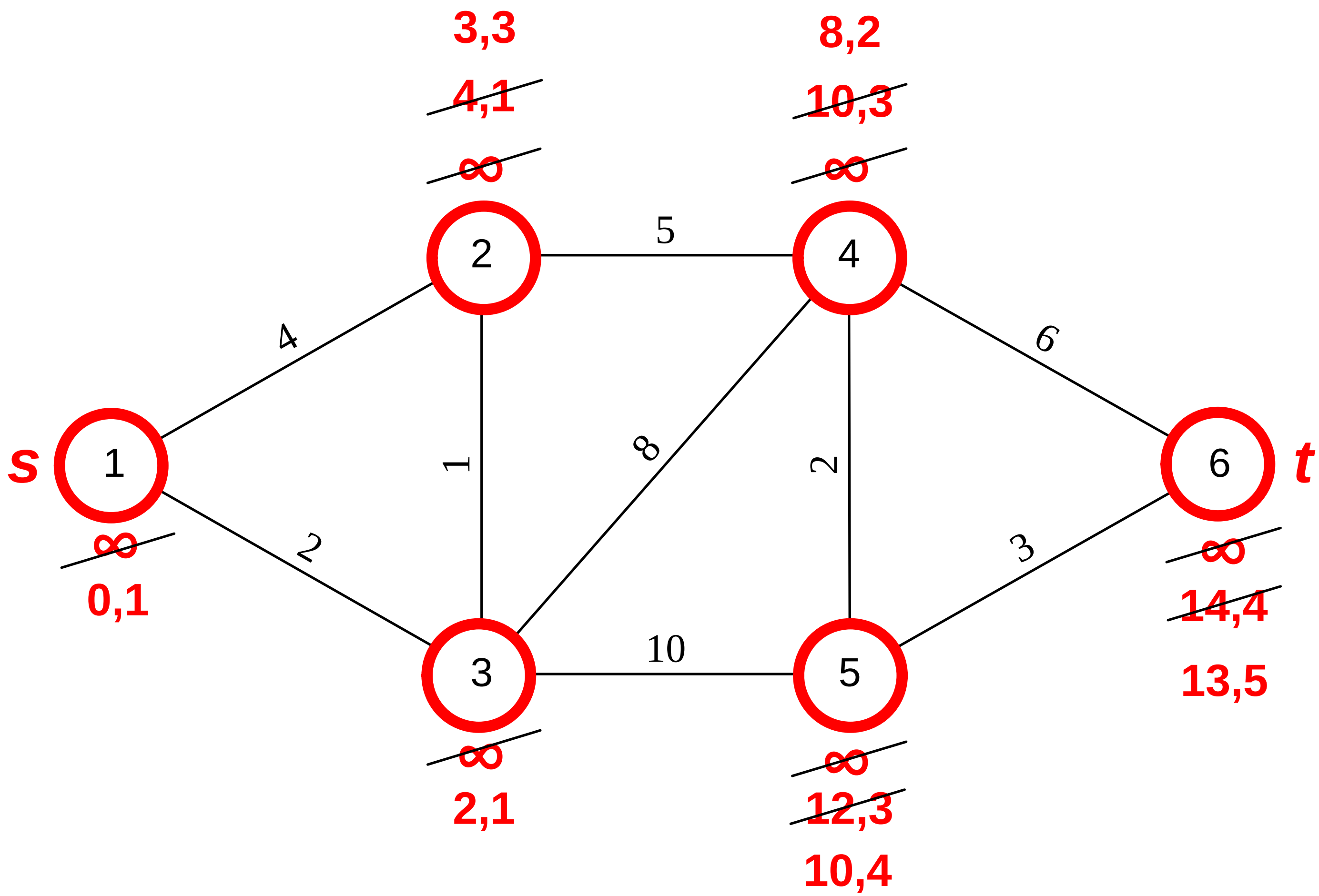
Dijkstra Shortest Path Procedure



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

Dijkstra Shortest Path Procedure

$$
\begin{array}{l}
\textbf{procedure } \mathit{dijkstra}(\mathbf{W}, n, s) \\
S \leftarrow \{\ \}, \quad \bar{S} \leftarrow \{1, \ldots, n\} \\
\textbf{for } i \in \bar{S},\ d(i) \leftarrow \infty,\ \textbf{endfor} \\
d(s) \leftarrow 0, \quad pred(s) \leftarrow 0 \\
\textbf{while } |S| < n \\
\quad i \leftarrow \arg\min_{j} \{ d(j) : j \in \bar{S} \} \\
\quad S \leftarrow S \cup i, \quad \bar{S} \leftarrow \bar{S} \setminus i \\
\quad \textbf{for } j \in \arg\{ W_{i(j)} : W_{ij} \neq 0 \} \\
\qquad \textbf{if } d(j) > d(i) + W_{ij} \\
