21. Set cover and TSP

- Set covering
- Cutting problems and column generation
- Traveling salesman problem

Set covering

- We are given a set of objects $M = \{1, 2, \ldots, m\}$.
- We are also given a set S of subsets of M. Example:
  - $M = \{1, 2, 3, 4, 5, 6\}$
  - $S = \{\{1, 2\}, \{1, 3, 5\}, \{1, 2, 4, 5\}, \{4, 5\}, \{3, 6\}\}$
- The problem is to choose a set of subsets (from S) so that all of the members of M are covered.
- The set $C = \{\{1, 2\}, \{1, 2, 4, 5\}, \{3, 6\}\}$ is a cover.
- We may be interested in finding a cover of minimum cost. Example: minimize the number of subsets used.
- Let $x_i = 1$ if the i^{th} member of S is used in the cover.

Formulation (of previous example)

Reminder: $S = \{\{1,2\},\{1,3,5\},\{1,2,4,5\},\{4,5\},\{3,6\}\}$

minimize $\quad x_1 + x_2 + x_3 + x_4 + x_5$

subject to

$$
\begin{array}{rcrcrcrcl}
x_1 & + & x_2 & + & x_3 & & & \geq & 1 \\
x_1 & & & + & x_3 & & & \geq & 1 \\
 & & x_2 & & & + & x_5 & \geq & 1 \\
 & & & & x_3 & + & x_4 & \geq & 1 \\
 & & x_2 & + & x_3 & + & x_4 & \geq & 1 \\
 & & & & & & x_5 & \geq & 1
\end{array}
$$

$$x_j \in \{0,1\} \quad \forall j$$

More set covering: Kilroy county

- There are 6 cities in Kilroy County.
- The county must determine where to build fire stations to serve these cities. They want to build the stations in some of the cities, and to build the minimum number of stations needed to ensure that at least one station is within 15 minutes driving time of each city.
- Can we formulate an integer program whose solution gives the minimum number of fire stations and their locations?

Driving distances

	1	**2**	**3**	**4**	**5**	**6**
1	0	10	20	30	30	20
2	10	0	25	35	20	10
3	20	25	0	15	30	20
4	30	35	15	0	15	25
5	30	20	30	15	0	12
6	20	10	20	25	12	0

Model

- set of cities: $M = \{1, 2, 3, 4, 5, 6\}$
- $x_j = 1$ if we build a fire station in city j
- cities within 15 minutes of each city:
  $S = \{\{1, 2\}, \{1, 2, 6\}, \{3, 4\}, \{3, 4, 5\}, \{4, 5, 6\}, \{2, 5, 6\}\}$

Kilroy county model

$$S = \{\{1,2\},\{1,2,6\},\{3,4\},\{3,4,5\},\{4,5,6\},\{2,5,6\}\}$$

$$\begin{aligned}
&\underset{x}{\text{minimize}} && x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \\
&\text{subject to:} && \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \\
& && x_i \in \{0,1\} \quad \forall i
\end{aligned}$$

- Optimal solution: $x_2 = x_4 = 1$ (two fire stations)

Set covering/partitioning/packing

If A is a matrix consisting only of 0's and 1's, then:

- **Set covering:** $\min c^T x : Ax \geq 1, x \in \{0,1\}^n$
  Minimal cover (overlap is allowed)
- **Set partitioning:** $\min c^T x : Ax = 1, x \in \{0,1\}^n$
  Minimal partition (each element is hit exactly once)
- **Set packing:** $\max c^T x : Ax \leq 1, x \in \{0,1\}^n$
  Pack as much as possible (no overlap allowed)

