Mathematics for Decisions

Integer Linear Programming: TSP and Branch-and-Bound

Romeo Rizzi, Alice Raffaele, and Matteo Zavatteri

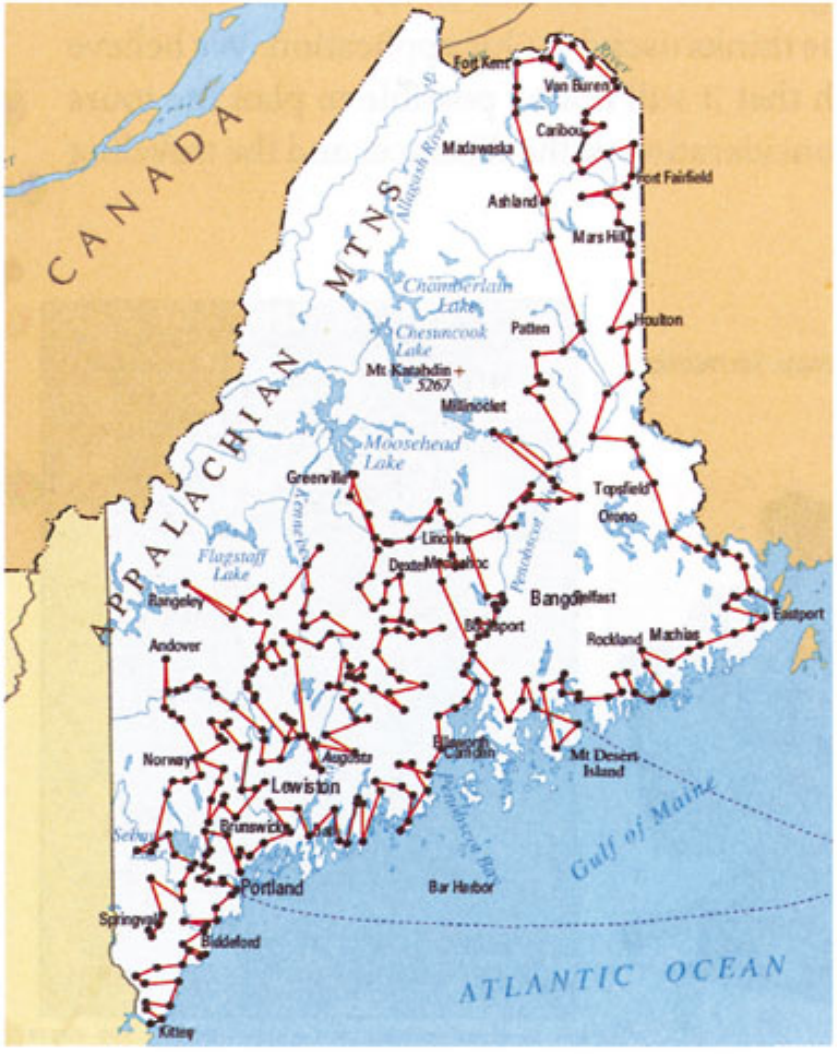
University of Verona

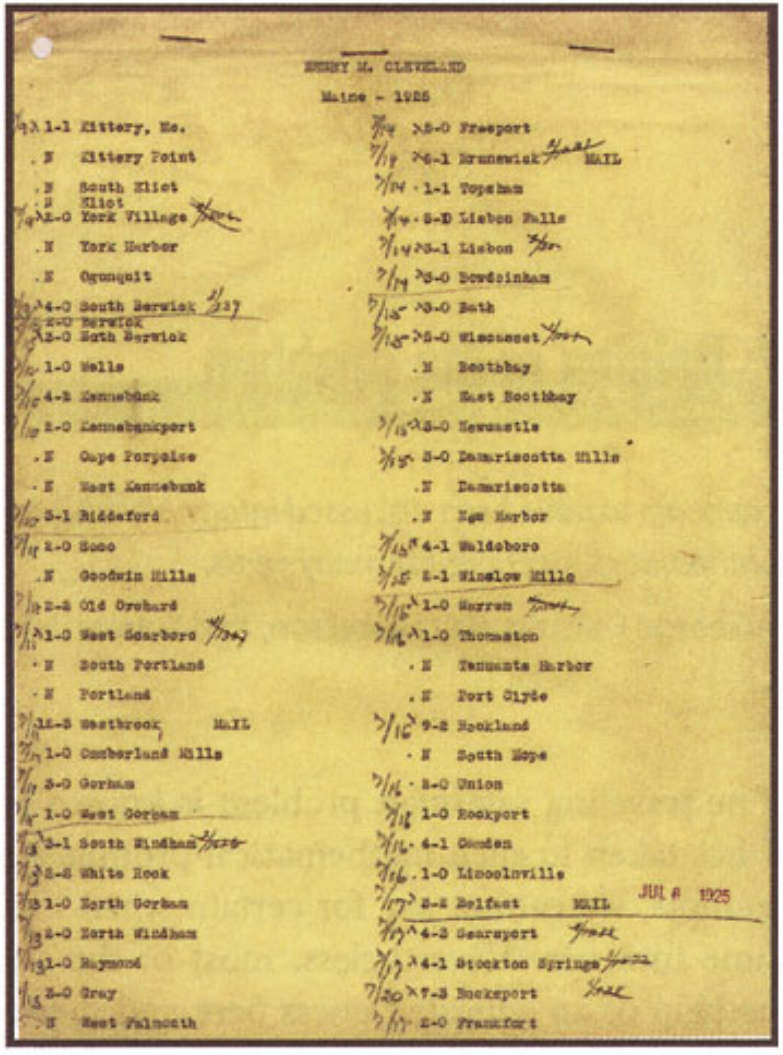
romeo.rizzi@univr.it, alice.raffaele@univr.it, matteo.zavatteri@univr.it

