CIVIL-557

Decision aid methodologies in transportation

Lab 2: Using a mathematical solver II (Branch & Cut)

Tom Haering

Transport and Mobility Laboratory (TRANSP-OR)
École Polytechnique Fédérale de Lausanne (EPFL)

28.02.2023

Overview

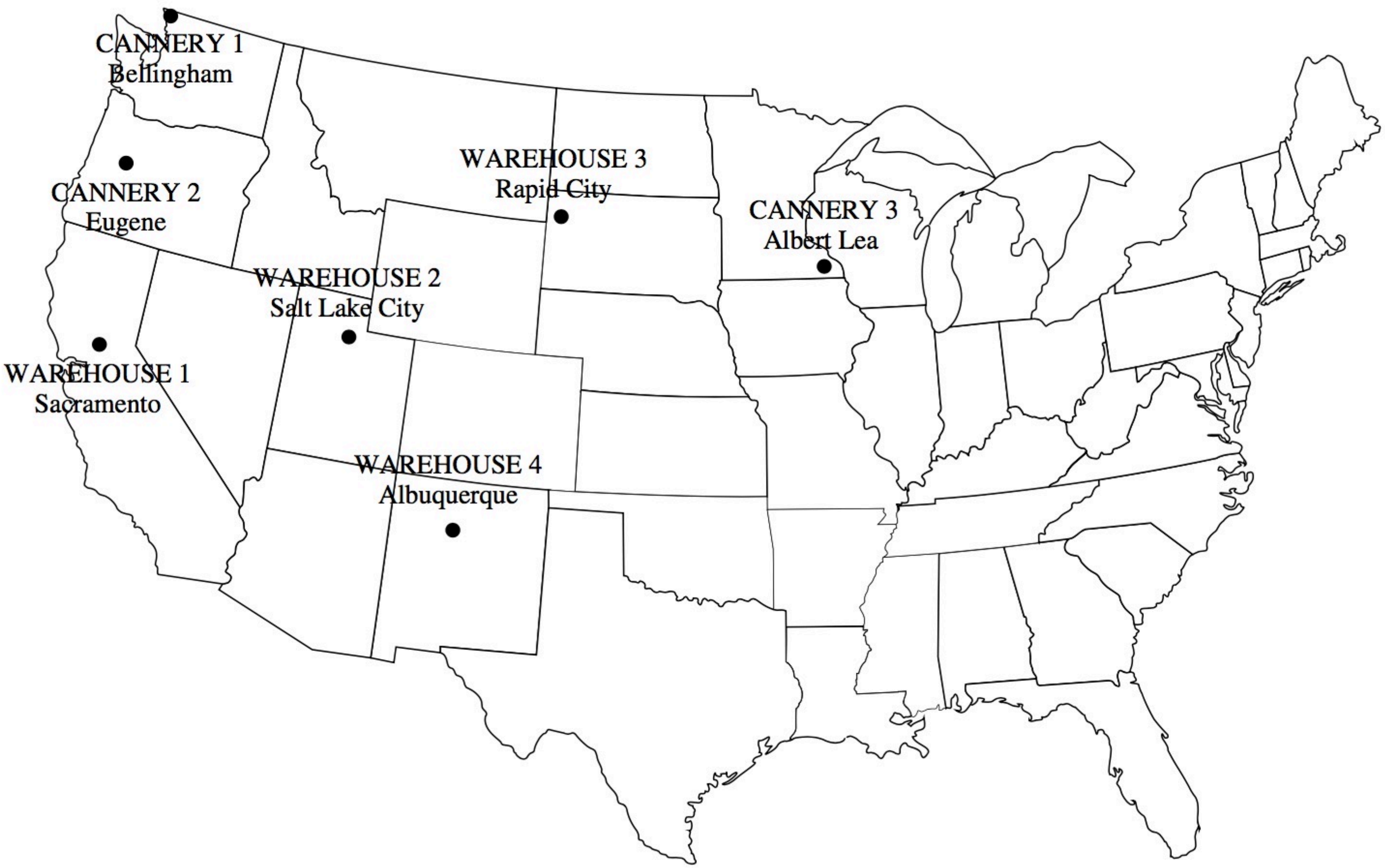
- Solution from last weeks bonus exercise
- What is Branch & Cut?
- Example: Traveling salesman problem (TSP)
- Extension to vehicle routing problem (VRP)