Cutting problem

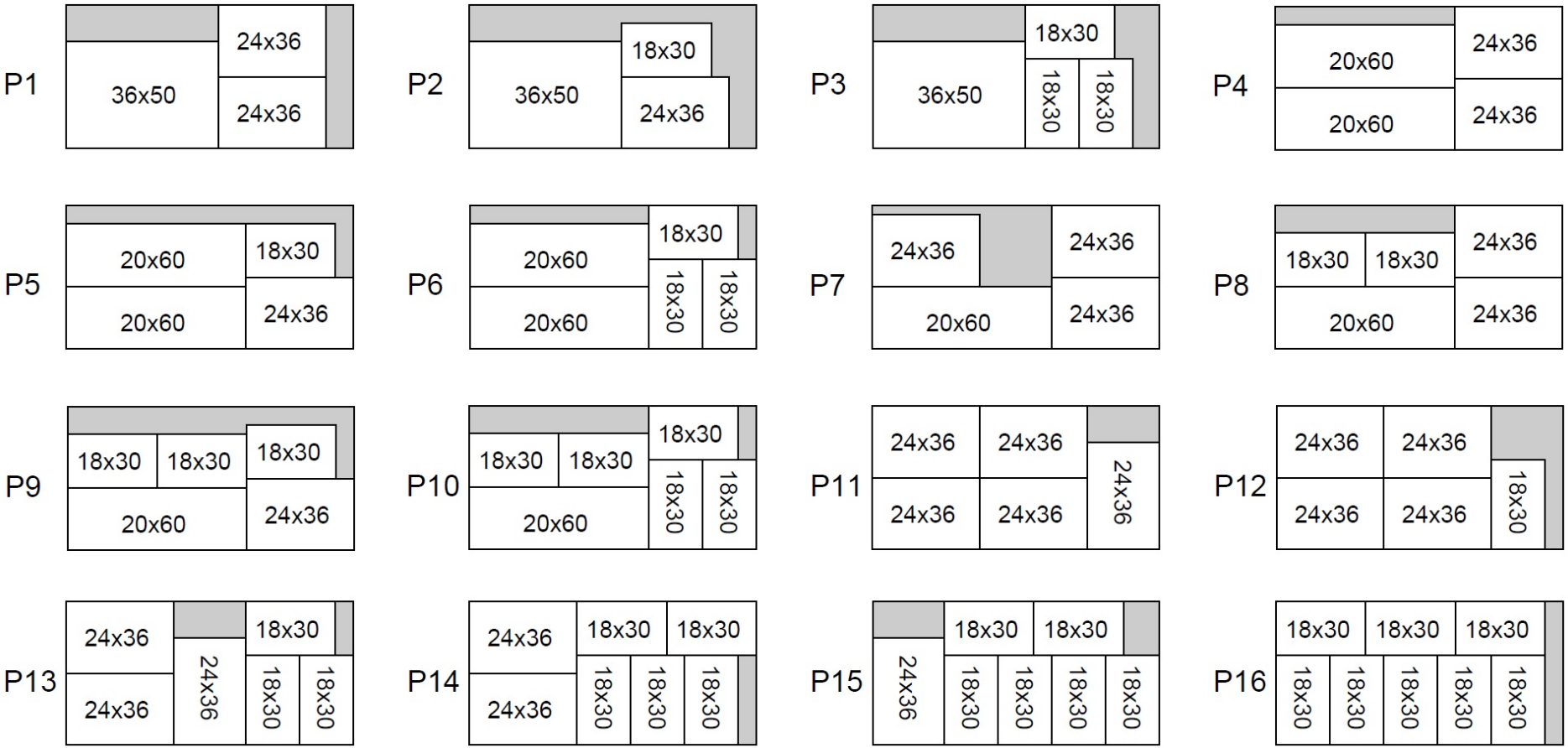
A sheet metal workshop has received the following order:

Dimensions	36×50	24×36	20×60	18×30
Quantity	8	13	5	15

Theses pieces of sheet metal need to be cut from stock sheets, which measure 48×96. How can this order by satisfied by using the least number of large sheets?

- There are relatively few ways to cut a large sheet into smaller sheets.
- Enumerate all the possibilities!
  This is called **column generation**.

Cutting problem



- This enumeration can be tedious, but the problem is easy to solve once we have figured it out.