December 16, 2021

Overview

Mathematical road trips in 1925





HENRY M. CLEVELAND

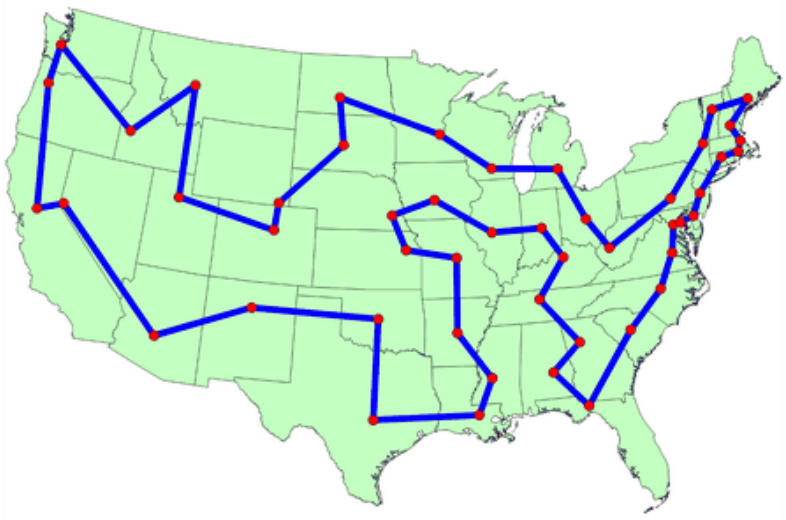
Maine - 1925

1-1	Kittery, Me.	8-0	Freeport
.N	Kittery Point	6-1	Brunswick MAIL
.N	South Eliot	1-1	Topsham
.N	Eliot		
2-0	York Village	8-0	Lisbon Falls
.N	York Harbor	3-1	Lisbon
.N	Ogunquit	3-0	Bowdoinham
4-0	South Berwick	3-0	Bath
2-0	Berwick		
2-0	North Berwick	8-0	Wiscasset
1-0	Wells	.N	Boothbay
4-2	Kennebunk	.N	East Boothbay
2-0	Kennebunkport	3-0	Newcastle
.N	Cape Porpoise	3-0	Damariscotta Mills
.N	West Kennebunk	.N	Damariscotta
3-1	Biddeford	.N	Egg Harbor
2-0	Saco	4-1	Waldoboro
.N	Goodwin Mills	2-1	Winslow Mills
2-2	Old Orchard	1-0	Warren
1-0	West Scarboro	1-0	Thomaston
.N	South Portland	.N	Tenants Harbor
.N	Portland	.N	Port Clyde
16-3	Westbrook MAIL	9-2	Rockland
1-0	Cumberland Mills	.N	South Hope
3-0	Gorham	2-0	Union
1-0	West Gorham	1-0	Rockport
3-1	South Windham	4-1	Camden
8-8	White Rock	1-0	Lincolnville
1-0	North Gorham	2-2	Belfast MAIL JUL 8 1925
2-0	North Windham	4-2	Searsport
1-0	Raymond	4-1	Stockton Springs
2-0	Gray	7-3	Rockport
.N	West Falmouth	2-0	Frankfort

"Dear Sirs
My route list is balled up the worst I ever saw. Will take half as long again to work it as last year. [...] I wish you would send me my old list 1924 from Dexter on as it is much better than this. I don't see how you could break it out as you did especially from Albion to Madison would be jumping all over the map. This section I changed. The river from Bangor down has no bridge and you have those towns down as if I could cross it anywhere. Last year's list was made out the best of any one and I can't see the object of changing it over. I think I have made myself plain.

Yours truly, H. M. Cleveland"

The Traveling Salesman Problem



"It is shown that a certain tour of 49 cities, one in each of the 48 states and Washington DC, has the shortest road distance" (Dantzig, Fulkerson and Johnson, 1954)

Problem definition: a salesman must visit n cities exactly once and must return to the original point of departure; the distance (or cost) between each pair of cities (i, j) is known and denoted by c_{ij}. Find the salesman's shortest route by computing a minimum-length cycle.

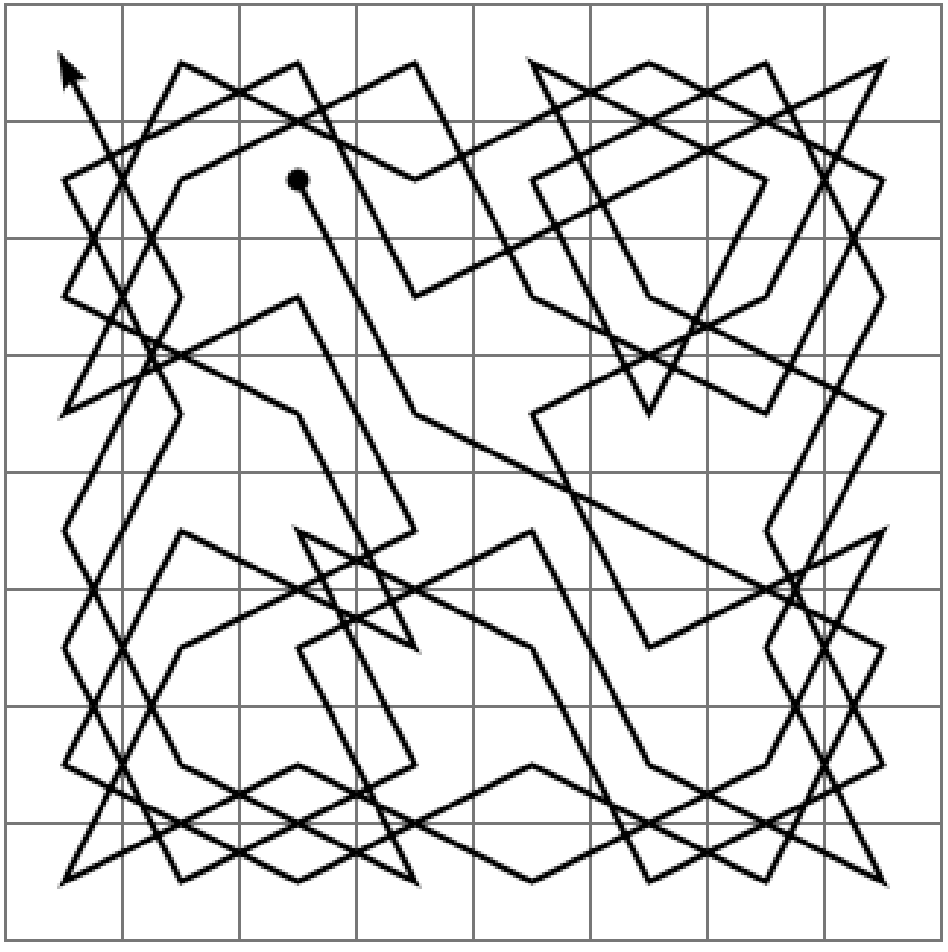
- Since every city must be visited, the solution will be a cyclic permutation of all cities.
- Given n cities, the number of possible solutions is exactly $(n-1)!/2$, when distances are symmetric $\rightarrow$ It is not so easy to check all possible solutions, even with a computer, especially when solving large instances:

No. cities	No. tours	Time
5	12	12 microsecs
8	2520	2.5 millisecs
10	181,440	0.18 secs
12	19,958,400	20 secs
15	87,178,291,200	12.1 hours
18	177,843,714,048,000	5.64 years
20	60,822,550,204,416,000	1927 years

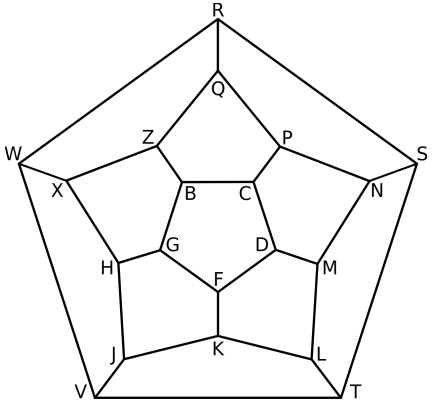
- In 1972, Karp proved the problem to be NP-Hard: this dooms exact methods to take exponential time in the worst case; anyway, there are many heuristics that quickly yield feasible solutions of reasonable quality. Moreover, the metric case (i.e., when distances satisfy the triangle inequality) allows for approximation algorithms.

Some history

- 1766: Euler studied the **Knight's tour** problem

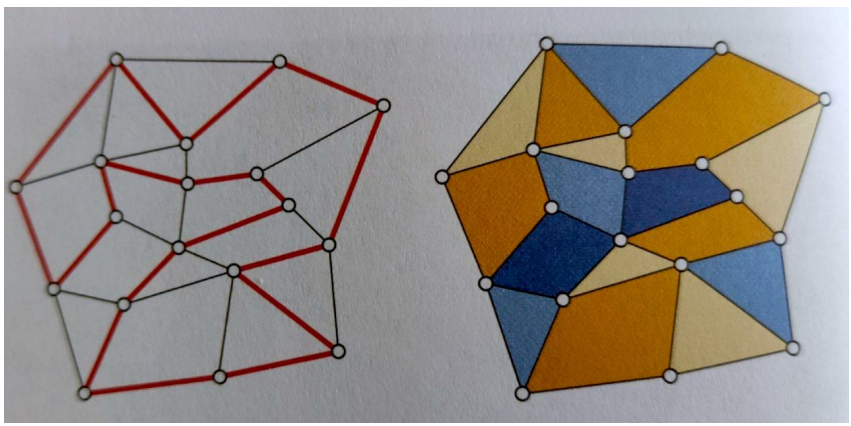


- 1856: Kirkman studied **circuits in polyhedra**, suggesting that there may always be a circuit passing through each vertex only once. He later found an example of a polyhedron without any.
- 1856-1859: Hamilton developed the **Icosian Game**:



"I have found that some young persons have been much amused by trying a new mathematial game which the Icosian furnishes, one person sticking five pins in any five consecutive points, such as abcde or abcde′, and the other player then aiming to insert, which by theory in this letter can always be done, fifteen other pins in cyclical succession, so as to cover all the other points, and to end in immediate proximity to the pin wherewith his antagonist has begun."

