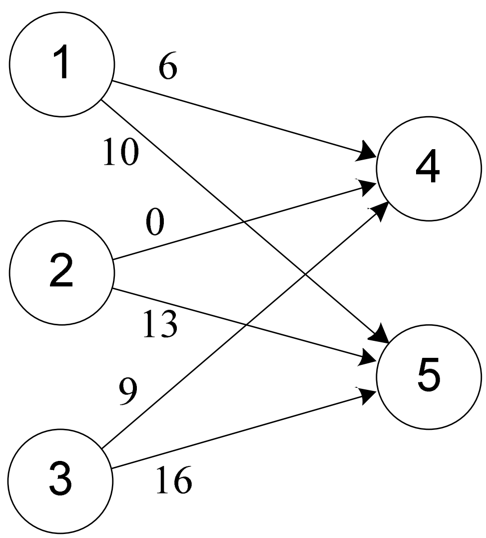
Networks 2: Shortest Paths and Road Networks

- Great circle distances are not accurate
  - Over shorter (< 50 mi) distances
  - In areas with waterways that restrict road travel
- Actual distance using road network more accurate
- Usually want to find the shortest travel time path
  - Not the shortest distance path

Graph Representations

- Complete bipartite directed (or digraph):
  - Suppliers to multiple DCs, single mode of transport



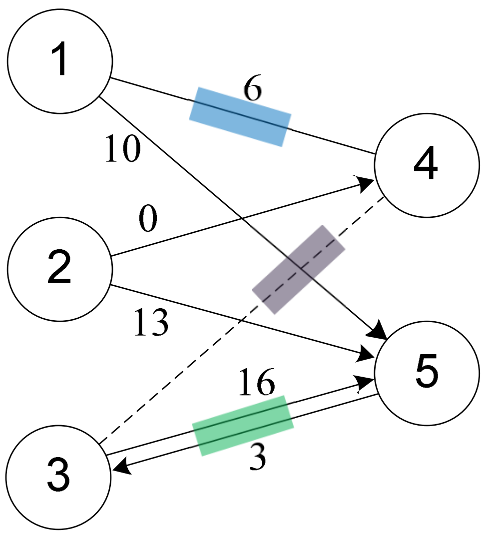
C:	1	2
1:	6	10
2:	0	13
3:	9	16

Interlevel matrix

$$W = \underbrace{\begin{matrix} 0 & 0 & 0 & \boxed{6} & 10 \\ 0 & 0 & 0 & \sqrt{\varepsilon} & 13 \\ 0 & 0 & 0 & 9 & 16 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}}_{\text{Weighted adjacency matrix}}$$

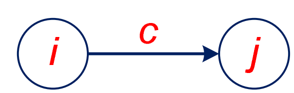
Graph Representations

- Bipartite:
  - One- or two-way connections between nodes in two groups



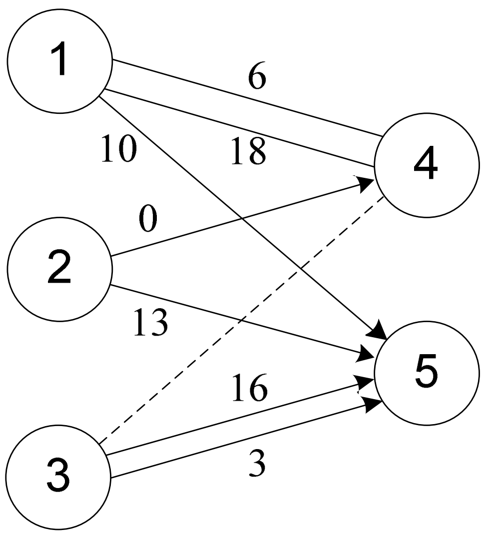
$$
W = \begin{matrix} 0 & 0 & 0 & 6 & 10 \\ 0 & 0 & 0 & \sqrt{\varepsilon} & 13 \\ 0 & 0 & 0 & 0 & 16 \\ 6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \end{matrix}
$$

$$
IJC = \left.\begin{matrix} 4 & 1 & 6 \\ 5 & 3 & 3 \\ 1 & 4 & 6 \\ 2 & 4 & 0 \\ 1 & 5 & 10 \\ 2 & 5 & 13 \\ 3 & 5 & 16 \end{matrix}\right\} \text{Arc list matrix}
$$



Graph Representations

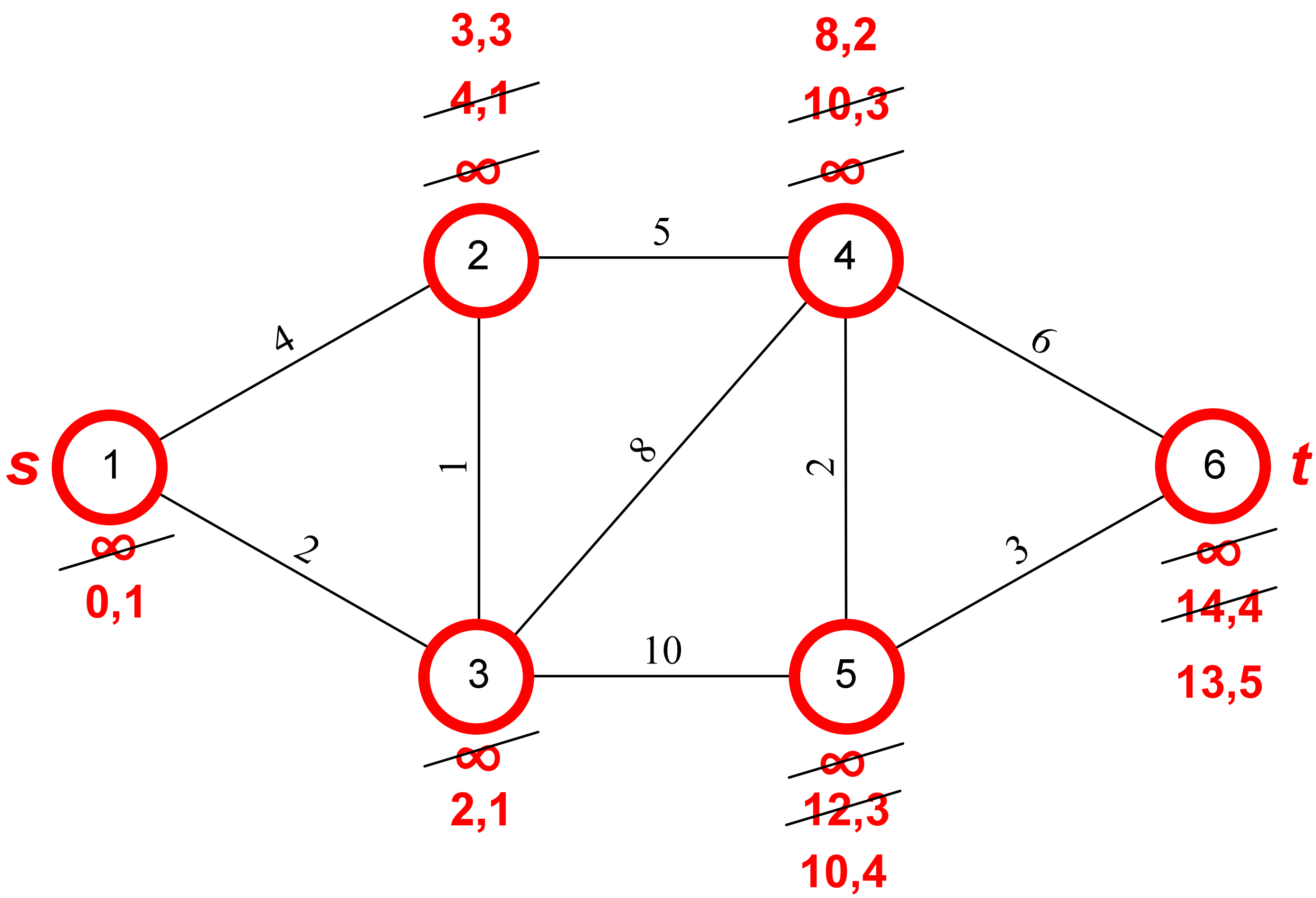
- Multigraph:
  - Multiple connections, multiple modes of transport
  - Simple graph does not have multiple connections




```
IJC = 1     -4      6
      1     -4     18
      1      5     10
      2      4      0
      2      5     13
      3      5     16
      3      5      3
no_W =
      0      0      0     24     10
      0      0      0    NaN     13
      0      0      0      0     19
     24      0      0      0      0
      0      0      0      0      0
```

