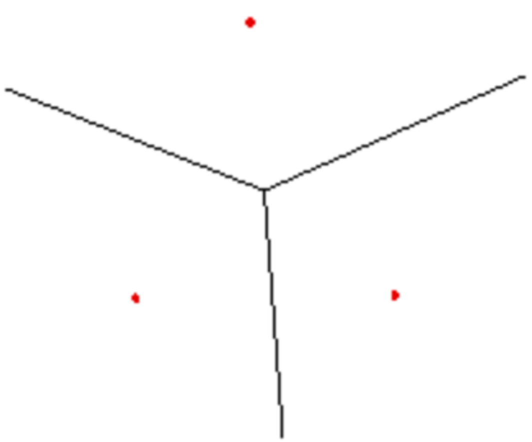






Voronoi Diagram

- Closest Voronoi diagram of a set of sites *S* decomposes the plane into regions around each site $s_i \in S$ such that each point within the region around s_i is closer to s_i than to any other site in *S*.



- Farthest Voronoi diagram of a set of sites *S* decomposes the plane into regions such that each point in the region of s_i is farther to s_i than to any other site in *S*

THANK YOU