- 1884: Tait offered an elegant proof of the **4-color conjecture** essentially by assuming that Kirkman's conjecture was true for three-regular three-connected graphs. To see the connection between circuits and map coloring:
  - Think of the boundaries of the regions in a map as the edges of a graph, with the intersection points as vertices.
  - A Hamiltonian circuit through this boundary graph gives a way to color the map: such a circuit does not cross itself, so it has an inside and an outside; the border edges on the inside, that are not part of the circuit, cut across the inner area.
  - We can color these inner regions with two colors, switching every time we cross one of the non-circuit edges.



- 1930: Menger was the first to talk about **Hamiltonian paths** and to introduce travel costs, and thus to study the general formulation of the problem:

  "*We use the term 'Messenger problem' (because this question is faced in practice by every postman, and, by the way, also by many travelers) for the task, given a finite number of points with known pairwise distances, to find the shortest path connecting the points.*"

  The Messenger problem is recorded, in German, as part of the published documentation of the Vienna Mathematics Colloquium.
- 1937: Flood tried to obtain near-optimal solutions in reference to the **routing of school buses**, after hearing the problem in a seminar talk by Whitney at Princeton in 1934.

- 1946: Tutte offered a **counterexample** to Tait's conjecture on circuits in three-regular three-connected graphs.
- 1949: The **name** *Traveling Salesman Problem* appeared for the first time in Julia Robinson's paper "*On the Hamiltonian Game (A Traveling Salesman Problem)*". It is unknown who actually coined the name.
- 1954: Dantzig, Fulkerson and Johnson offered the first ILP formulation for the **Symmetric TSP**, and then also extended it to the **Asymmetric** version.

1954: Drummer's Delight, *Newsweek*

SCIENCE

Newsweek—Bensi

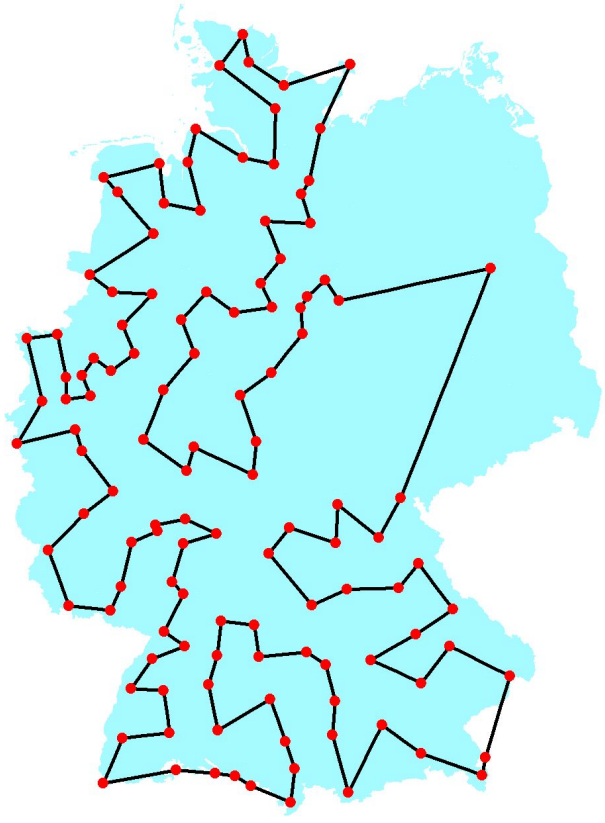
FINDING the shortest route for a traveling salesman—starting from a given city, visiting each of a series of other cities, and then returning to his original point of departure—is more than an after-dinner teaser. For years it has baffled not only goods- and salesmen-routing businessmen but mathematicians as well. If a drummer visits 50 cities, for example, he has 10^{62} (62 zeros) possible itineraries. No electronic computer in existence could sort out such a large number of routes and find the shortest.

Three Rand Corp. mathematicians, using Rand McNally road-map distances between the District of Columbia and major cities in each of the 48 states, have finally produced a solution (see above). By an ingenious application of linear programming—a mathematical tool recently used to solve production-scheduling problems—it took only a few weeks for the California experts to calculate "by hand" the shortest route to cover the 49 cities: 12,345 miles.

1962: Procter & Gamble

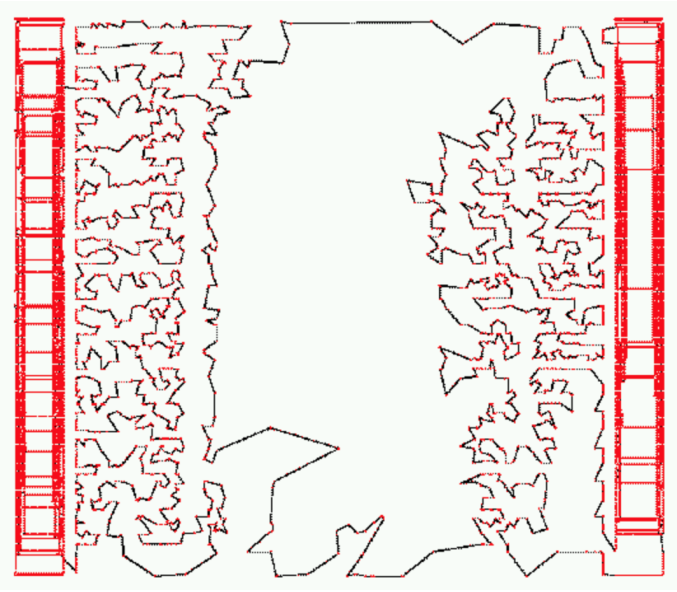
- Advertising campaign which featured a contest with a $10,000 prize.
- From the official rules: "*Imagine that Toddy and Muldoon want to drive around the country and visit each of the 33 locations represented by dots on the contest map, and that in doing so, thy want to travel the shortest possible route. You should plan a route for them from location to location which will result in the shortest total mileage from Chicago, Illinois back to Chicago, Illinois.*"