Can’t represent using adjacency matrix

Dijkstra Shortest Path Procedure



Path: $1 \leftarrow 3 \leftarrow 2 \leftarrow 4 \leftarrow 5 \leftarrow 6: \quad 13$

Dijkstra Shortest Path Procedure

procedure $dijkstra(\mathbf{W}, n, s)$

$S \leftarrow \{\ \}, \quad \bar{S} \leftarrow \{1, \ldots, n\}$

for $i \in \bar{S}$, $d(i) \leftarrow \infty$, **endfor**

$d(s) \leftarrow 0, \quad pred(s) \leftarrow 0$

while $|S| < n$

$\quad i \leftarrow \arg\min_j \{d(j) : j \in \bar{S}\}$

$\quad S \leftarrow S \cup i, \quad \bar{S} \leftarrow \bar{S} \setminus i$

$\quad$ **for** $j \in \arg\{W_{i(j)} : W_{ij} \neq 0\}$

$\quad\quad$ **if** $d(j) > d(i) + W_{ij}$

$\quad\quad\quad d(j) \leftarrow d(i) + W_{ij}$

$\quad\quad\quad pred(j) \leftarrow i$

$\quad\quad$ **endif**

$\quad$ **endfor**

endwhile

return d, pred

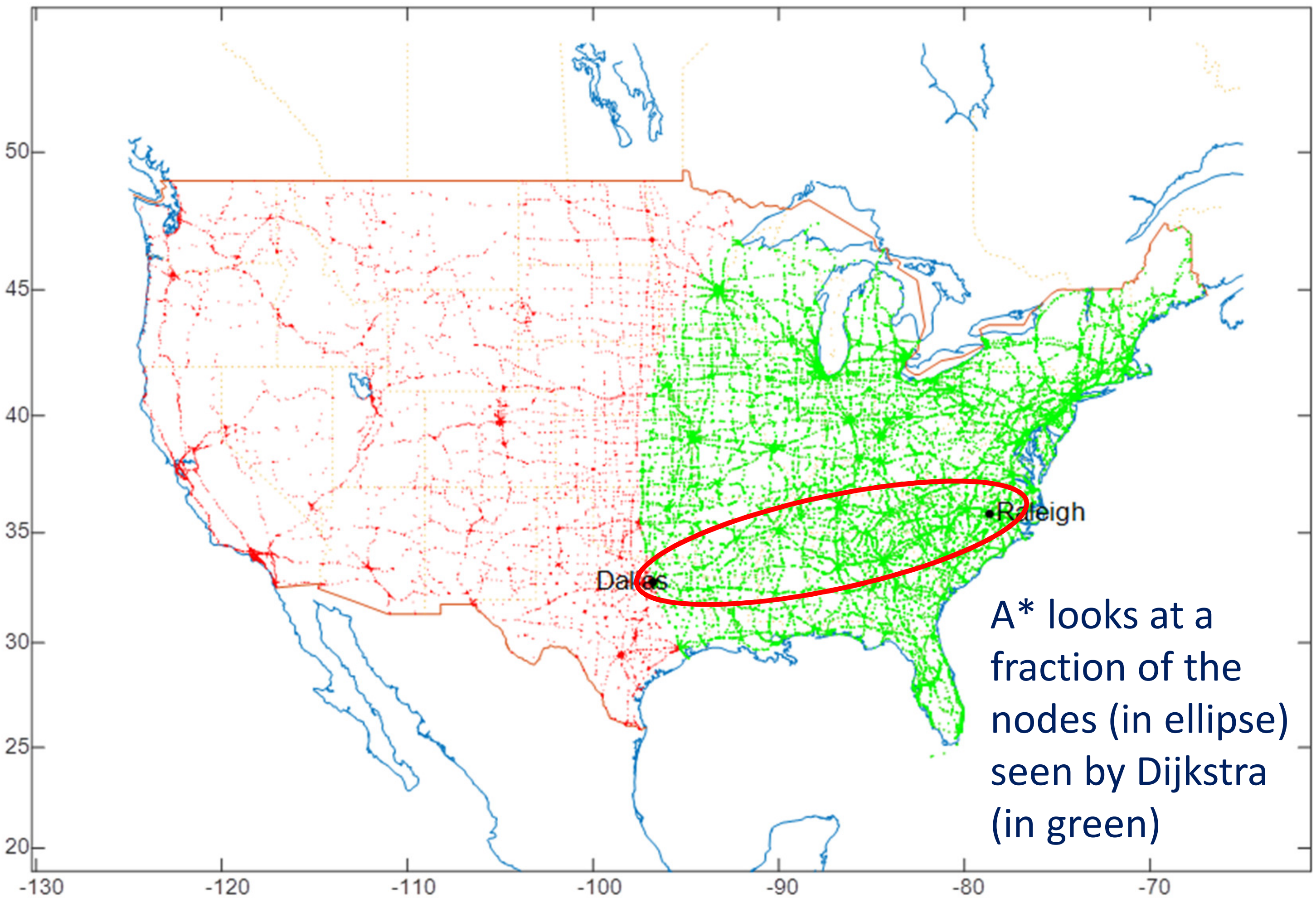
Procedure	Problem	Time Complexity
Simplex	LP	$O(2^m)$
Ellipsoid	LP	$O(m^4)$
Hungarian	Transportation	$O(m^3)$
Floyd-Warshall	Shortest Path with Cycles	$O(m^3)$
Dijkstra (linear min)	Shortest Path without Cycles	$O(m^2)$
Dijkstra (Fibonocci heap)	Shortest Path without Cycles	$O(n \log m)$
Number of nodes		m
Number of arcs		n