Cutting problem

Summary of demands: (vector b)

Dimensions	36×50	24×36	20×60	18×30
Quantity	8	13	5	15

Summary of possible patterns, each is a column: (matrix A)

Pattern	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
36×50	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
24×36	2	1	0	2	1	0	3	2	1	0	5	4	3	2	1	0
20×60	0	0	0	2	2	2	1	1	1	1	0	0	0	0	0	0
18×30	0	1	3	0	1	3	0	2	3	5	0	1	3	5	6	8

- x_j is the number of times we use pattern j (integer).
- set covering problem where each pattern can be reused.

Model for cutting problem

$$\begin{aligned} \text{minimize} \quad & x_1 + \cdots + x_{16} \\ \text{subject to:} \quad & Ax \geq b \\ & x_i \in \mathbb{Z}_+ \end{aligned}$$

- Not a feasible method if we have many (100s) of patterns.
- If we use fewer columns (fewer patterns), the solution will still be an upper bound on the optimal one.
- This approach is also useful for problems such as flight crew scheduling: list the possible crew pairings with their associated costs, and solve the set cover problem.

Another cutting example

A plumber stocks standard lengths of pipe, all of length 19 m. An order arrives for:

- 12 lengths of 4m
- 15 lengths of 5m
- 22 lengths of 6m

How should these lengths be cut from standard stock pipes so as to minimize the number of standard pipes used?

- some similarity to a knapsack problem
- how should we model this?

Another cutting example

One possible model:

- Let N be an upper bound on the number of standard pipes. One possible upper bound is $N = 16$ (3 pipes per standard).
- Let $z_j = 1$ if we end up cutting standard pipe j.
- Let x_{ij} = number of pipes of length i to cut from pipe j.
- If $x_{ij} > 0$ then $z_j = 1$ (fixed cost).
- Obvious upper bounds: $x_{1j} \leq 4$, $x_{2j} \leq 3$, and $x_{3j} \leq 3$.

Another cutting example: Model 1

$$\min_{x,z} \quad \sum_{j=1}^{N} z_j \qquad \text{(total number of pipes cut)}$$

$$\text{s.t.} \quad 4x_{1j} + 5x_{2j} + 6x_{3j} \leq 19 \quad \text{for } j = 1, \ldots, N \qquad \text{(cuts are feasible)}$$

$$\sum_{j=1}^{N} x_{1j} \geq 12, \quad \sum_{j=1}^{N} x_{2j} \geq 15, \quad \sum_{j=1}^{N} x_{3j} \geq 22 \qquad \text{(orders must be met)}$$

$$x_{1j} \leq 4z_j, \quad x_{2j} \leq 3z_j, \quad x_{3j} \leq 3z_j \quad \forall j \qquad \text{(fixed cost constraints)}$$

$$x_{ij} \geq 0 \text{ integer}, \quad z_j \in \{0, 1\}$$

- A lot of symmetry. We can break the symmetry by using for example the constraint: $z_1 \geq z_2 \geq \cdots \geq z_N$.
- But should we? Symmetry was good for Sudoku, even though we had to use more variables...

Implementation of Model 1

IJulia code: CuttingPipe.ipynb

Solver comparison:

	Cbc solver	**Gurobi solver**
standard formulation	0.05 sec	0.02 sec
with symmetry broken	0.02 sec	0.0055 sec

I give up...

Using 10 times more demand:

	Cbc solver	**Gurobi solver**
standard formulation	0.6 sec	3.3 sec
with symmetry broken	6.5 sec	19 sec

Moving forward

Downsides of the first model:

- very large space with a lot of redundancy
- solve time scales poorly with problem size
- (scaling the demand should just scale the solution!)

Observations:

- The optimal solution will consist of patterns, such as $(5 + 6 + 6)$ or $(4 + 4 + 5 + 6)$.
- Even if there are many possible patterns, the optimal solution will only use a few different ones.
- Can we take advantage of these facts?