1977: Groetschel



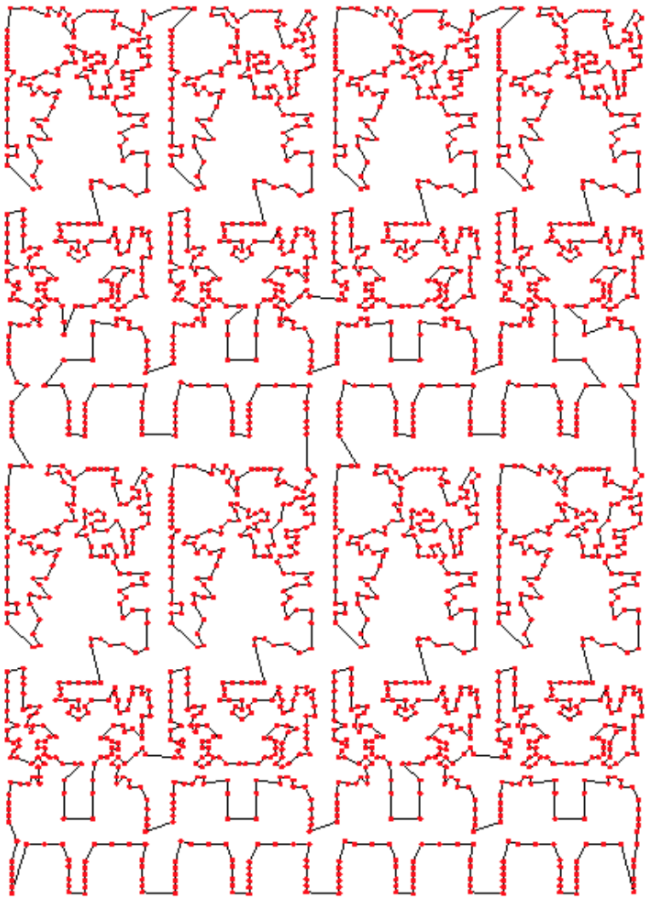
Groetschel found the optimal tour of 120 cities from what was then West Germany.

1987: Padberg and Rinaldi



Padberg and Rinaldi found the optimal tour through a layout of 2,392 points obtained from Tektronics Incorporated.

1994: Applegate, Bixby, Chvátal and Cook

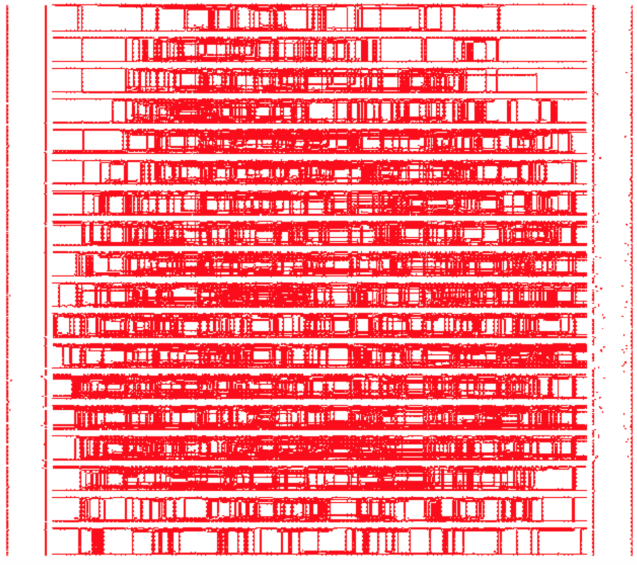


Applegate, Bixby, Chvátal, and Cook found the optimal tour for a 7,397-city TSP that arose in a programmable logic array application at AT&T Bell Laboratories.

2004: Applegate, Bixby, Chvátal, Cook and Helsgaun

Applegate, Bixby, Chvátal, Cook, and Helsgaun found the optimal tour of 24,978 cities in Sweden.

2006: Cook et al.



In 2006, Cook and others computed an optimal tour through an 85,900-city instance given by a microchip layout problem, currently the largest solved TSPLIB (i.e., a library of sample instances for the TSP) instance. For many other instances with millions of cities, solutions can be found that are guaranteed to be within 2–3% of an optimal tour.

Just a few applications

- Planning (e.g., pickups and deliveries, book tours, flight times)
- Mapping genomes
- Aiming telescopes, X-rays and lasers (e.g., finding planets, X-ray cristallography, lasers for crystal art)
- Guiding industrial machines (e.g., drilling circuit boards, soldering a printed circuit board, customizing computer chips)
- Scheduling jobs (e.g., production of smart cards)

How to tackle ILP problems

- **Heuristic algorithms**:
  - Approaches that yield an output within the given time limit. To compare different methods, one can check the quality of the solutions returned and the computational resources used.
  - In the real world, the time to design and implement a heuristic can also make the difference.
- **Approximation algorithms**:
  - Polynomial-time algorithms that are guaranteed to return feasible solutions whose values are within proven bounds from the optimum of the objective function.
- **Exact algorithms**:
  - Methods that always return an optimal feasible solution (if any exists).

A few heuristic algorithms

- **Constructive algorithms** (e.g., various greedy and randomized adaptive search approaches such as farthest insertion, random insertion and nearest neighbor): these generate feasible solutions starting from the instance. Example of *Nearest neighbor*: https://www.youtube.com/watch?v=E85l6euMsd0
- **Local Search** (e.g., 2-opt, k-opt, Variable Neighborhood Search, Exponential Neighborhood Search). Example of *2-opt*: https://www.youtube.com/watch?v=3zSmjkpvcgw)
- **Meta-heuristics** (e.g., Simulated Annealing, Tabu Search, Genetic, Memetic and Ant Colony algorithms, etc.)

Approximation algorithms

- Given a *minimum* problem, let x^* and x^H be the optimal solution and the solution obtained with the approximation algorithm, respectively.
- We can say that $\frac{x^H - x^*}{x^*} \leq \varepsilon \rightarrow x^H \leq (1 + \varepsilon)x^*$.
- ε represents the error.
- For problems such as the metric TSP, there exist poly-time algorithms to generate feasible solutions.

Exact algorithms

- Main general frameworks to design exact algorithms for NP-hard problems:
  - **Branch-and-Bound**
  - Cutting Planes
  - Branch-and-Cut
- Let's define some basic concepts before discussing Branch-and-Bound.

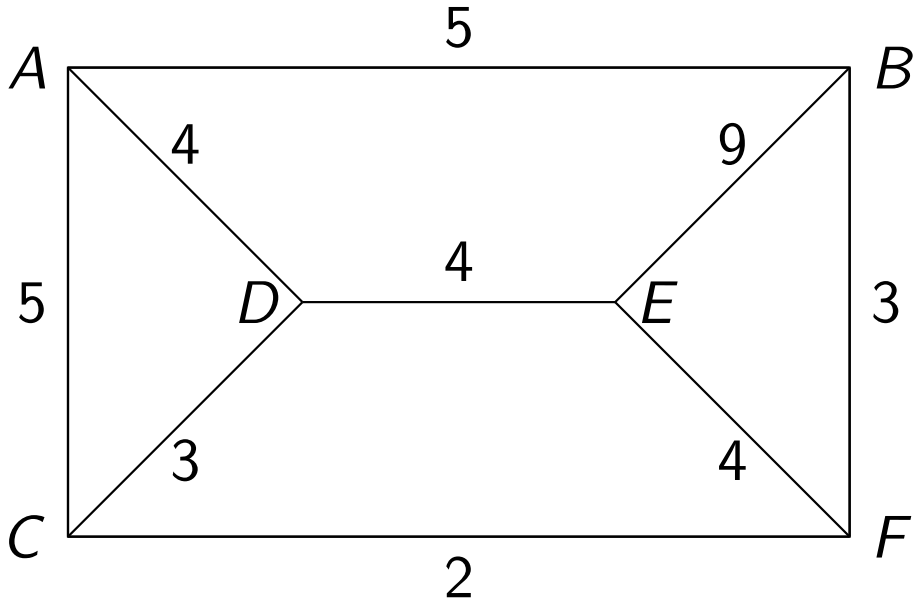
Lower and upper bounds

Let $val(x, I)$ be the function computing the value of the objective function of solution x to instance I.
Let $opt(I)$ be an optimal solution to instance I.