Other Shortest Path Procedures

- Dijkstra requires that all arcs have positive or negative lengths
  - It is a “label setting” algorithm since step to final solution made as each node labeled
  - Can find longest path (used, e.g., in CPM) by negating *all* arc lengths
- Networks with only *some* negative arcs require slower “label correcting” procedures that repeatedly check for optimality at all nodes or detect a negative cycle
  - Requires $O(n^3)$ via Floyd-Warshall algorithm (cf., $O(n^2)$ Dijkstra)
  - Negative arcs used in project scheduling to represent maximum lags between activities
- A* algorithm adds to Dijkstra an heuristic LB estimate of each node’s remaining distance to destination
  - Used in AI search for all types of applications (tic-tac-toe, chess)
  - In path planning applications, great circle distance from each node to destination could be used as LB estimate of remaining distance

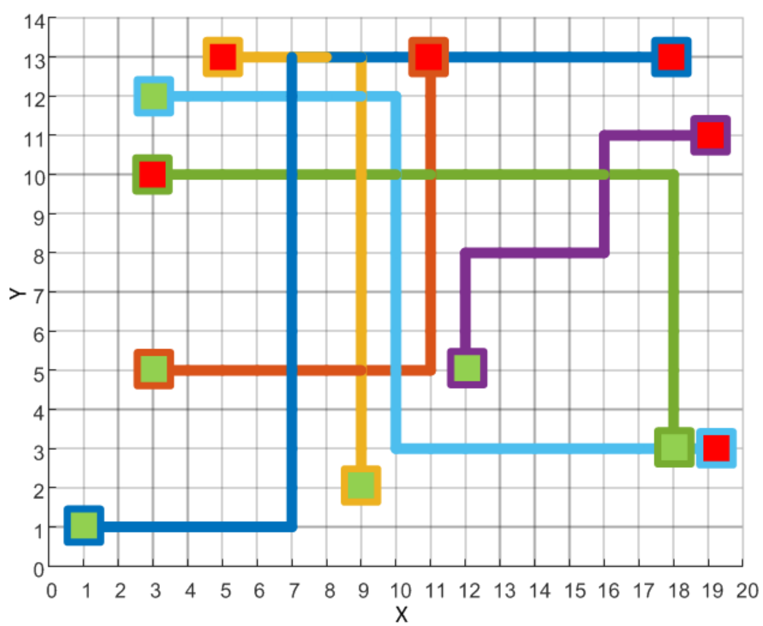
A* Path Planning Example 1

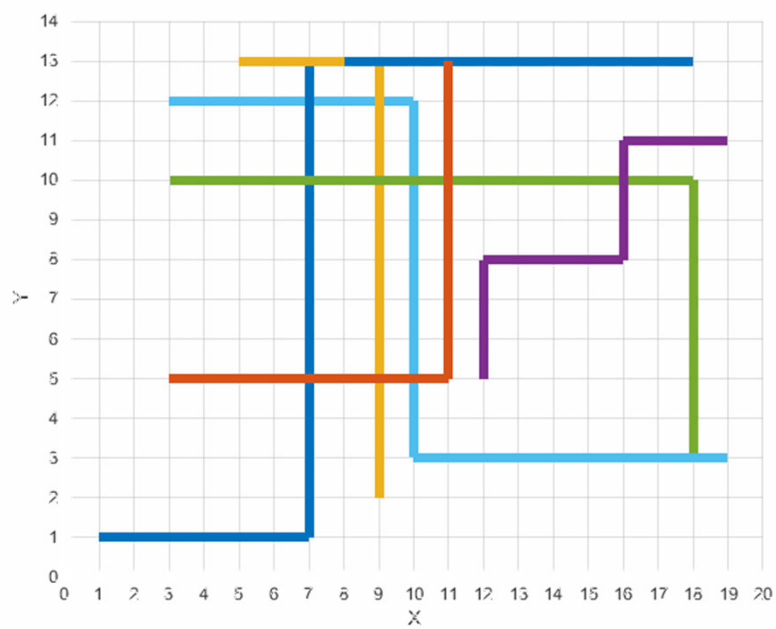
$$d_{A^*}(\text{Raleigh, Dallas}) = d_{dijk}(\text{Raleigh}, i) + d_{GC}(i, \text{Dallas}), \quad \text{for each node } i$$