Model 2

- There are only 9 different ways to cut the pipe:

$6+6+6$	$6+6+5$	$6+6+4$
$6+5+5$	$6+5+4+4$	$5+5+5+4$
$5+5+4+4$	$5+4+4+4$	$4+4+4+4$

- Each cut pattern is a column of this matrix:

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 2 & 1 & 2 & 3 & 4 \\ 0 & 1 & 0 & 2 & 1 & 2 & 2 & 1 & 0 \\ 3 & 2 & 2 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\underset{x}{\text{minimize}} \quad \sum_{j=1}^{9} x_j \qquad \text{(how many times each pattern is used)}$$

$$\text{subject to:} \quad Ax \geq \begin{bmatrix} 12 \\ 15 \\ 22 \end{bmatrix} \qquad \text{(orders must be met)}$$

$$x_j \geq 0 \text{ integer}$$

Implementation of Model 2

IJulia code: CuttingPipe.ipynb

Solver comparison:

Much better!

	Cbc solver	**Gurobi solver**
original problem	0.003 sec	0.003 sec
scaled by 10	0.003 sec	0.003 sec
scaled by 100	0.003 sec	0.003 sec
scaled by 1,000	0.003 sec	0.003 sec
scaled by 1,000,000	0.003 sec	0.003 sec

Larger instances

- If we had much larger pieces of standard pipe, there would be a very large number of possible patterns.
- No longer feasible to enumerate them all
- We shouldn't have to! Each column represents a potential vertex. Most will lie inside the feasible polyhedron.
- Column generation strategy:
  - Pick a small number of columns to start
  - Solve the LP relaxation
  - Use the dual to help pick a new column
  - Add the column to the A matrix and repeat

Column generation

1. Start with columns $A = \begin{bmatrix} a_1, \ldots, a_k \end{bmatrix}$
2. Solve MIP: minimize $\sum_i x_i$ subject to $Ax \geq b$.
3. Obtain dual variable $\lambda_\star$ to the $Ax \geq b$ constraint.
4. If we add a new column $\tilde{a}$, cost will drop by $\tilde{a}^\mathsf{T} \lambda_\star$.
   - Suppose $\tilde{a}^\mathsf{T} = \begin{bmatrix} p & q & r \end{bmatrix}^\mathsf{T}$ is the new column.
   - Solve MIP: maximize $\tilde{a}^\mathsf{T} \lambda_\star$ subject to $4p + 5q + 6r \leq 19$.
   - Add new column to A: $a_{k+1} = \tilde{a}_\star$.
5. Repeat from step 2.

Column generation can be an effective strategy when it's impossible to enumerate all possible columns. Only columns that are deemed likely to help reduce the cost are added.

Traveling salesman

A traveling salesman must visit a set of cities N and incur a minimal total cost. The salesman starts and ends at home.

- c_{ij} is the cost of traveling from city i to city j.
- This is perhaps the most famous combinatorial optimization problem. There are lots of applications!
- Obvious choice of decision variables:

$$x_{ij} = \begin{cases} 1 & \text{we travel from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

Possible TSP model

$$
\begin{aligned}
\underset{x}{\text{minimize}} \quad & \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} & & \\
\text{subject to:} \quad & \sum_{i \in N} x_{ij} = 1 & \forall j & \quad \text{(one in-edge)} \\
& \sum_{j \in N} x_{ij} = 1 & \forall i & \quad \text{(one out-edge)} \\
& x_{ii} = 0 & \forall i & \quad \text{(no self-loops)}
\end{aligned}
$$

Will this do the trick? (No! this is a standard min-cost flow problem with an exact LP relaxation. The TSP is an NP-complete problem so surely we can't solve it this way...)