- **Lower Bound (LB)**:
  - Given an instance I, a *lower bound* to $val(x, I)$ is a real b such that $b \leq val(x, I)$ for every feasible solution x to I.
  - Bound b_2 is *tighter* than bound b_1 when $b_1 \leq b_2$.
  - A bound b is *tight* when there exists a feasible solution x such that $val(x, I) = b$.
- **Upper Bound (UB)**:
  - Given an instance I, an *upper bound* to $val(x, I)$ is a real B such that $B \geq val(opt(I), I)$.
  - Bound B_1 is *tighter* than bound B_2 when $B_1 \leq B_2$.
  - A bound b is *tight* when $B = val(opt(I), I)$.

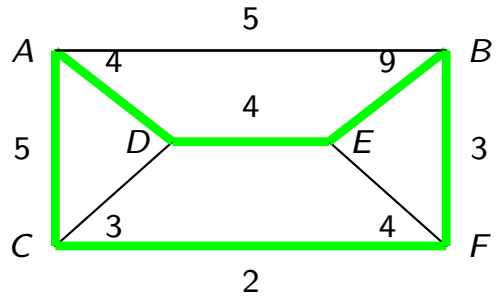
- By a lower bound (upper bound) for a certain problem, we often mean a closed formula or a poly-time algorithm that, given I, yields a lower bound (upper bound) for I.
- We will see how both lower and upper bounds can be helpful (the tighter the bound, the more useful).
- Heuristic and approximation algorithms provide upper bounds.
- Approximation algorithms also provide lower bounds. Why?

TSP Example

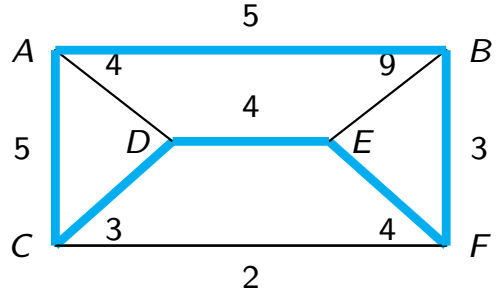


What can represent a LB for this simple graph? And an UB?

- This is a feasible solution of value 27:



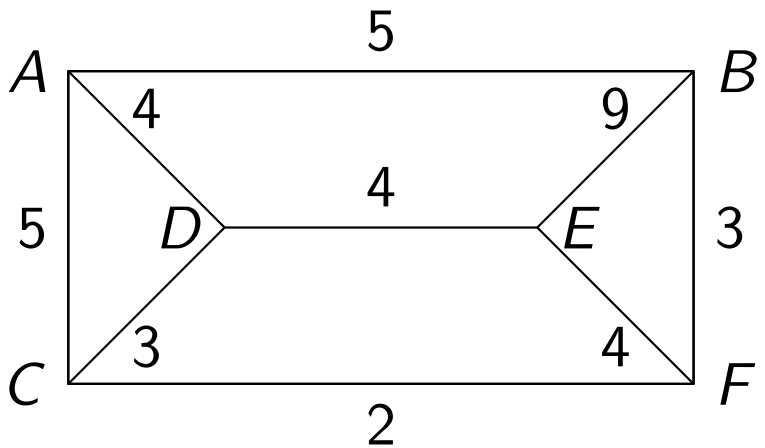
- It certifies that 27 is an upper bound on the minimum cost. Once we know it, we will never be willing to pay more than 27.
- The following solution of cost 24 provides a tighter (more interesting) upper bound:



- Thus, for minimization problems, feasible solutions represent upper bounds.

A lower bound for the TSP

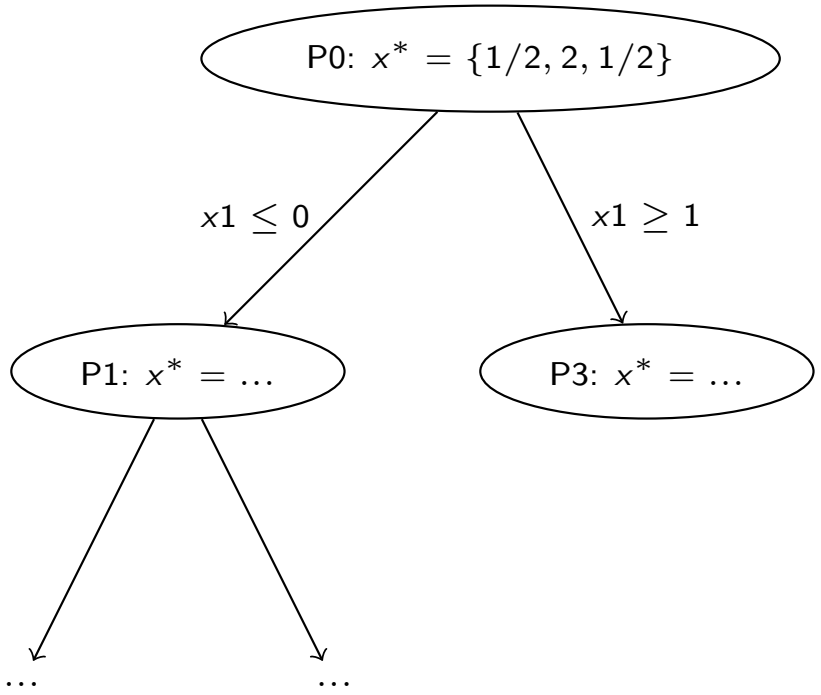
- The cost of any tour is $\frac{1}{2} \sum$ (sum of the costs of the two tour edges adjacent to v).
- The sum of the two tour edges adjacent to a given vertex $v \geq$ sum of the two edges of least cost adjacent to v.
- The cost of any tour is $\geq \frac{1}{2} \sum$ (sum of the costs of the two least cost edges adjacent to v).
- In our example, LB $= \frac{1}{2}$ [(4+5) + (3+5) + (2+3) + (3+4) + (4+4) + (2+3)] $= \frac{1}{2}$ 42 $= 21$.



- Let's introduce an exact method that exploits bounds...

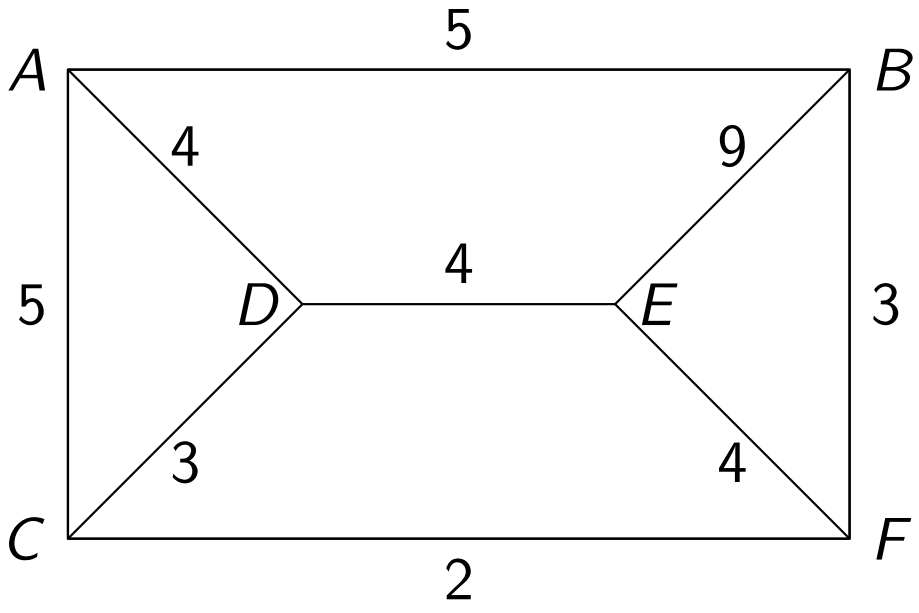
Branch-and-Bound

- It essentially explores all the solutions, but to avoid seeing them all (like a brute-force algorithm would) it implicitely **enumerates** them by exploring a tree and **branching** on possible elementary choices.
- It avoids exploring entire subtrees by comparing **lower bounds and upper bounds** and pruning some branches accordingly.