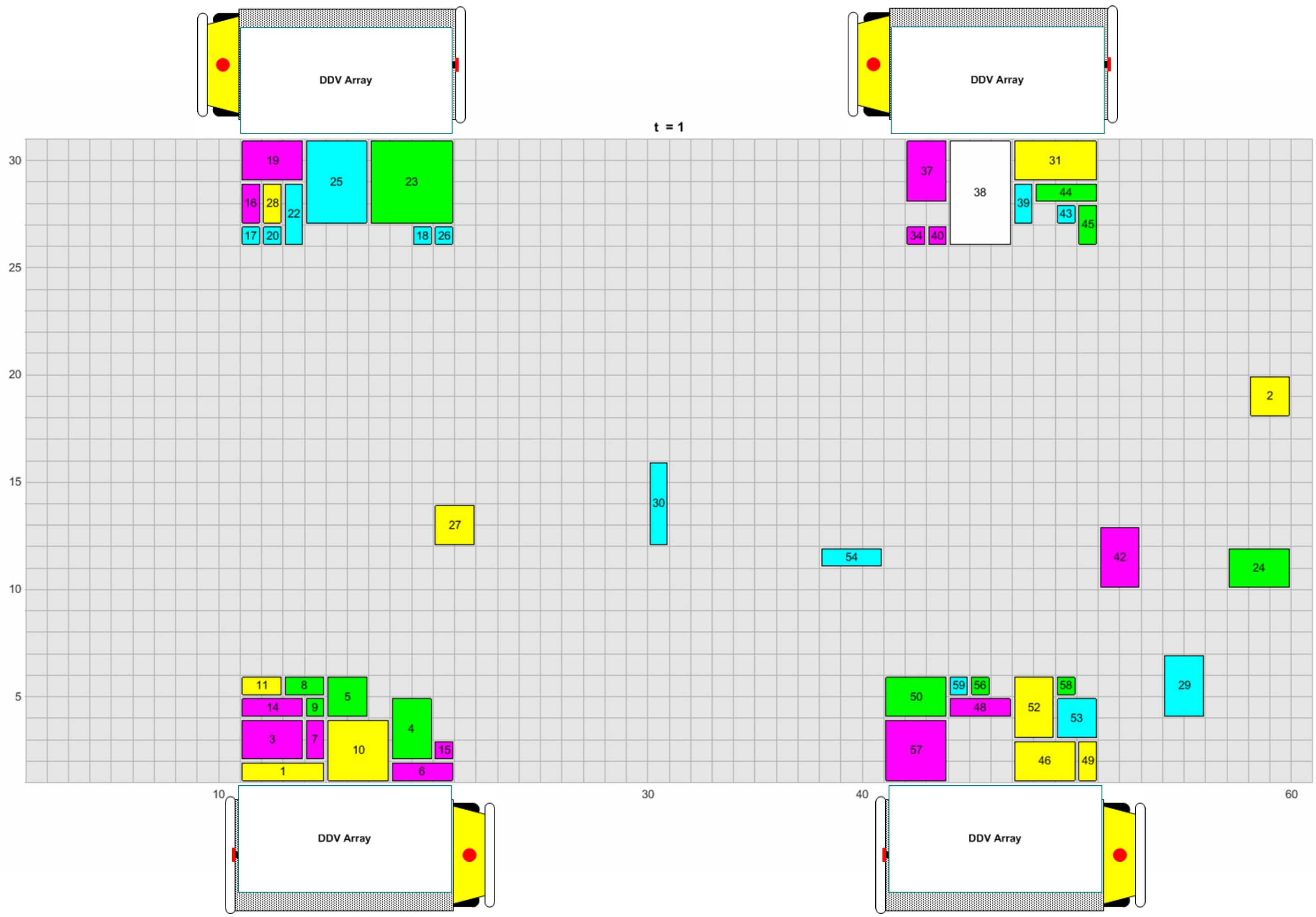
A* Path Planning Example 2

- 3-D (x,y,t) A* used for planning path of each container in a DC
- Each container assigned unique priority that determines planning sequence
  – Paths of higher-priority containers become obstacles for subsequent containers





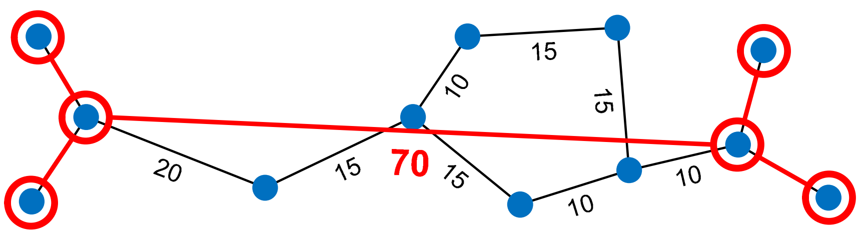
A* Path Planning Example 2



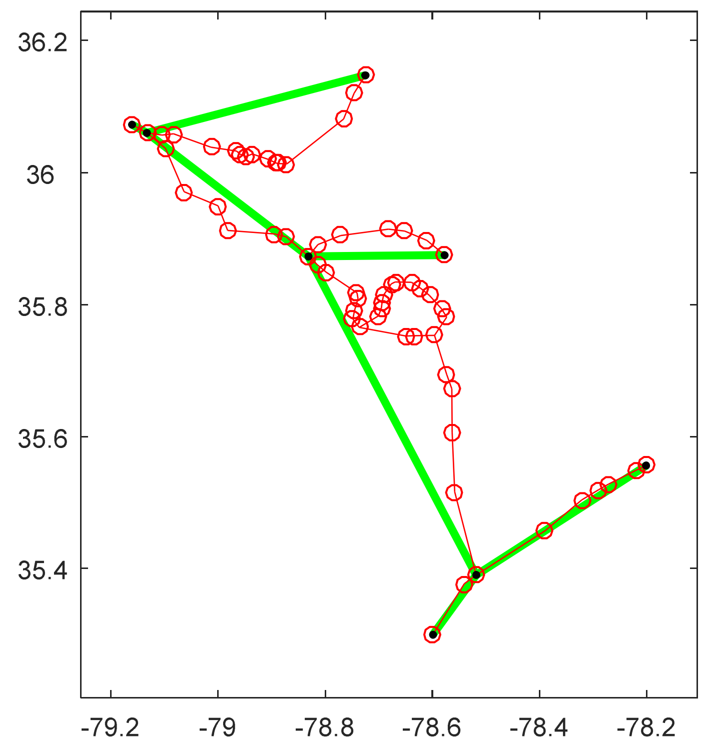
Road Network Modifications

1. Thin

- Remove all degree-2 nodes from network
- Add cost of both arcs incident to each degree-2 node
- If results in multiple arcs between pair of nodes, keep minimum cost