\qquad\quad d(j) \leftarrow d(i) + W_{ij} \\
\qquad\quad pred(j) \leftarrow i \\
\qquad \textbf{endif} \\
\quad \textbf{endfor} \\
\textbf{endwhile} \\
\textbf{return d, pred}
\end{array}
$$

```
%% DIJKSTRA Matlab code,
%  given W, n, and s
S = []; nS = 1:n;
d = inf(1,n);
d(s) = 0; pred(s) = 0;
while length(S) < n
    [di,idx] = min(d(nS));
    i = nS(idx);
    S = [S i];
    nS(idx) = [];
    pred(d > di + W(i,:)) = i;
    d = min(d,di + W(i,:));
end
d, pred
```

Index to index vector nS

Order important

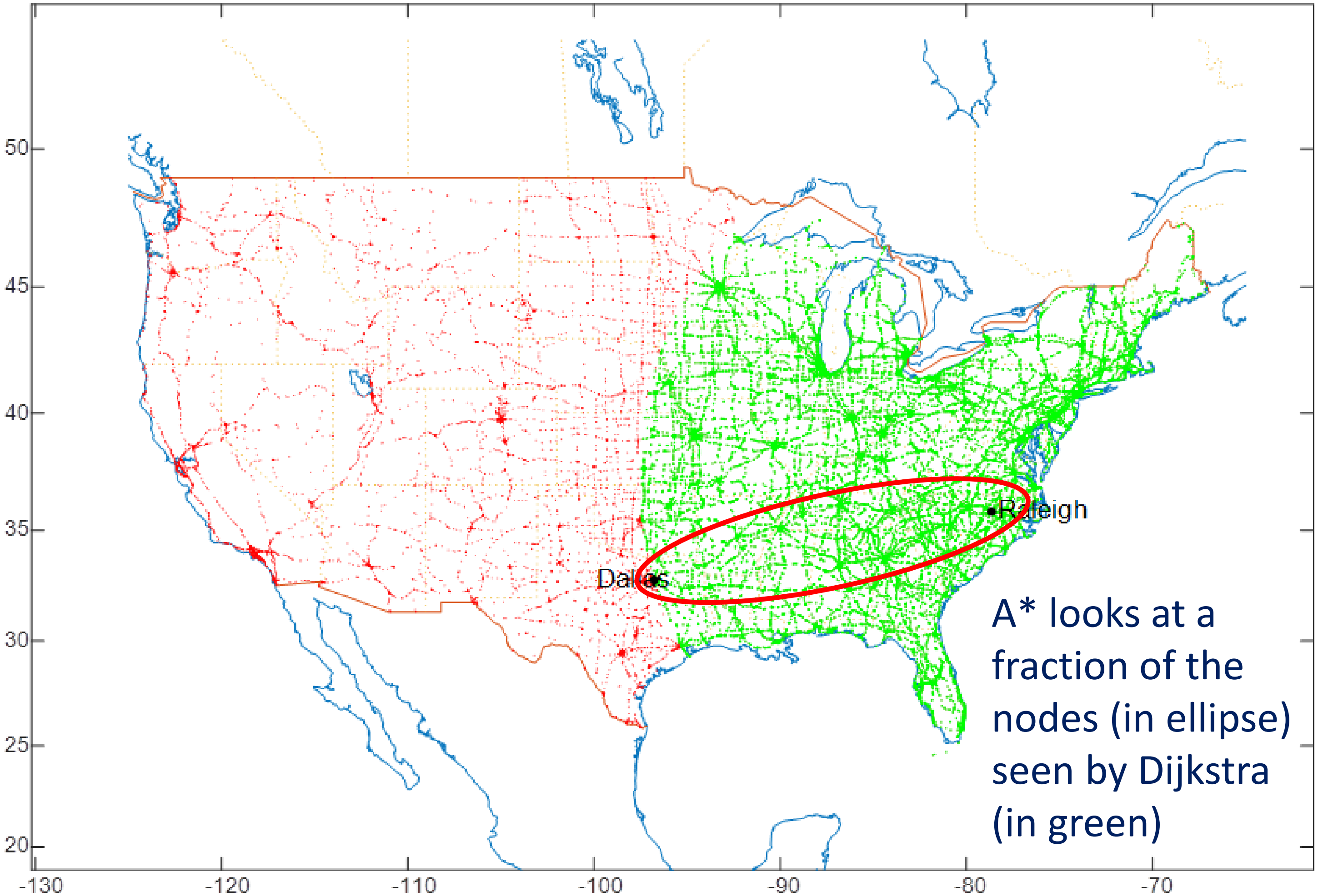
$O(2^n)$	Simplex (LP)
$O(n^4)$	Ellipsoid (LP)
$O(n^3)$	Hungarian (transportation)
$O(n^2)$	Dijkstra (linear min)
$O(m \log n)$	Dijkstra (Fibonocci heap)
m	no. arcs