TSP example

Distances betweeen 10 major airports in the US (in miles):

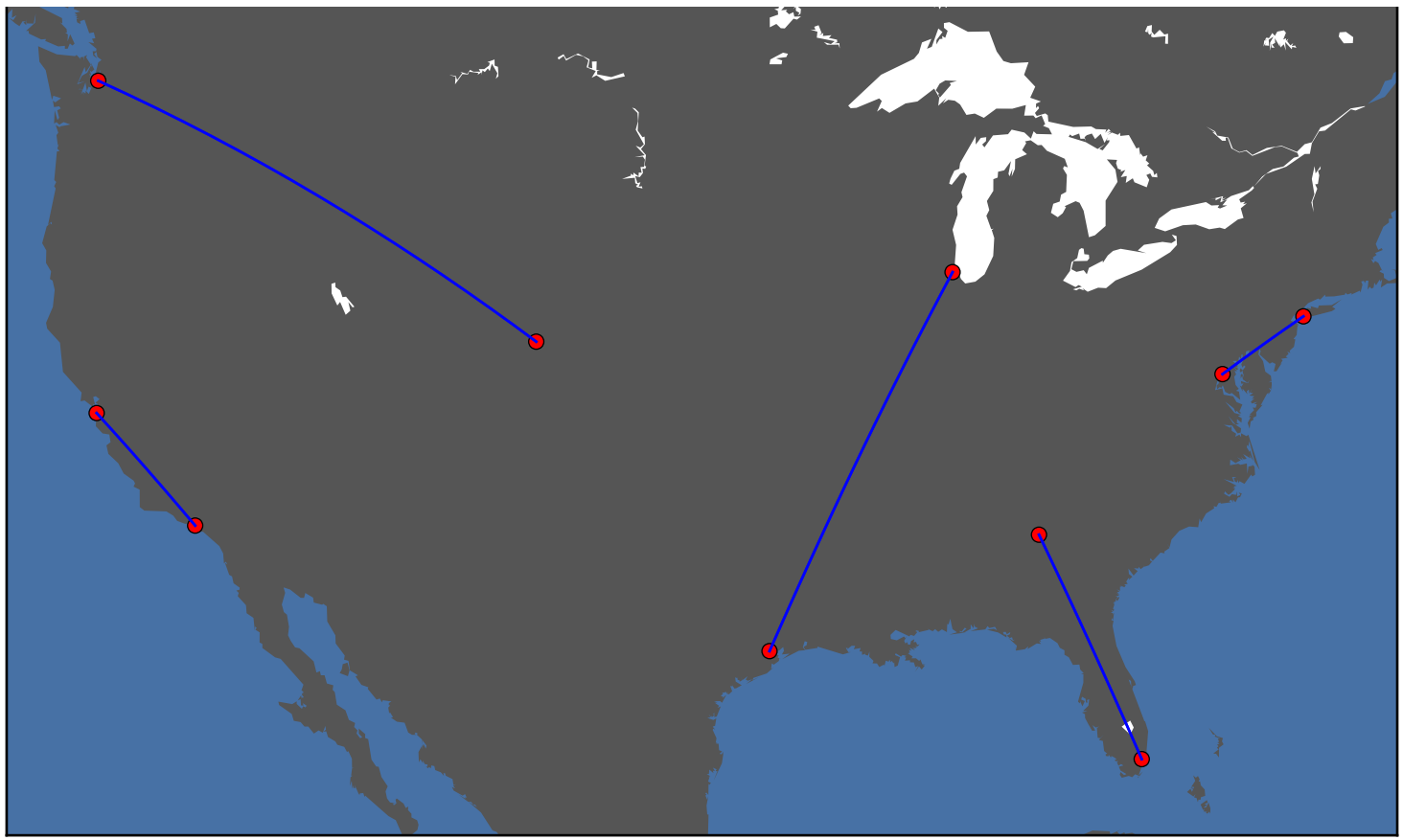
	ATL	ORD	DEN	HOU	LAX	MIA	JFK	SFO	SEA	DCA
ATL	0	587	1212	701	1936	604	748	2139	2182	543
ORD	587	0	920	940	1745	1188	713	1858	1737	597
DEN	1212	920	0	879	831	1726	1631	949	1021	1494
HOU	701	940	879	0	1379	968	1420	1645	1891	1220
LAX	1936	1745	831	1379	0	2339	2451	347	959	2300
MIA	604	1188	1726	968	2339	0	1092	2594	2734	923
JFK	748	713	1631	1420	2451	1092	0	2571	2408	205
SFO	2139	1858	949	1645	347	2594	2571	0	678	2442
SEA	2182	1737	1021	1891	959	2734	2408	678	0	2329
DCA	543	597	1494	1220	2300	923	205	2442	2329	0

The model from the previous slide is an assignment problem!
(think back to the swim relay example)

Julia code: TSP.ipynb

TSP example

- The cheapest assignment leads to a fragmented solution: {ATL,MIA},{LAX,SFO},{SEA,DEN},{HOU,ORD},{DCA,JFK}



looks like the relaxation isn't what we wanted!