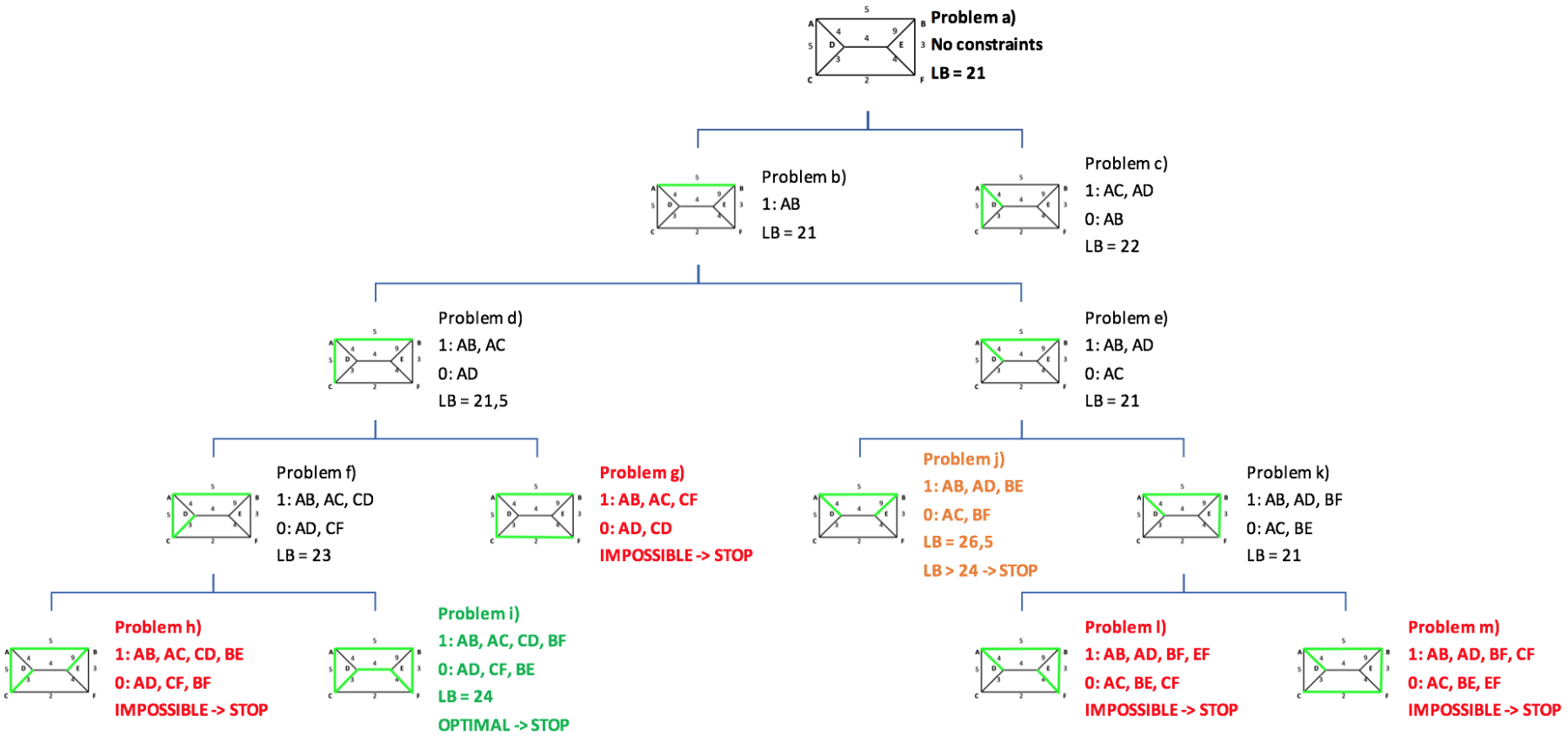


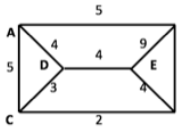
- A node in the enumeration tree is *solved* when either:
  - an optimal solution to the corresponding subproblem is found (i.e., LB = UB),
  - or we can conclude that either:
    - a feasible solution to the subproblem may be found, but it will always be worse than the best one found so far (i.e., the *incumbent* solution),
    - or the subproblem cannot be solved.
- Otherwise, after solving the subproblem the algorithm takes a definitive decision on one of the variables and branches.

Example

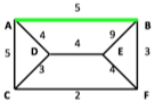


A tree search (Branch-and-Bound)

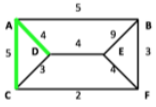




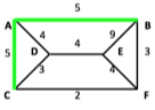
Problem a)
No constraints
LB = 21



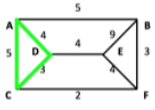
Problem b)
1: AB
LB = 21



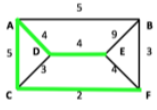
Problem c)
1: AC, AD
0: AB
LB = 22



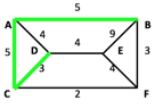
Problem d)
1: AB, AC
0: AD
LB = 22,5



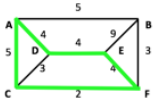
Problem n)
1: AC, AD, CD
0: AB, CF
IMPOSSIBLE -> STOP



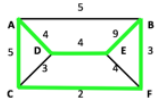
Problem o)
1: AC, AD, CF, DE
0: AB, CD
LB = 22,5



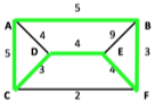
Problem f)
1: AB, AC, CD
0: AD, CF
LB = 23



Problem p)
1: AC, AD, CF, EF
0: AB, CD, BF
IMPOSSIBLE -> STOP

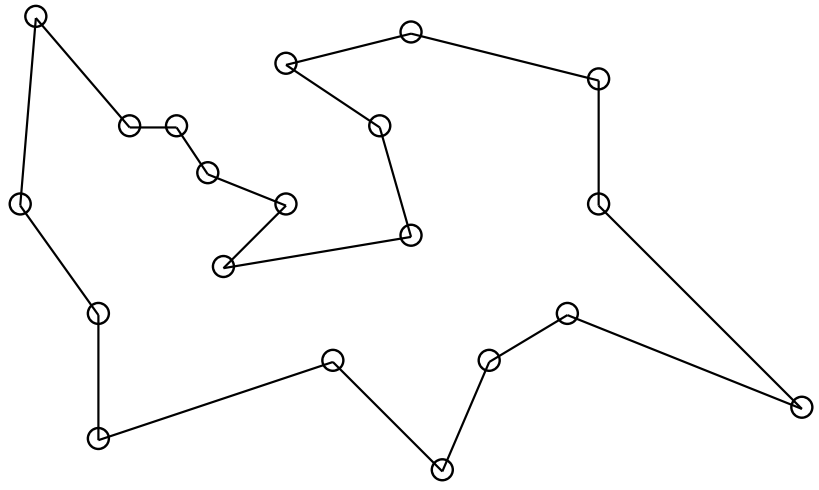


Problem q)
1: AC, AD, CF, DE, EF, BE
0: AB, CD, BF
LB = 27
LB > 24 -> STOP



Problem i)
1: AB, AC, CD, BF
0: AD, CF, BE
LB = 24
OPTIMAL -> STOP

ILP formulation of TSP (undirected graphs)



- Let V be the set of vertices and E the set of edges.
- We introduce a binary variable x_e for every edge $e \in E$: $x = 1$ iff the edge e is included in the tour.
- The goal is to minimize the total cost of edges selected, i.e., traveled by the salesman.