Thinned I-40 Around Raleigh

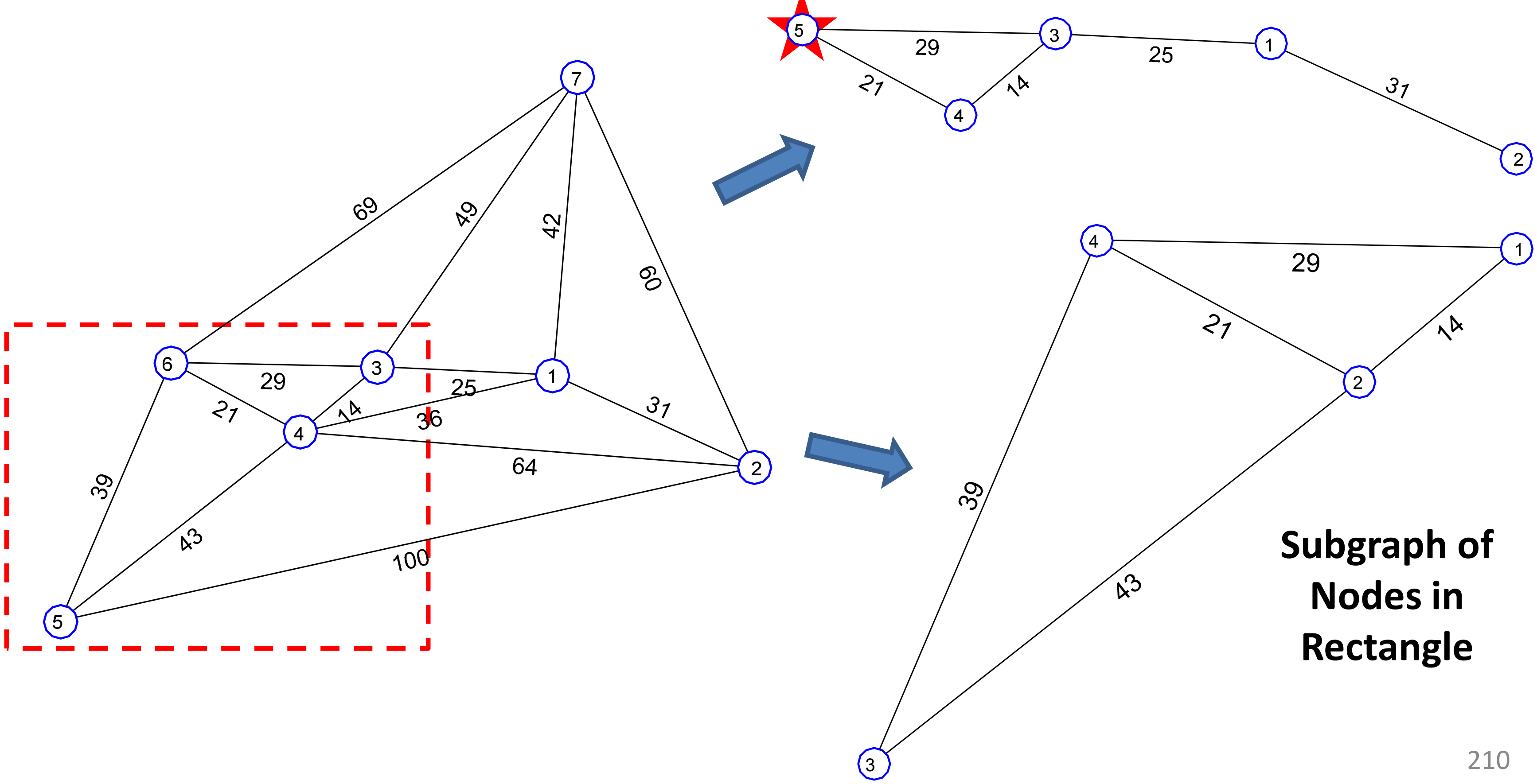


Road Network Modifications

2. Prune and Reindex

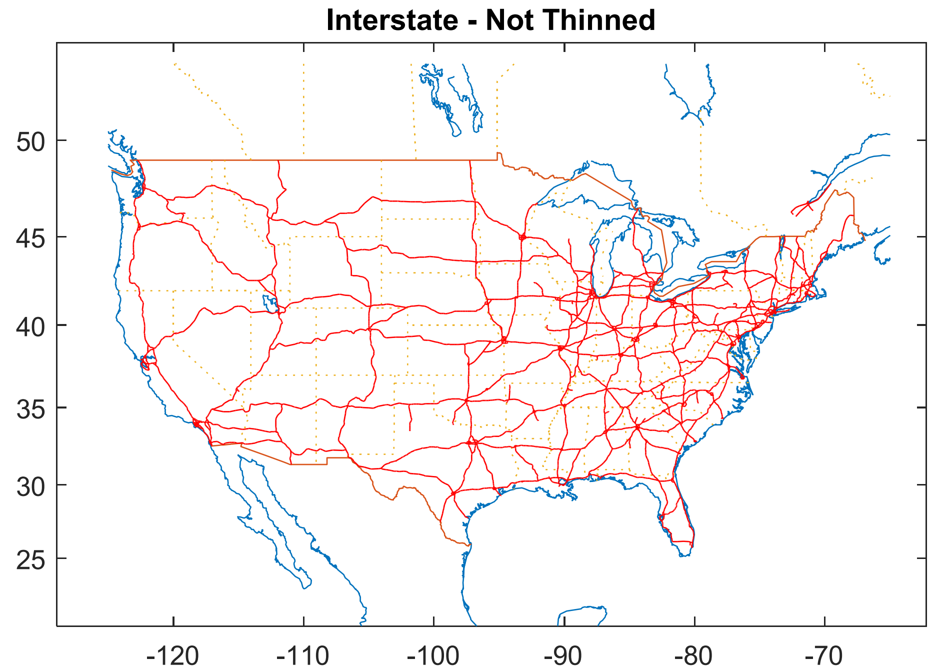
– Extract portion of graph with only those nodes and/or arcs that satisfy some condition