Other Shortest Path Procedures

- Dijkstra requires that all arcs have nonnegative 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

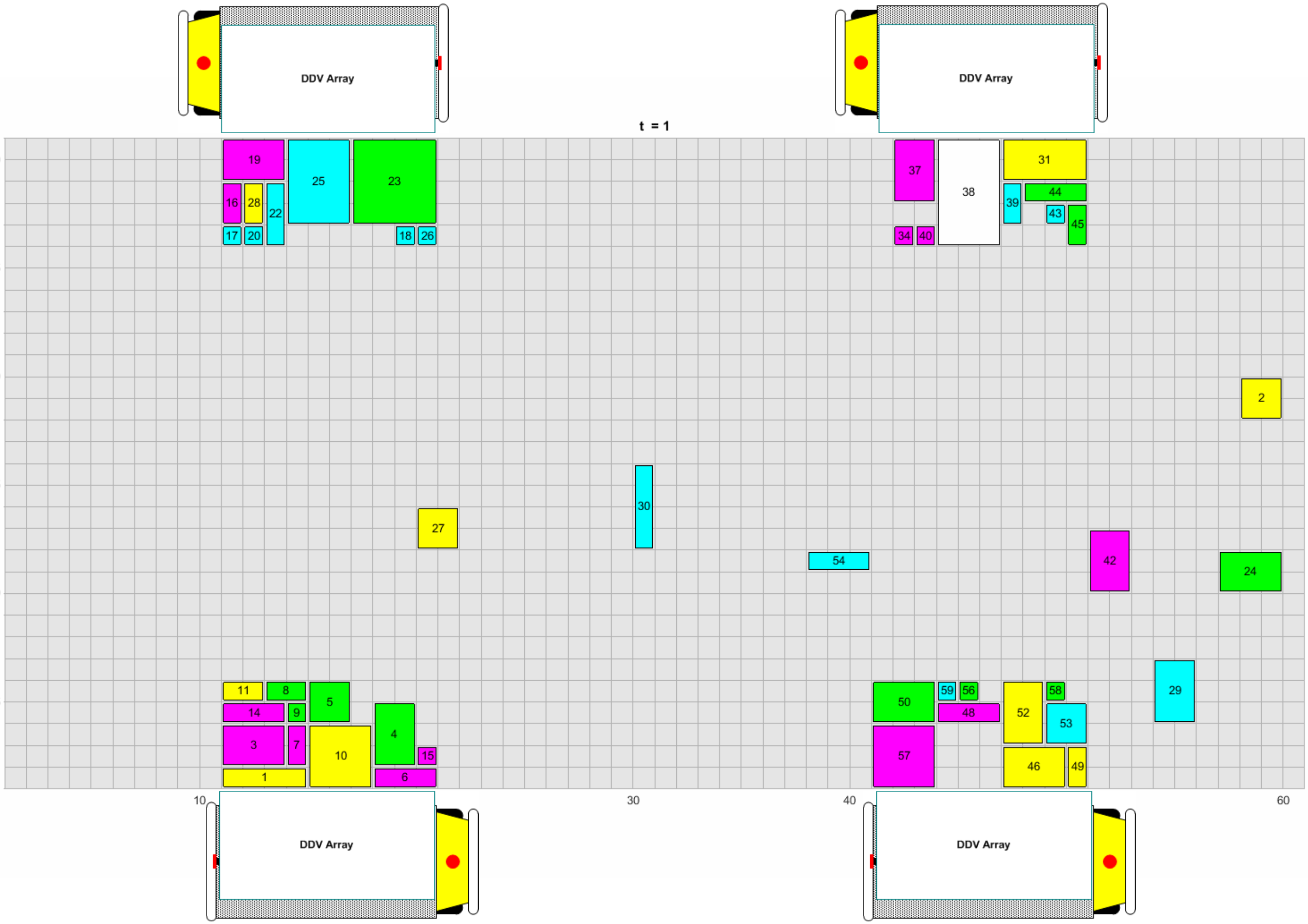
$$d_{A^*}(\text{Raleigh, Dallas}) = d_{dijk}(\text{Raleigh}, i) + d_{GC}(i, \text{Dallas}), \quad \text{for each node } i$$