- For every vertex v, precisely two of the edges incident with v are selected.
- For any vertex $v \in V$, let $\delta(v)$ be the set of edges incident with v
- We obtain the following initial formulation:

$$min \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V \tag{2}$$

$$x_e \in \{0, 1\}. \tag{3}$$

Constraints (2) are called *assignment constraints* and ensure that every vertex has a degree of 2...

- For every vertex v, precisely two of the edges incident with v are selected.
- For any vertex $v \in V$, let $\delta(v)$ be the set of edges incident with v
- We obtain the following initial formulation:

$$min \quad \sum_{e \in E} c_e x_e \tag{1}$$

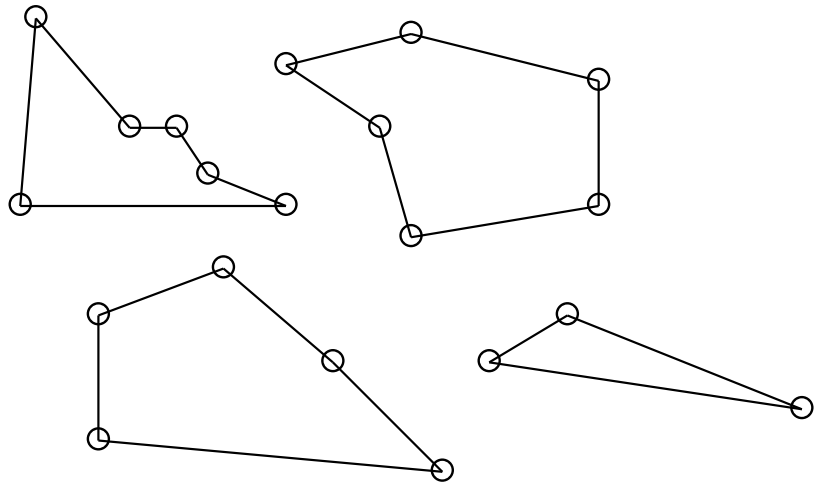
$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V \tag{2}$$

$$x_e \in \{0, 1\}. \tag{3}$$

Constraints (2) are called *assignment constraints* and ensure that every vertex has a degree of 2...

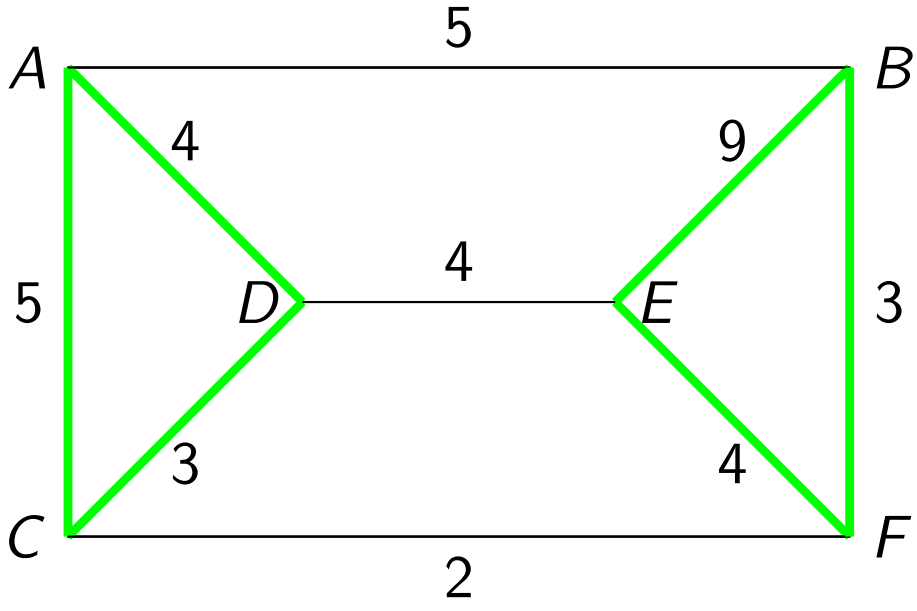
...but it's not enough!

Indeed, by using such a formulation, a solution like the following would be feasible:



But we do not want our salesman to "teleport" or "jump" from one city to another $\rightarrow$ Our formulation is hence incomplete.

In the previous example, we would have obtained the following solution:



Anyway, this solution provides valuable information: it is a lower bound for the problem.

Solving the example with AMPL

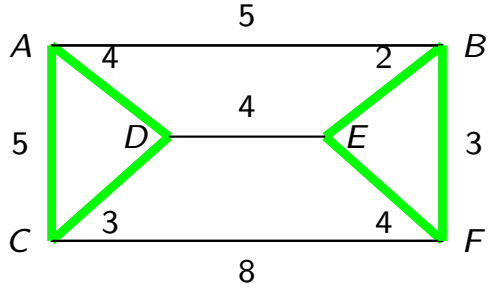
Consider the previous example but change the costs of (B,E) from 9 to 2 and (C,F) from 2 to 8:

Model:

```
set V ordered;
set E within {i in V, j in V: ord(i) < ord(j)};
param cost{E};
var X{E} binary;
minimize TourCost: sum {(i,j) in E} cost[i,j] * X[i,j];
subject to VisitAllVertices {i in V}:
sum {(i,j) in E} X[i,j] + sum {(j,i) in E} X[j,i] = 2;
```

Data:

```
set V := A B C D E F;
set E := (A,B) (A,C) (A,D)
         (B,E) (B,F) (C,D)
         (C,F) (D,E) (E,F);
param cost :=
[A,B] 5
[A,C] 5
[A,D] 4
[B,E] 2
[B,F] 3
[C,D] 3
[C,F] 8
[D,E] 4
[E,F] 4;
```



The AMPL solver finds an optimal solution with value 21, by selecting (A,C), (A,D), (B,E), (B,F), (C,D), (E,F). But it is not what we want...

To formulate the correct model for TSP, we need to add more constraints.

Subtour Elimination Constraints (I)

- TSP asks for a **Hamiltonian circuit** (i.e., a circuit passing through each vertex exactly once, without subtours).
- Given $S \subsetneq V$, $S \neq \emptyset$, the cut with shores S and V/S is the set of those edges with one endpoint in S and the other in V/S; we denote it by $\delta(S)$.
- We must impose that the solution is **connected**.

Subtour Elimination Constraints (I)

- TSP asks for a **Hamiltonian circuit** (i.e., a circuit passing through each vertex exactly once, without subtours).
- Given $S \subsetneq V$, $S \neq \emptyset$, the cut with shores S and V/S is the set of those edges with one endpoint in S and the other in V/S; we denote it by $\delta(S)$.
- We must impose that the solution is **connected**.
- For every proper subset of vertices, at least two edges in the solution have one endpoint in S and the other outside S:

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subsetneq V, S \neq \emptyset$$

Subtour Elimination Constraints (II)

- Actually, we can write:

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset V : 2 \leq |S| \leq |V| - 2$$

  Why?

- Violated constraints will have the following form:

$$\sum_{e \in \delta(S)} x_e^* < 2.$$

- These constraints are known as *Subtour Elimination Constraints* (SEC).

The SEC problem

- By introducing this family of constraints, we are adding an **exponential** number of constraints, since we are considering the power set of V (except the empty set).
- Thus, this approach is not polynomial for a double reason:
  - we adopted an ILP model;
  - its formulation has an exponential number of variables and constraints.

