Traveling salesman problem (TSP) ILP

Ben Rosenberg

April 26, 2026

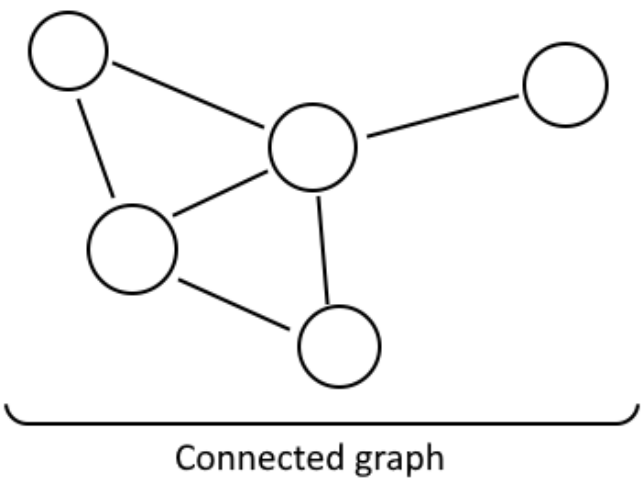
Overview

- Background
- Definition of TSP
- Initial ILP model
- Subtour elimination constraints
- Constraint generation
- OR-Tools model
- TSP approximation algorithms/heuristics

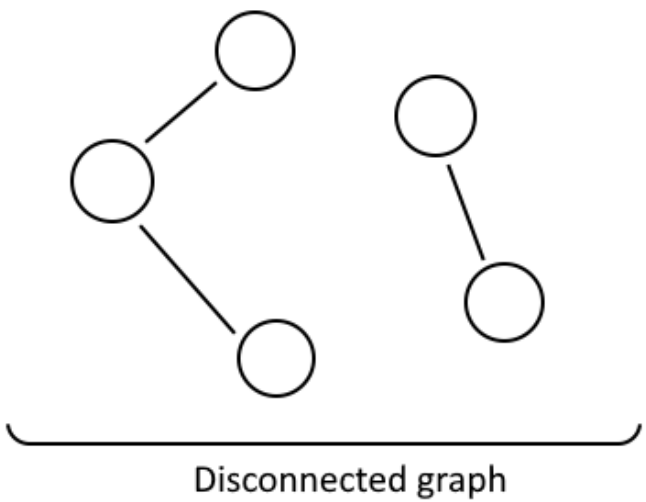
Background

Let $G = (V, E)$ be a graph.

A graph G is *connected* if for every pair of vertices $(u, v) \in V \times V$, there is a path $u, w_1, w_2, \ldots, w_n, v$ from u to v, where all of $(u, w_1), (w_1, w_2), \ldots, (w_n, v)$ are edges in E.



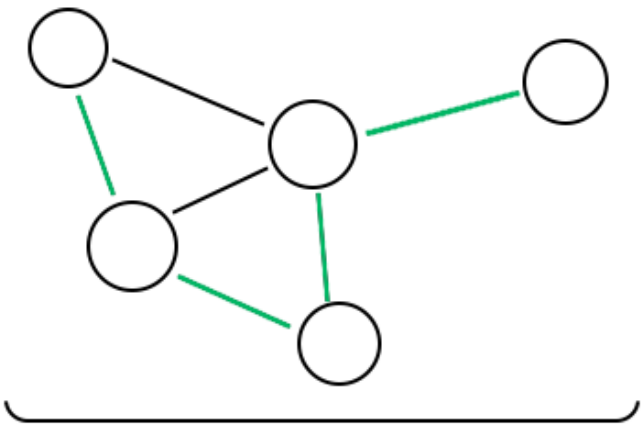
Connected graph



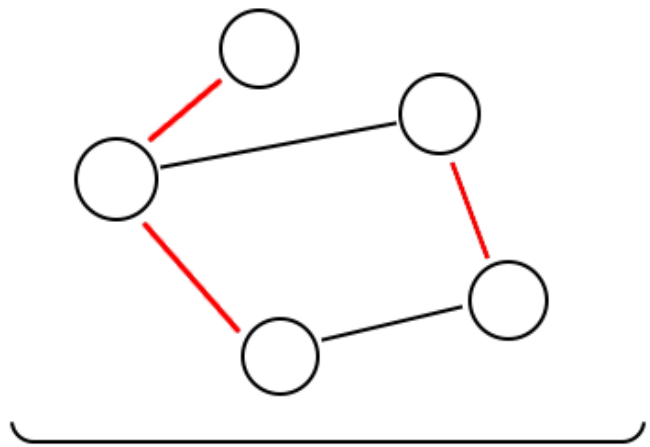
Disconnected graph

Background (cont.)