A* Path Planning Example 2

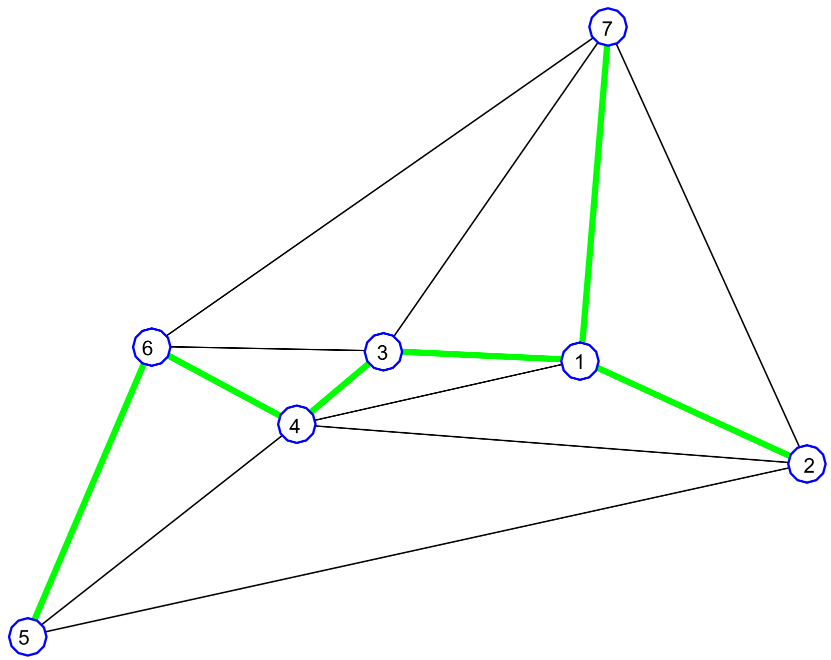
- 3-D (x,y,t) A* used for planning path of each container in a DC
- Each container assigned unique priority that determines planning sequence
  - Paths of higher-priority containers become obstacles for subsequent containers