Subgraph of Arcs < 35

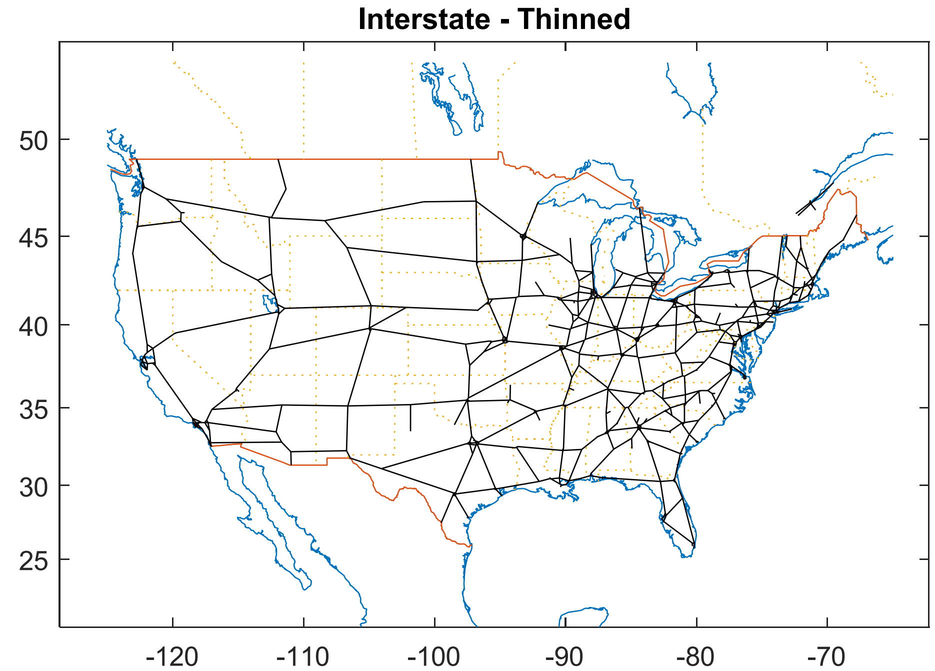


Road Network Modifications

11,124 Arcs



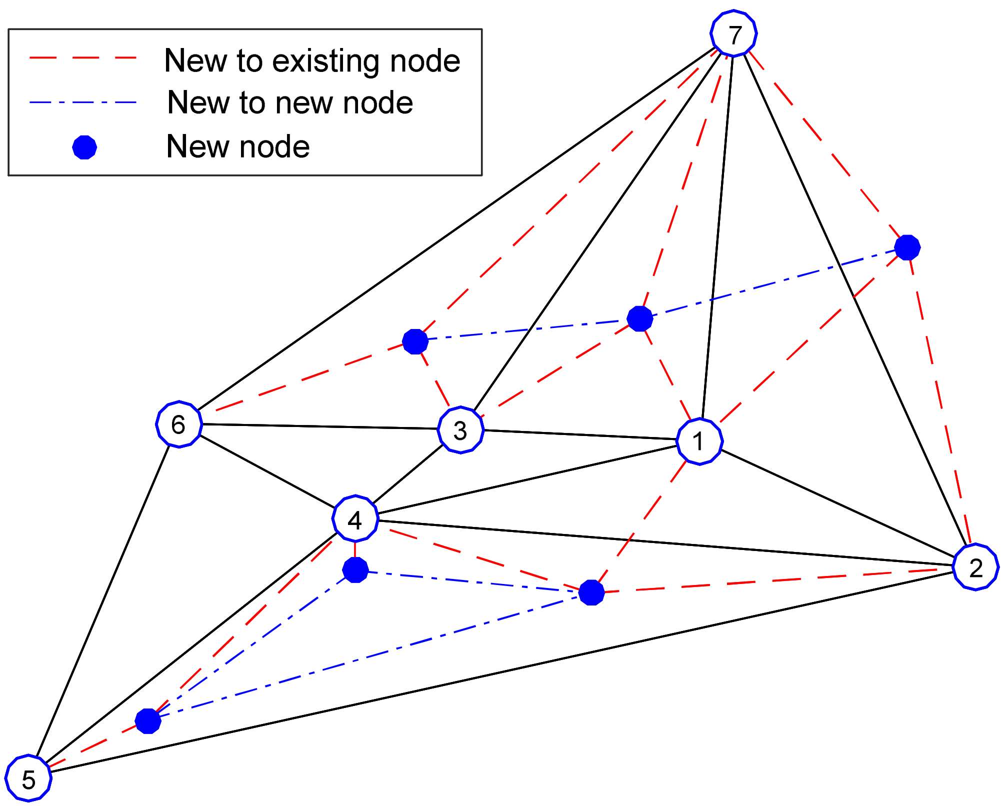
1,223 Arcs



Road Network Modifications

3. Add connector

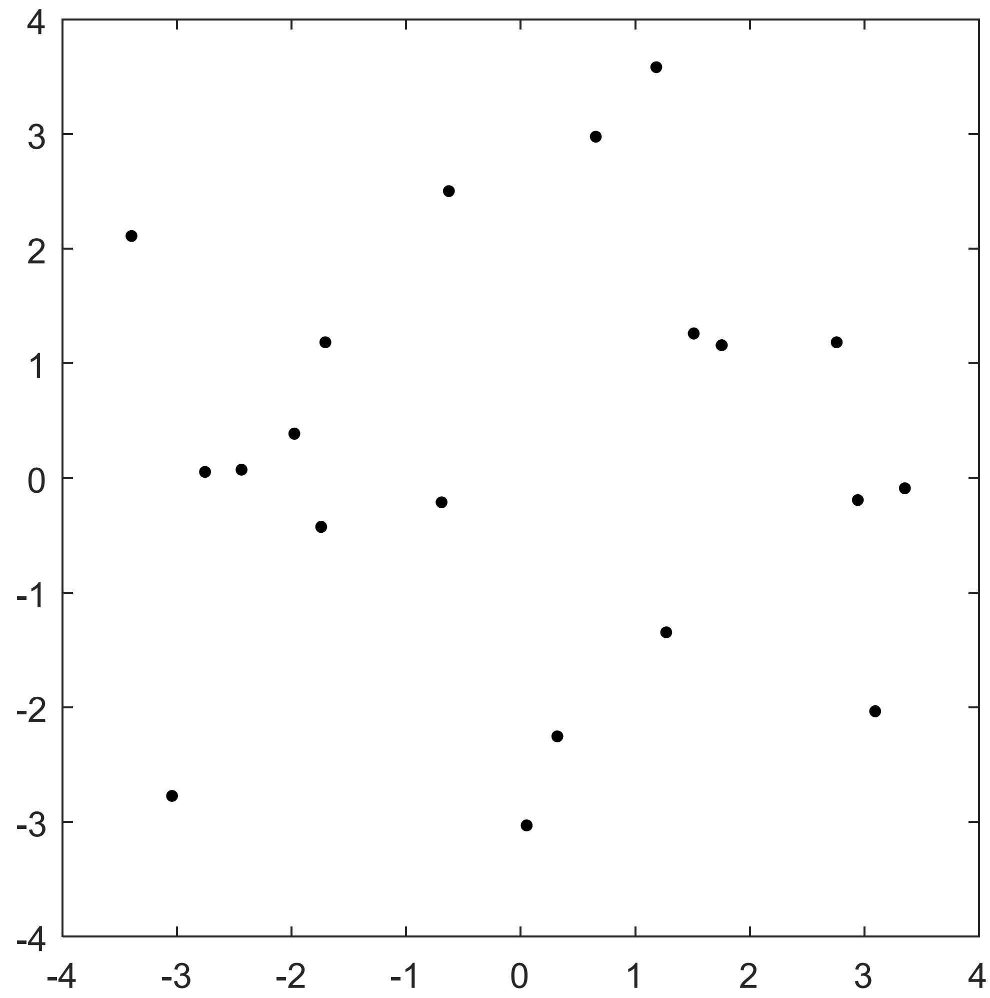
- Given new nodes, add arcs that connect the new nodes to the existing nodes in a graph and to each other



- Distance of connector arcs = GC distance x circuity factor (1.3)
- New node connected to 3 closest existing nodes, except if
  - Ratio of closest to 2nd and 3rd closest < threshold (0.1)
  - Distance shorter using other connector and graph