A *path* in G is a list of edges
$e_1 = (u_1, v_1), e_2 = (u_2, v_2), \ldots, e_n = (u_n, v_n)$ in E where
$v_1 = u_2, v_2 = u_3, \ldots, v_{n-1} = u_n$.



A path (green)

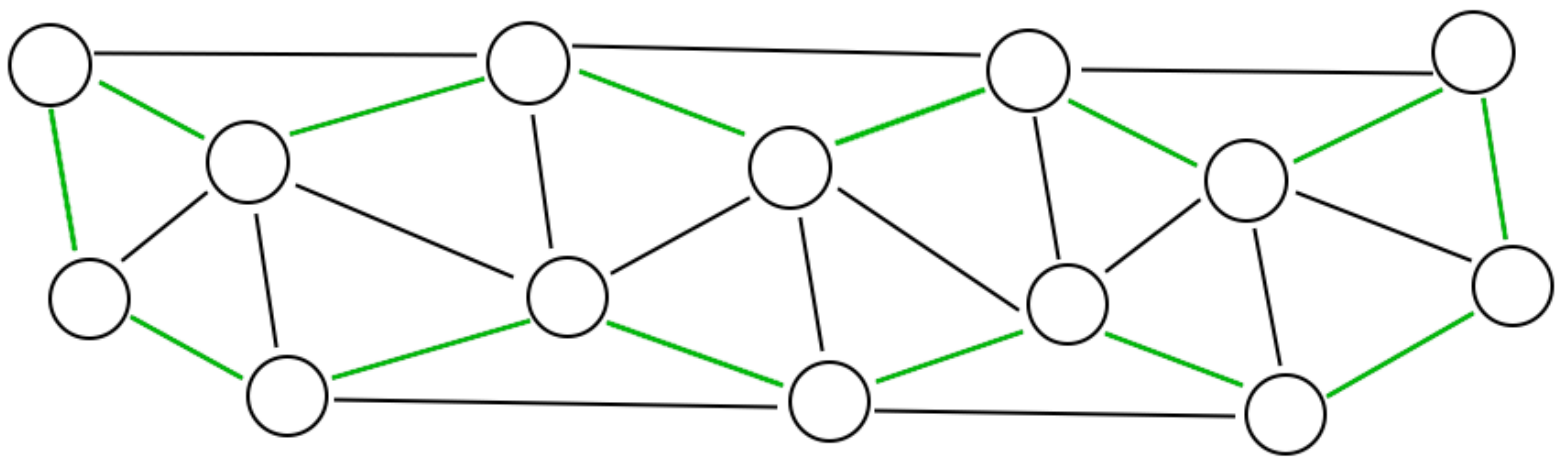


Not a path (red)

Background (cont.)

A *cycle* in G is a path $w_1, w_2, \ldots, w_n$ where $w_1 = w_n$.

A *Hamiltonian cycle* is a cycle that visits each vertex in V exactly once.



TSP

In traveling salesman problem (TSP), we are given a (connected, undirected) graph $G = (V, E)$, with edge costs given by a function $C : E \to \mathbb{R}$.

The objective is to find a Hamiltonian cycle of minimum cost.

Initial ILP model

Decision variables: binary x_{v_1,v_2} where $x_{v_1,v_2} = 1$ if the edge (v_1, v_2) is included in the solution, and 0 otherwise

Constraints: All vertices must be included in the circuit exactly once:

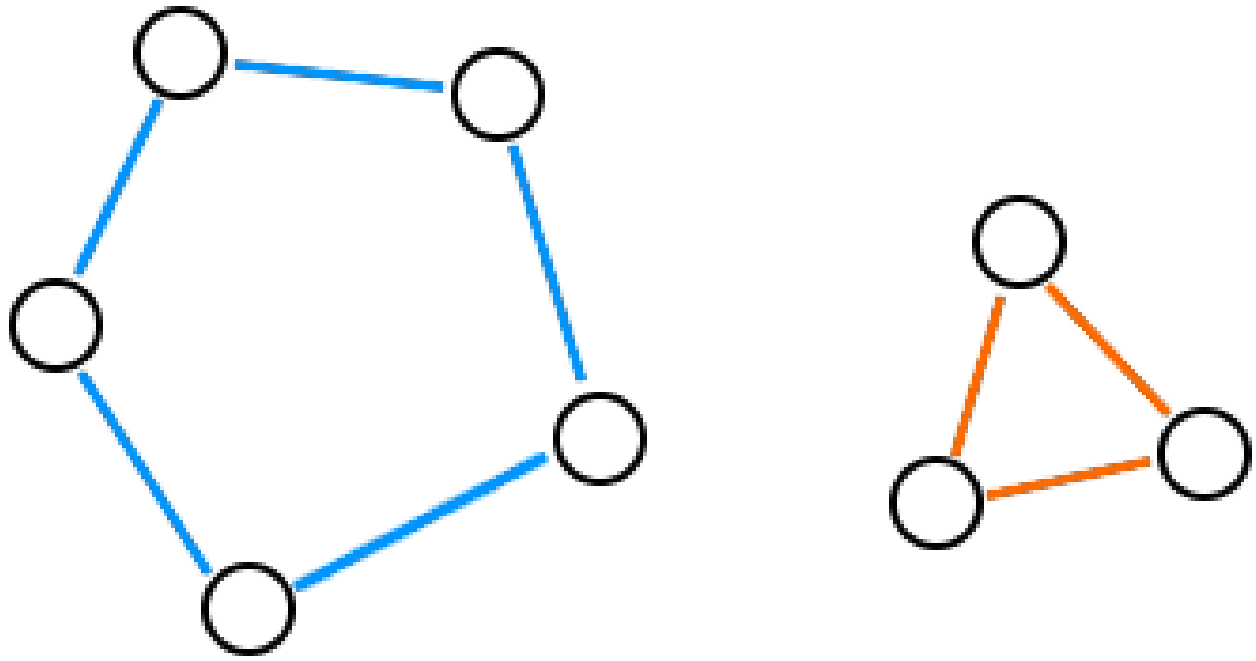
$$\sum_{v \in V} x_{u,v} = 1 \quad \forall u \in V, u \neq v$$

$$\sum_{v \in V} x_{v,u} = 1 \quad \forall u \in V, u \neq v$$

Objective: Minimize total cost:

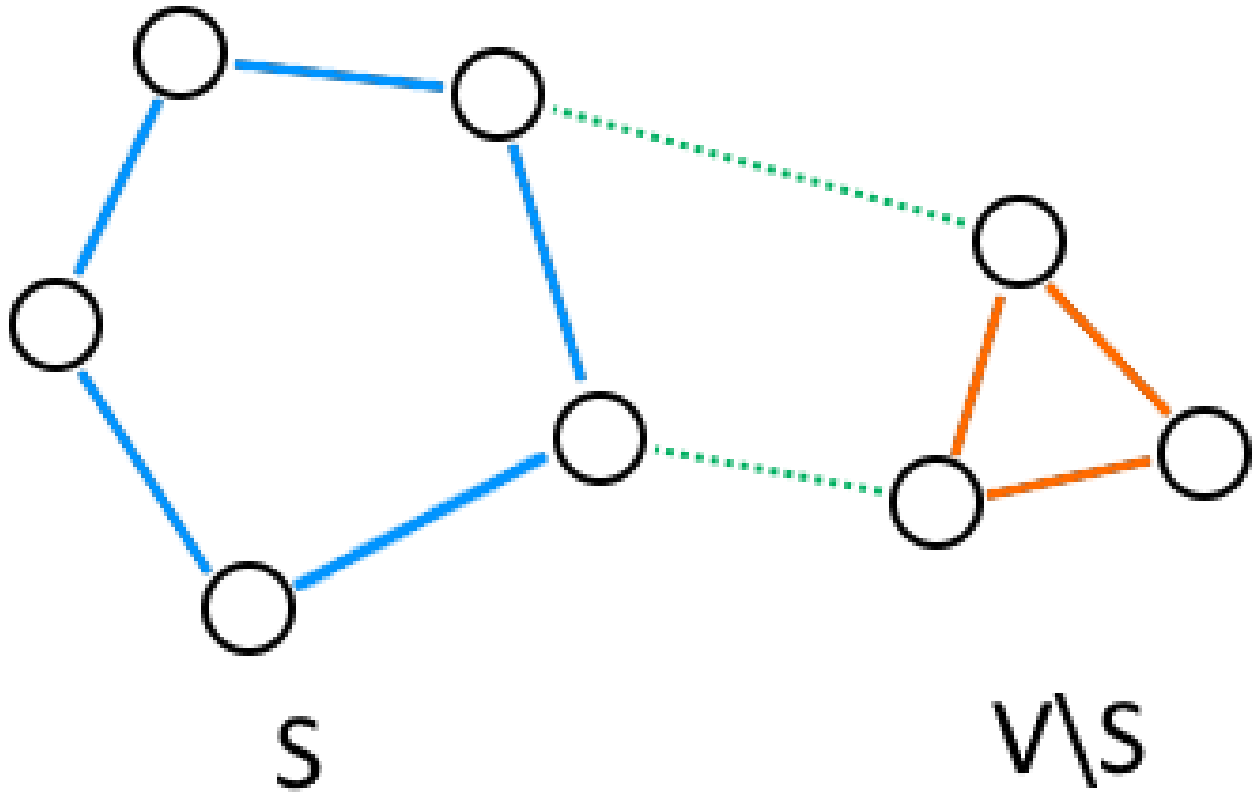
$$\min \sum_{\{u,v\} \in E} C(\{u, v\}) \cdot x_{u,v}$$

Oops!