A* Path Planning Example 2



Minimum Spanning Tree

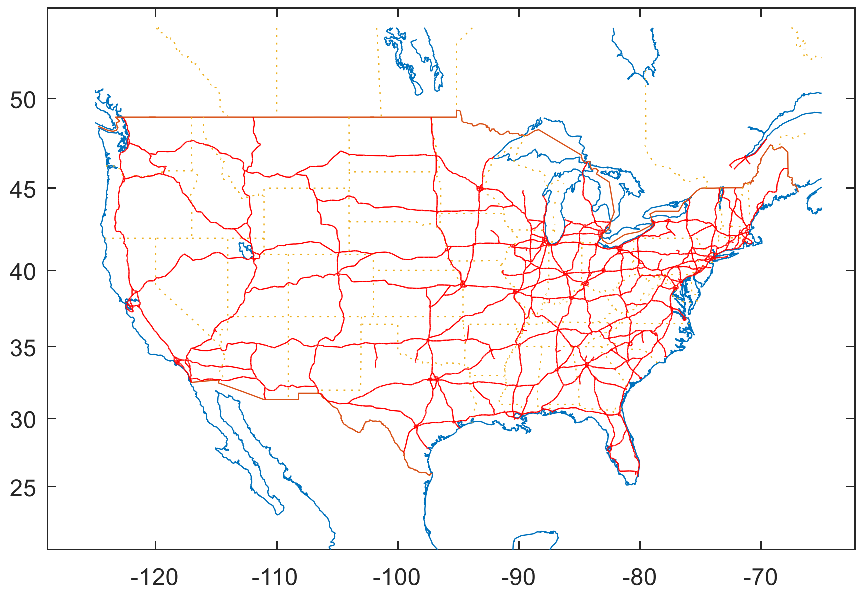
- Find the minimum cost set of arcs that connect all nodes
  - Undirected arcs: Kruskal’s algorithm (easy to code)
  - Directed arcs: Edmond’s branching algorithm (hard to code)



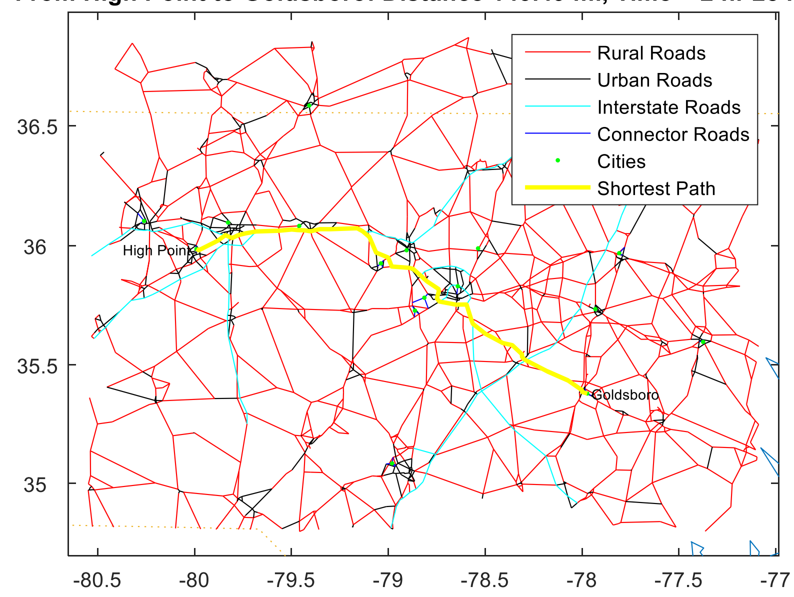
U.S. Highway Network

- Oak Ridge National Highway Network
  - Approximately 500,000 miles of roadway in US, Canada, and Mexico
  - Created for truck routing, does not include residential
  - Nodes attributes: XY, FIPS code
  - Arc attributes: IJD, Type (Interstate, US route), Urban