Computational Geometry

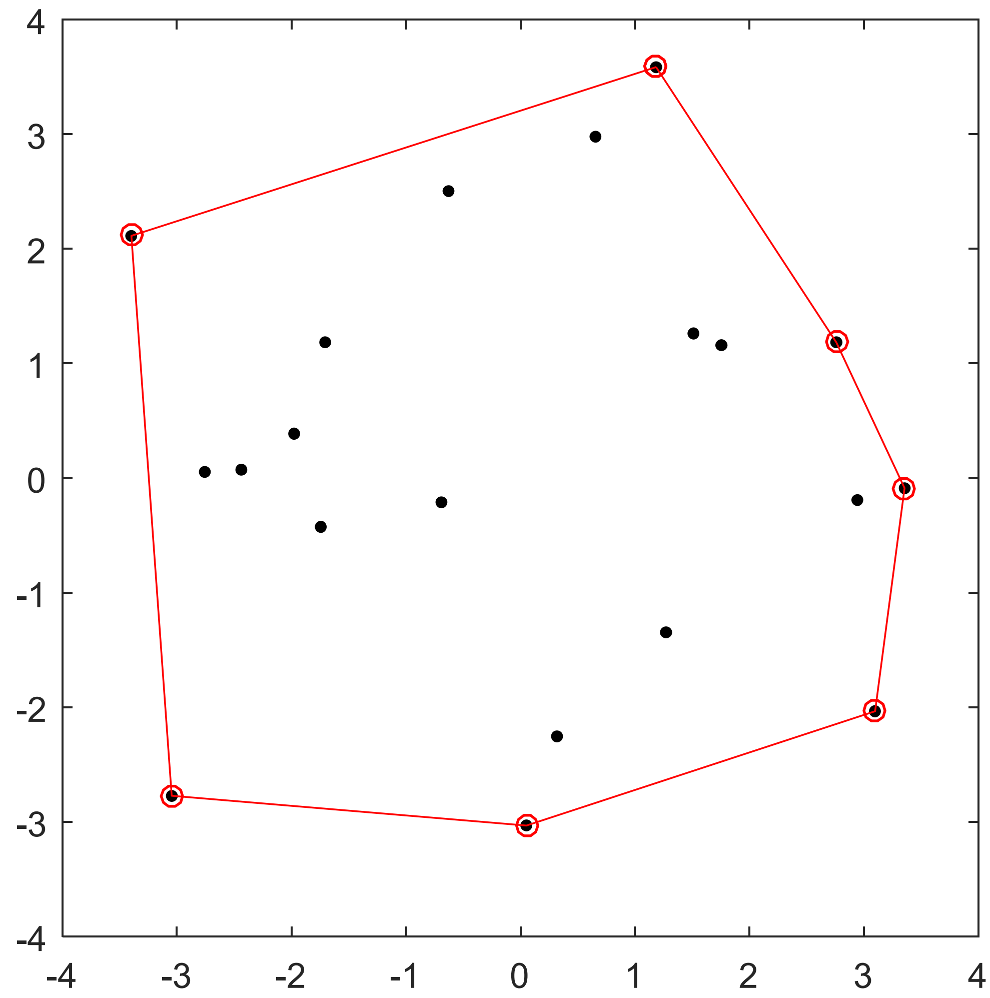
- Design and analysis of algorithms for solving geometric problems
  - Modern study started with Michael Shamos in 1975
- Facility location:
  - geometric data structures used to "simplify" solution procedures



Convex Hull

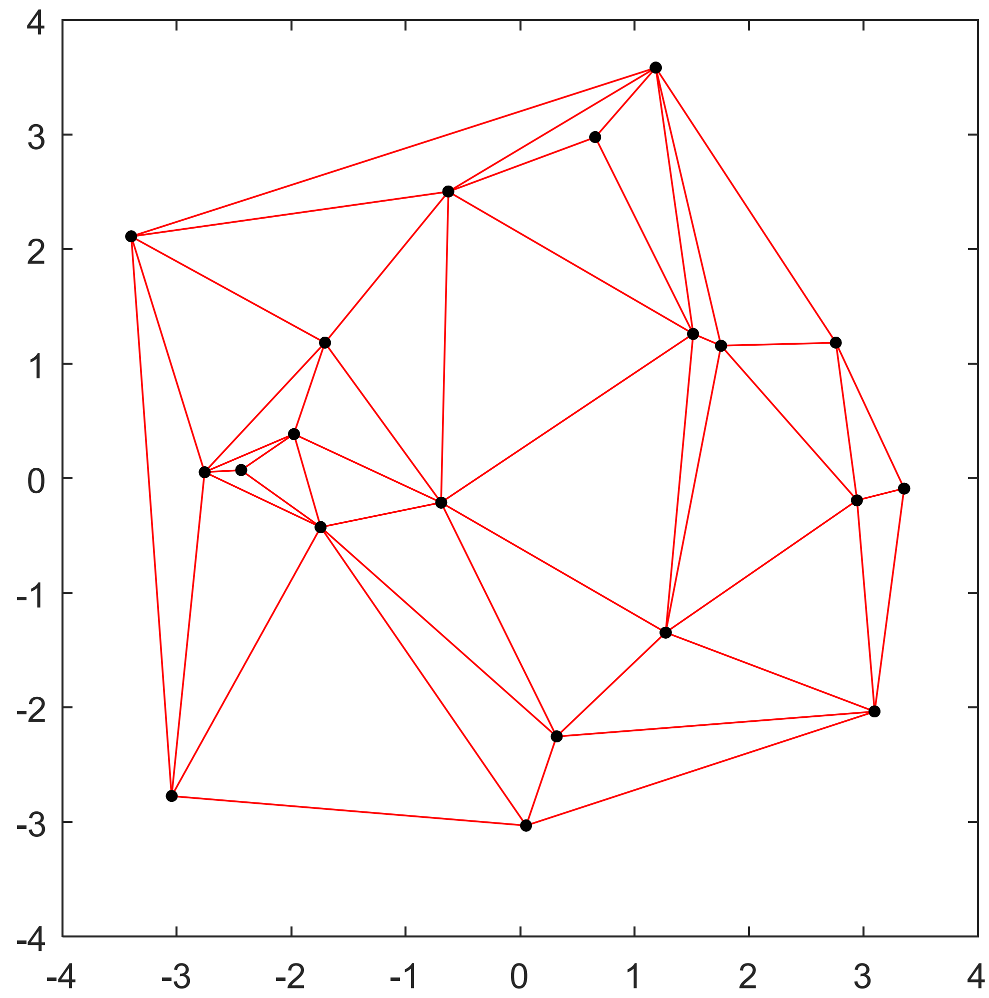
- Find the points that enclose all points
  - Most important data structure
  - Calculated, via Graham's scan in