Subtour elimination constraints

We need to ensure that for every subset S of V, S is connected to $V \backslash S$.



Subtour elimination constraints (cont.)

Subtour elimination constraints:

$$\sum_{u \in S} \sum_{v \in S,\, u \neq v} x_{u,v} \leq |S| - 1 \quad \forall S \in \mathcal{P}(V) : 2 \leq |S| \leq |V| - 1$$

Subtour elimination constraints (cont.)

How many of these constraints are there, if we have 50 vertices?

$$
\begin{aligned}
& 2^{50} - \binom{50}{50} - \binom{50}{49} - \binom{50}{2} - \binom{50}{1} - \binom{50}{0} \\
& = 2^{50} - 1 - 50 - 50 - 1225 - 50 - 1 \\
& \approx 2^{50} \\
& > 1 \text{ quadrillion}
\end{aligned}
$$

Constraint generation

Solution: avoid enforcing all constraints at once.

Process (constraint generation):

1. Try solving the TSP without any subtour elimination constraints
2. Detect subtours
3. Add constraint to prevent that subtour
4. Repeat until no subtours detected

In practice, this works pretty well. Most subsets of V wouldn't yield good solutions as subtours anyway, so we won't need to add a very large number of additional constraints before we find a valid solution.

OR-Tools model

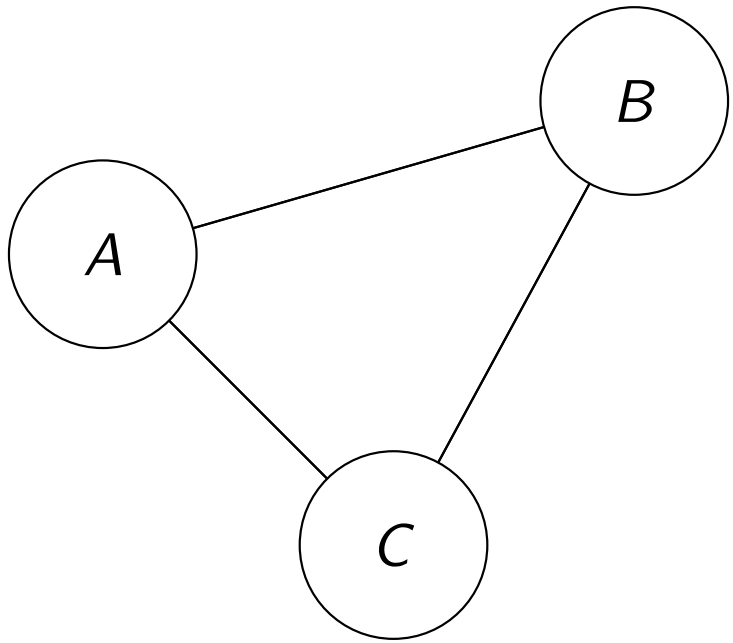
(Demo: OR-Tools TSP model)

Metric TSP

There are other TSP categories besides the one we examined.

One popular one is the metric TSP, also known as the Euclidean TSP, in which the triangle equality holds.

For the below instance to be metric, the cost of going from A to B should be lower than the cost of going from A to C to B, and vice-versa with A to C and B to C.



Approximation algorithms and heuristics

The difficulty of the TSP to solve perfectly has led to the creation of approximation algorithms that are less accurate but run in polynomial time.

Some approximation algorithms:

- Double-tree algorithm
  - Assumes metric TSP
  - Uses MST (minimum spanning tree) as a startpoint
  - Achieves 2x bound on optimal solution
  - Runs in $O(n^2)$
- Christofides algorithm
  - Assumes metric TSP
  - Uses MST (minimum spanning tree) as a startpoint
  - Achieves 1.5x bound on optimal solution
  - Runs in $O(n^3)$

Approximation algorithms and heuristics (cont.)

Some heuristics:

- Nearest neighbor
  - Start at some node, then always choose the node that is closest
  - Not very good in practice, doesn't even have a constant approximation ratio
- 2-Opt
  - Iteratively choose two edges in a TSP solution, and see if swapping them improves the result
  - No approximation ratio guarantee for this either, although can be applied on top of existing solutions, and can choose how long to run iterative checks

Approximation algorithms and heuristics (cont.)

In addition to these heuristics and algorithms, there are meta-heuristics that can generally be used to solve problems.

One example is simulated annealing, which can be combined with 2-opt to strike a balance between the perfection of an ILP solution and the running time of an approximation algorithm like Christofides.

(Show writeup on TSP solution methods.)