TSP example

- The cheapest assignment leads to a fragmented solution: {ATL,MIA},{LAX,SFO},{SEA,DEN},{HOU,ORD},{DCA,JFK}
- How can we exclude solutions involving subtours?
- To exclude for example the subtour $\{1 \to 2 \to 3 \to 1\}$, we musn't allow all three of those edges. We can accomplish this with the constraint: $x_{12} + x_{23} + x_{31} \leq 2$.
- In general, we would need to add the constraints:

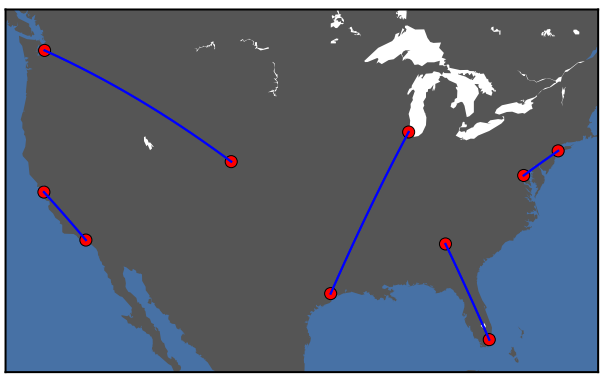
$$\sum_{(i,j) \in S} x_{ij} \leq |S| - 1 \quad \text{for all subtours } S$$

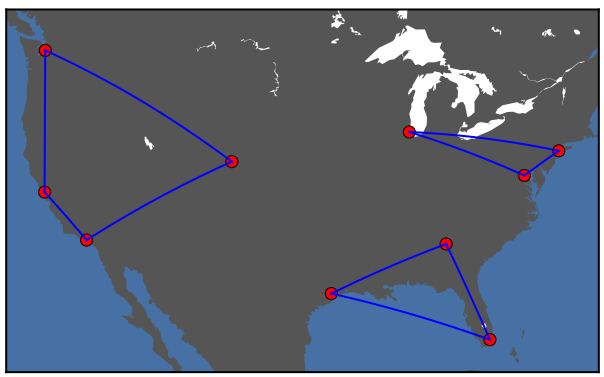
Bad news

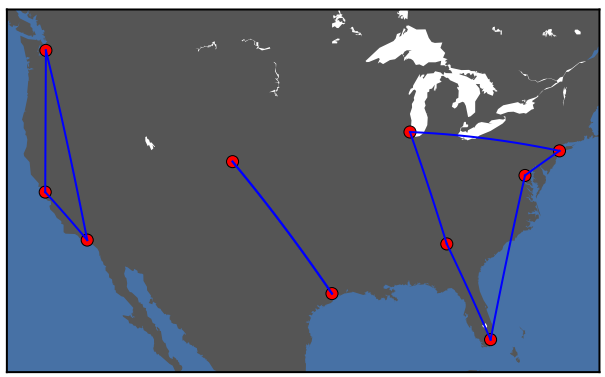
- Exponentially many "subtour elimination" constraints
- LP relaxation ceases to be exact