A correct ILP formulation of TSP (undirected graphs)

- Putting together the objective function (1) and constraints (2), (3) and (4), we get to the correct formulation:

$$\min \sum_{e \in E} c_e x_e \tag{1}$$

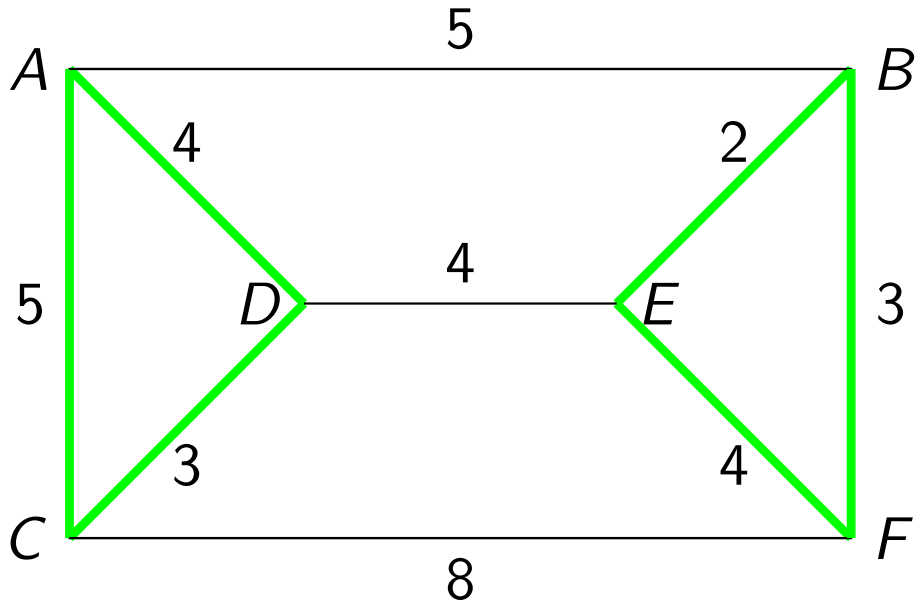
$$\text{subject to} \quad \sum_{e \in \delta(v)} x_e = 2, \ \forall v \in V \tag{2}$$

$$\sum_{e \in \delta(S)} x_e \geq 2, \forall S \subset V : 2 \leq |S| \leq |V| - 2 \tag{3}$$

$$x_e \in \{0, 1\}. \tag{4}$$

Another approach considering flow constraints

- How to avoid adding an exponential number of constraints?
- Consider the solution obtained before including subtours :



- Suppose we want to send some quantity of flow from vertex A to another vertex:
  - since A is connected to C and D, there is no problem sending it to them;
  - instead, we cannot send the flow to F.

Adding flow constraints

- For every edge $e = (u, v) \in E$, we introduce two variables $y_{(u,v)}$ and $y_{(v,u)}$ and we allow the flow to pass only through edges selected in the cycle:
  - $y_{(u,v)} \leq x_{(u,v)}$
  - $y_{(v,u)} \leq x_{(u,v)}$.
- We consider the vertices A and F as source and destination respectively; we want that the flow outgoing from A is equal to 1, whereas the incoming flow is 0:
  - $y(\delta^+(A)) = 1$
  - $y(\delta^-(A)) = 0$.
- We want all other vertices except F to conserve the flow:
  - $y(\delta^-(u)) = y(\delta^+(u)), \forall u \neq A, F$.

- In this way, we are forcing the flow to arrive in the only vertex left, i.e., F, thus connecting A and F.
- This could be repeated for every source vertex and for every destination vertex.
- Since the number of vertices is n, we are adding a polynomial number of constraints and variables.

Conclusions

- We defined the TSP, by using one simple formulation and one more complete and beautiful but, unfortunately, with an exponential number of constraints.
- We saw several methods to tackle an ILP problem (heuristic, approximation and exact algorithms).
- We solved an easy instance of TSP by using Branch-and-Bound.

References


William J. Cook, *In Pursuit of the Traveling Salesman – Mathematics at the Limits of Computation*, https://press.princeton.edu/books/paperback/9780691163529/in-pursuit-of-the-traveling-salesman


Gilbert Laporte, *A Short History of the Traveling Salesman Problem*, http://neumann.hec.ca/chairedistributique/common/laporte-short.pdf


Renata Mansini, *Algoritmi di Ottimizzazione*, University of Brescia, https://www.unibs.it/ugov/degreecourse/65037

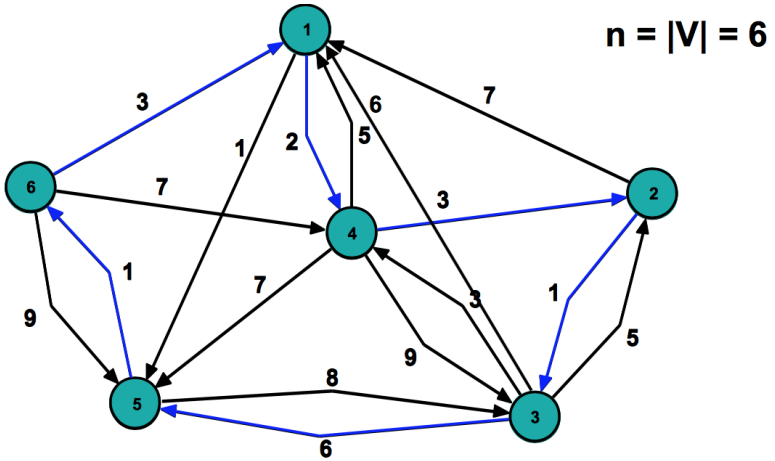

Operations Research Group, *Asymmetric TSP*, University of Bologna, http://www.or.deis.unibo.it/algottm/files/8_ATSP.pdf


Springer, *Encyclopedia of Optimization*, http://www.springer.com/it/book/9780387747583


University of Waterloo, *TSP*, http://www.math.uwaterloo.ca/tsp/


Wikipedia, *Traveling Salesman Problem*, https://en.wikipedia.org/wiki/Travelling_salesman_problem

Appendix A – The asymmetric TSP (ATSP)



- Let V be the set of vertices.
- Sometimes going from vertex i to vertex j does not cost the same than going from vertex j to vertex i, or maybe it is not possible to go through both directions.
- Then, instead of the set of edges E, we need to use the set of arcs A: this variant is called **asymmetric** TSP.
- We introduce a binary variable $x_{i,j}$ for every arc $(i, j) \in A$: if $x_{i,j} = 1$, then arc (i, j) appears on the tour.

ILP formulation of ATSP (directed graphs)

- Keeping in mind what we said about symmetric TSP, we can formulate ATSP as follows:

$$min \sum_{(i,j)\in A} c_{i,j}x_{i,j} \tag{1}$$

$$\text{subject to} \quad \sum_{j\neq i} x_{i,j} = 1,\ \forall i \in V \tag{2}$$

$$\sum_{i\neq j} x_{i,j} = 1,\ \forall j \in V \tag{3}$$

$$\sum_{i\in S, j\in S} x_{i,j} \leq |S| - 1, \forall S \subset V, S \neq \emptyset \tag{4}$$

$$x_{i,j} \in \{0,1\}. \tag{5}$$

- With constraints (2) and (3), we are distinguishing arcs entering in a certain vertex and arcs exiting from it.

Appendix B – William J. Cook, *The TSP: postcards from the edge of impossibility* (EURO 2019)

https://www.youtube.com/watch?v=5VjphFYQKj8