U.S. Interstate Road Network



From High Point to Goldsboro: Distance 143.49 mi, Time = 2 hr 28 min



FIPS Codes

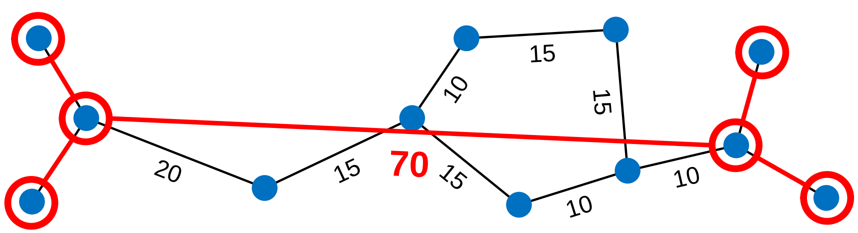
- Federal Information Processing Standard (FIPS) codes used to uniquely identify states (2-digit) and counties (3-digit)
  - 5-digit Wake county code = 2-digit state + 3-digit county = 37183 = 37 NC FIPS + 183 Wake FIPS

1	AL	Alabama	22	LA	Louisiana	40	OK	Oklahoma
2	AK	Alaska	23	ME	Maine	41	OR	Oregon
4	AZ	Arizona	24	MD	Maryland	42	PA	Pennsylvania
5	AR	Arkansas	25	MA	Massachusetts	44	RI	Rhode Island
6	CA	California	26	MI	Michigan	45	SC	South Carolina
8	CO	Colorado	27	MN	Minnesota	46	SD	South Dakota
9	CT	Connecticut	28	MS	Mississippi	47	TN	Tennessee
10	DE	Delaware	29	MO	Missouri	48	TX	Texas
11	DC	Dist Columbia	30	MT	Montana	49	UT	Utah
12	FL	Florida	31	NE	Nebraska	50	VT	Vermont
13	GA	Georgia	32	NV	Nevada	51	VA	Virginia
15	HI	Hawaii	33	NH	New Hampshire	53	WA	Washington
16	ID	Idaho	34	NJ	New Jersey	54	WV	West Virginia
17	IL	Illinois	35	NM	New Mexico	55	WI	Wisconsin
18	IN	Indiana	36	NY	New York	56	WY	Wyoming
19	IA	Iowa	37	NC	North Carolina	72	PR	Puerto Rico
20	KS	Kansas	38	ND	North Dakota	88		Canada
21	KY	Kentucky	39	OH	Ohio	91		Mexico