    $O(n \log n), \quad n$ points



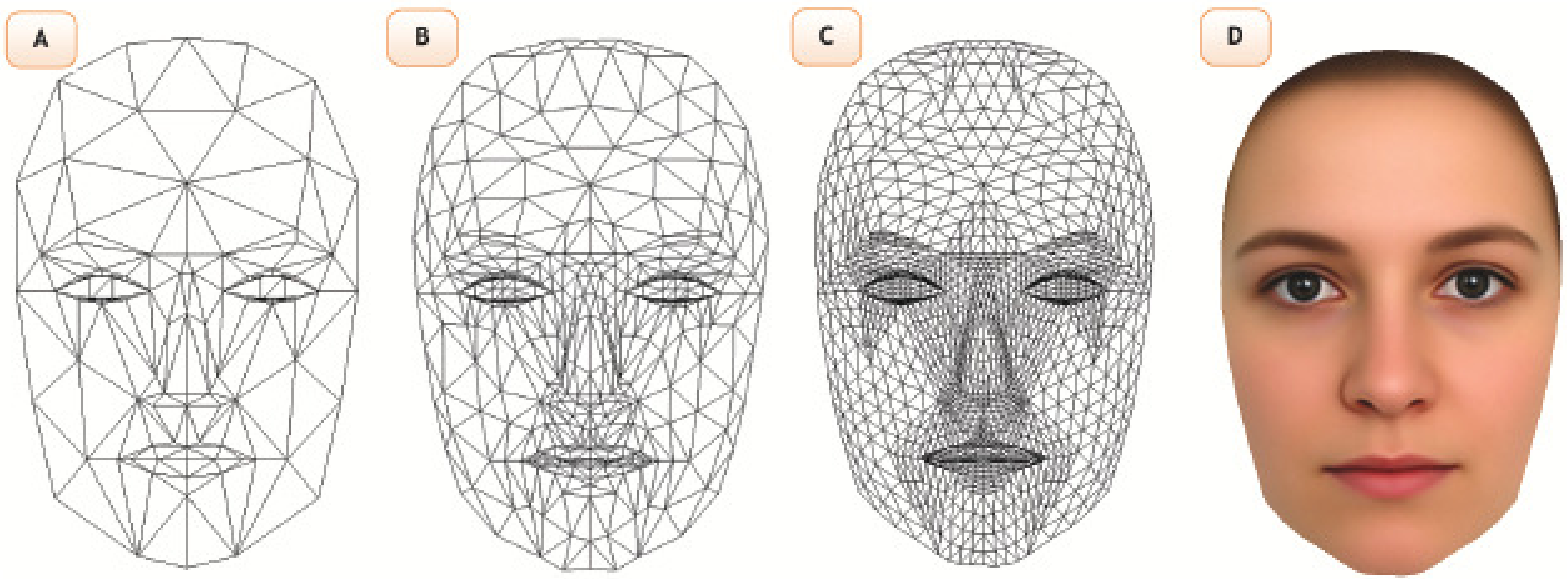
Delaunay Triangulation

- Find the triangulation of points that maximizes the minimum angle of any triangle
  - Captures proximity relationships
  - Used in 3-D animation
  - Calculated, via divide and conquer, in $O(n \log n)$, n points



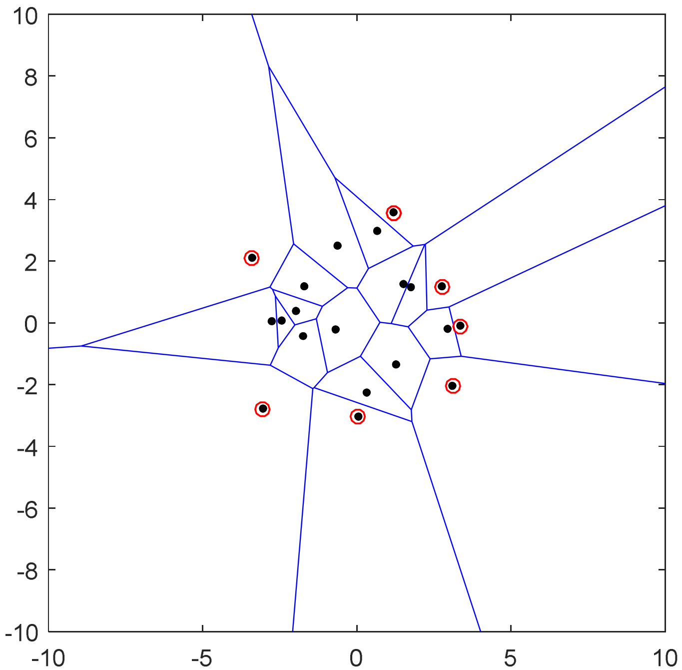
3-D Delaunay Triangulation

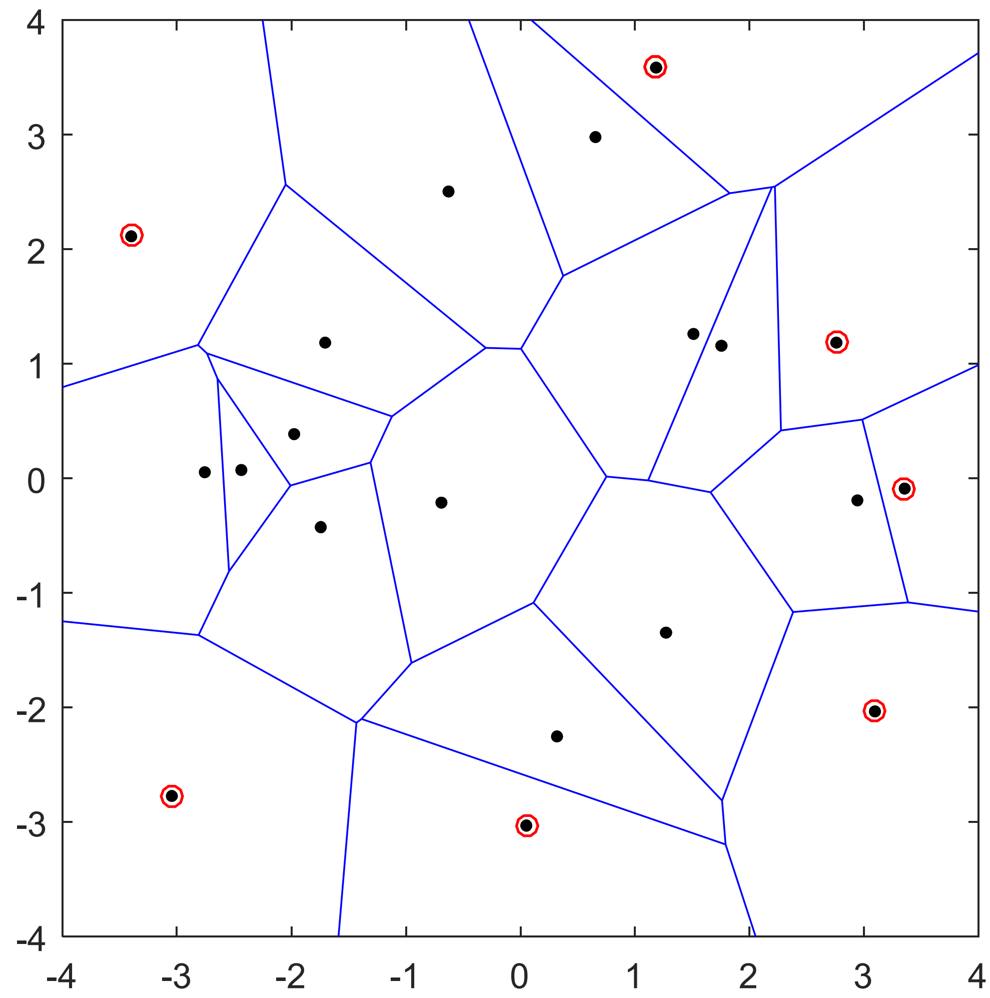
- Used in computer graphics to render synthetic images
  - Orientation of each triangle face used to determine reflection and shading relative to light sources and other objects



Voronoi Diagram

- Each region defines area closest to a point
  - Open face regions indicate points in convex hull





Delaunay-Voronoi

- Delaunay triangulation is straight-line dual of Voronoi diagram
  - Can easily convert from one to another