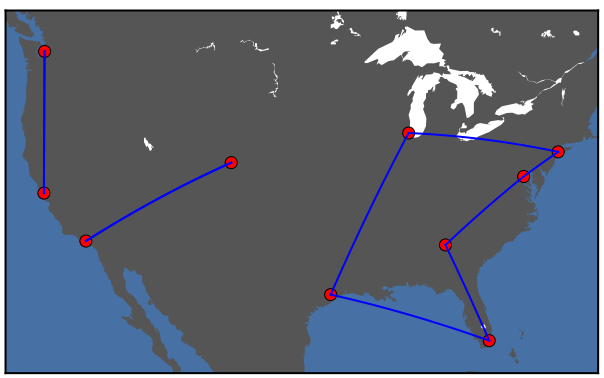
TSP subtour heuristic

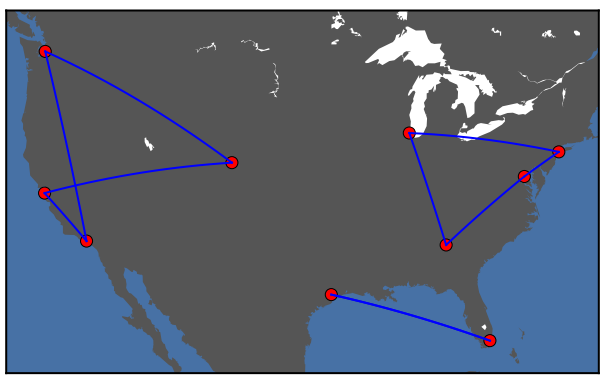
1. Solve the TSP without subtour exclusions.
2. If the optimal solution has subtours, add the associated subtour elimination constraints and solve again.
3. Repeat until the optimal solution has no subtours. If we're lucky, we get the optimal solution in just a few rounds.

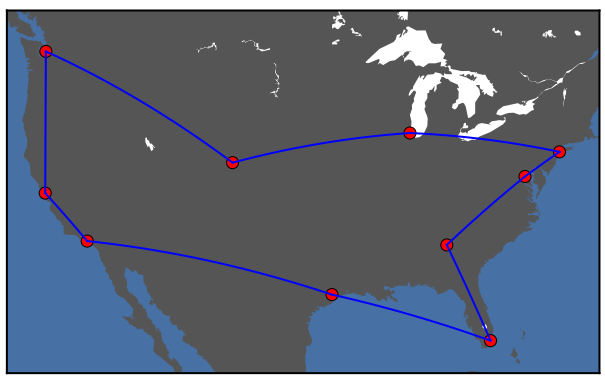

iteration 1


iteration 2


iteration 3


iteration 4


iteration 5


iteration 6

TSP direct formulation as a MIP

Idea: add variable u_i for each node $i \in N$. This will be the relative position of node i in the optimal tour.

- Due to Miller, Tucker, and Zemlin (1960)
- Require that $1 \leq u_i \leq n = |N|$.
- Let's call node 1 the start of our tour. Add the constraint: whenever $x_{ij} = 1$, we have $u_j \geq u_i + 1$.
  This should hold for all $i, j \in N$ with $j \neq 1$.
- It's a logic constraint: $x_{ij} = 1 \implies u_i - u_j \leq -1$.
  Equivalent to:
  $$u_i - u_j + n x_{ij} \leq n - 1$$
- A subtour of length k would imply $kn \leq k(n-1)$ (false!)

Miller–Tucker–Zemlin formulation

$$
\begin{aligned}
\underset{x,u}{\text{minimize}} \quad & \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ij} & \\
\text{subject to:} \quad & \sum_{i \in N} x_{ij} = 1 & \forall j \in N \\
& \sum_{j \in N} x_{ij} = 1 & \forall i \in N \\
& x_{ii} = 0 & \forall i \in N \\
& 1 \leq u_i \leq n & \forall i \in N \\
& u_i - u_j + n x_{ij} \leq n - 1 & \forall i, \forall j \neq 1
\end{aligned}
$$

- The $1 \leq u_i \leq n$ constraints are not actually needed.
- It's exact! Not exponentially many constraints!
- Still an MIP, only practical for small n.

TSP wrap-up

- We can formulate TSP as an MIP in a couple different ways
- Subtours increase the complexity substantially
  - eliminate subtours adaptively (as in column generation)
  - solve larger MIP via MTZ formulation
- Many other techniques available for solving large instances of TSP. e.g. branch and cut. More on this later!