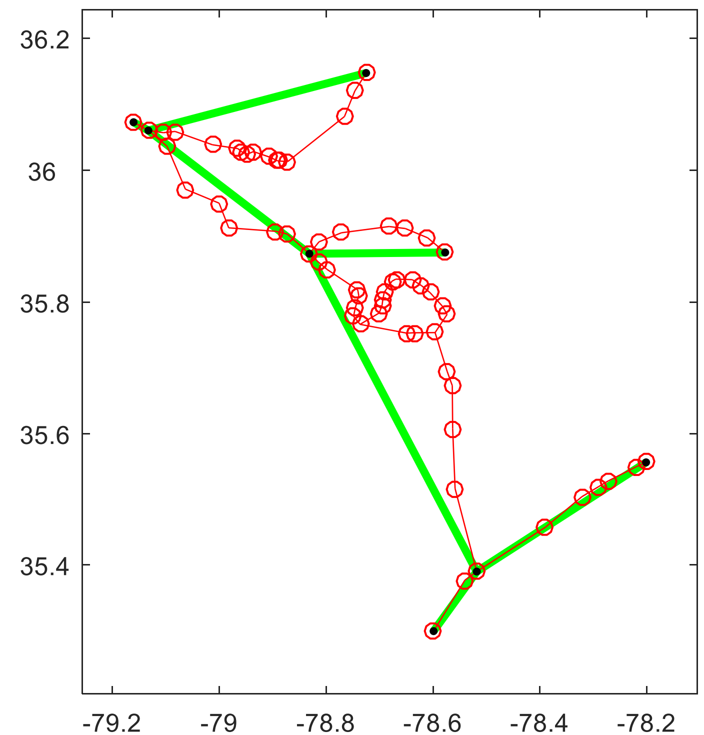
Road Network Modifications

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



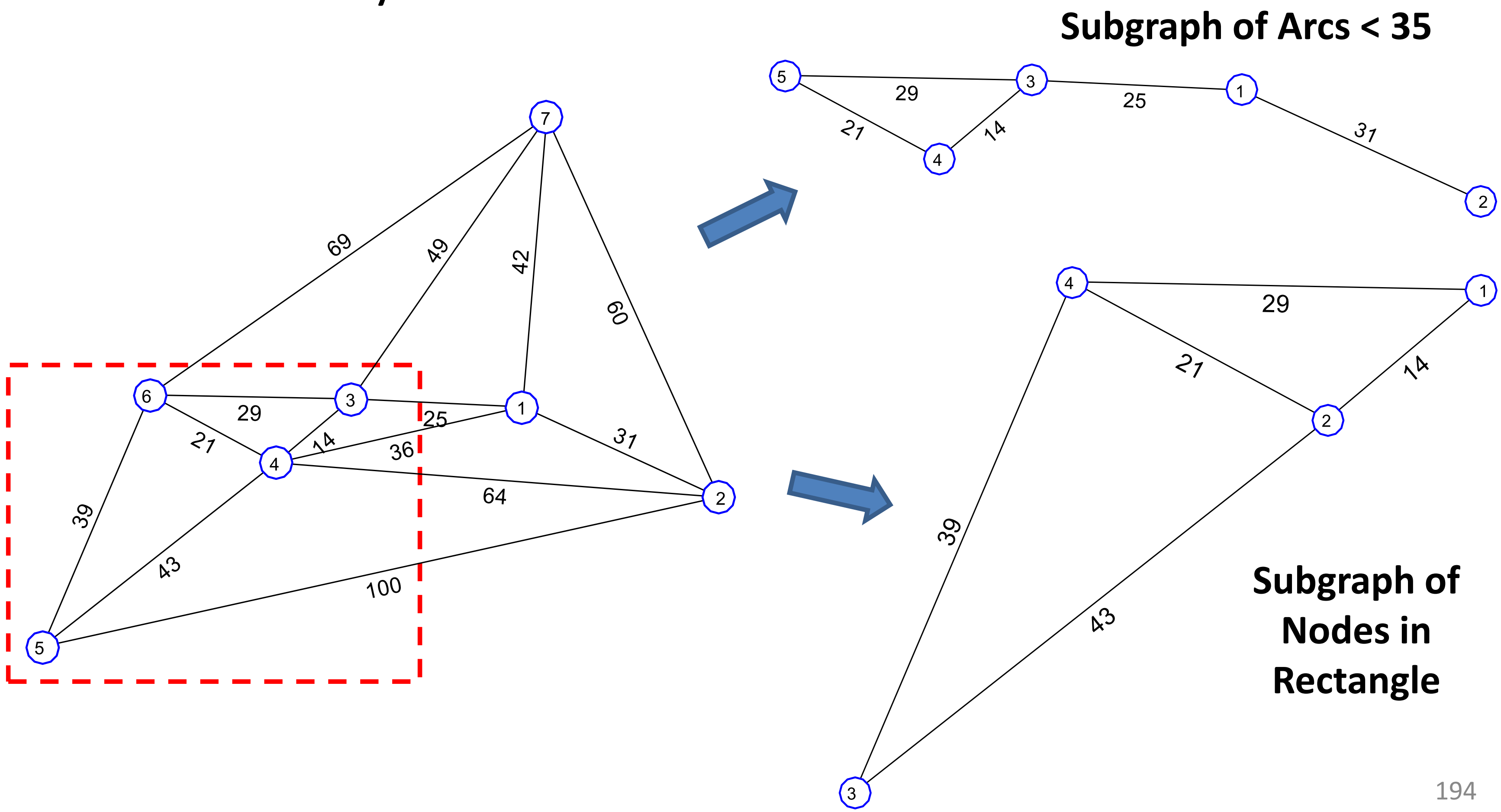
Thinned I-40 Around Raleigh



Road Network Modifications

2. Subgraph

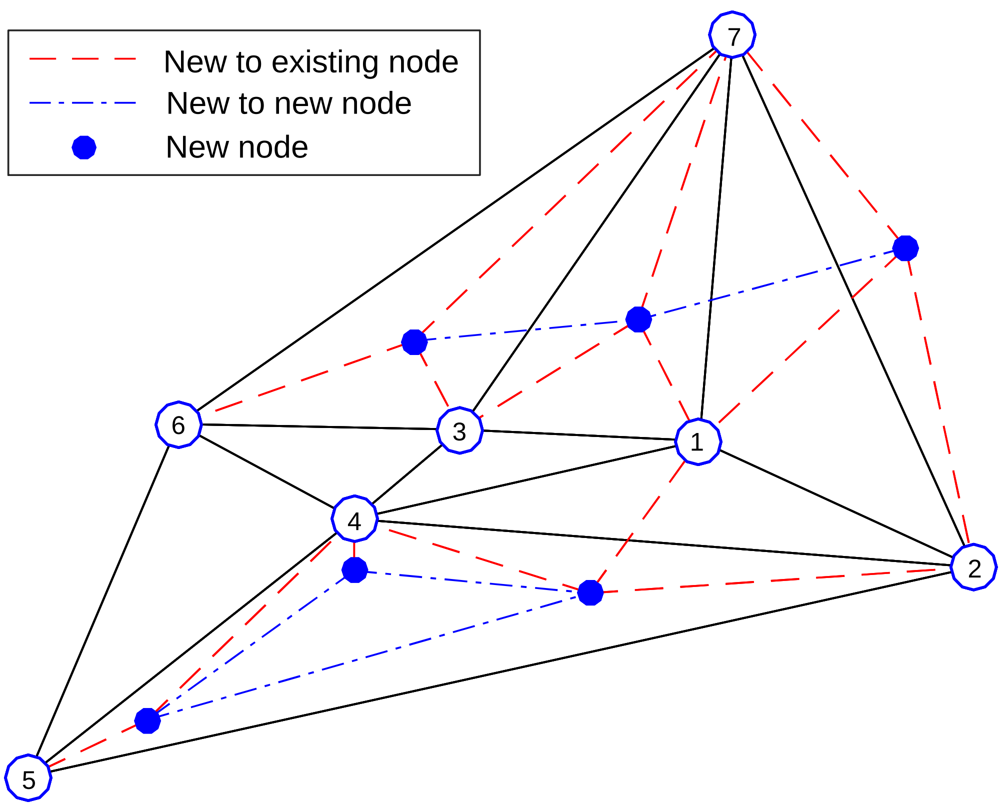
– Extract portion of graph with only those nodes and/or arcs that satisfy some condition



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 (1.5)

– New node connected to 3 closest existing nodes, except if

  – Ratio of closest to 2^{nd} and 3^{rd} closest < threshold (0.1)

  – Distance shorter using other connector and graph