Transportation problem (last weeks bonus exercise)

Transportation Problem

One of the main products of the P&T COMPANY is canned peas. The peas are prepared at three canneries (near Bellingham, Washington; Eugene, Oregon; and Albert Lea, Minnesota) and then shipped by truck to four distributing warehouses in the western United States (Sacramento, California; Salt Lake City, Utah; Rapid City, South Dakota; and Albuquerque, New Mexico). Because the shipping costs are a major expense, management is initiating a study to reduce them as much as possible. For the upcoming season, an estimate has been made of the output from each cannery, and each warehouse has been allocated a certain amount from the total supply of peas. This information (in units of truckloads), along with the shipping cost per truckload for each cannery-warehouse combination, is given in Table 8.2. Thus, there are a total of 300 truckloads to be shipped. The problem now is to determine which plan for assigning these shipments to the various cannery-warehouse combinations would *minimize the total shipping cost.*

Transportation Problem – Layout



Transportation Problem – Table 8.2

TABLE 8.2 Shipping data for P & T Co.

		Shipping Cost ($) per Truckload				
		Warehouse				
		1	**2**	**3**	**4**	**Output**
Cannery	1	464	513	654	867	75
	2	352	416	690	791	125
	3	995	682	388	685	100
Allocation		80	65	70	85	

Transportation Problem

- Formulate the problem mathematically
- Implement the problem and report the optimal solution
- Solve the problem for a large instance (use N= 20, M = 30, and files Lab1_cost.npy, Lab1_ output.npy and Lab1_ allocation.npy).

Transportation Problem Solution

$$min \sum_{i=1}^{N} \sum_{j=1}^{M} cost_{ij} x_{ij}$$

$$s.t. \sum_{j=1}^{M} x_{ij} \leq \; ouput_i \qquad \forall i \in \{1, \ldots, N\}$$

$$\sum_{i=1}^{N} x_{ij} \geq \; allocation_j \qquad \forall j \in \{1, \ldots, M\}$$

$$x_{ij} \; \in \; \mathbb{N} \qquad \forall i, j$$

TRANSP-OR

EPFL

Transportation Problem Solution


```
N = 20
M = 30
cost = np.load("Lab1_cost.npy", cost)
output = np.load("Lab1_output.npy", output)
allocation = np.load("Lab1_allocation.npy", allocation)
```

```
m = gp.Model()

x = {(i, j): m.addVar(vtype=GRB.INTEGER)
     for i in range(N) for j in range(M)}

m.setObjective(gp.quicksum(cost[i,j] * x[i, j]
                for i in range(N) for j in range(M)),
               GRB.MINIMIZE)

for i in range(N):
    m.addConstr(gp.quicksum(x[i, j] for j in range(M)) <= output[i])
    m.addConstr(gp.quicksum(x[i, j] for j in range(M)) <= output[i])
for j in range(M):
    m.addConstr(gp.quicksum(x[i, j] for i in range(N)) >= allocation[j])

m.optimize()

print(f"Optimal objective = {m.ObjVal}")
for i in range(N):
    for j in range(M):
        print(f"Optimal x[{i},{j}] value = {x[i,j].x}")
```

Optimal objective value is = 516468

Overview

- Solution from last weeks Bonus exercise
- What is Branch & Cut?
- Example: Traveling salesman problem
- Set-partitioning problem

What is Branch & Cut?

What is Branch & Cut?

- **Simple:** It's Branch & Bound with added cuts at every (or some) nodes
- Mainly used for solving problems with **integer** variables
- Often, we **only add a cut** at **integer solutions**, but sometimes also at fractional solutions

What is Branch & Cut?

Relaxation

All binary variables
$b_i \in [0, 1]$

$b_1 = 0$
Rest $\in [0, 1]$

$b_1 = 1$
Rest $\in [0, 1]$

Node 1

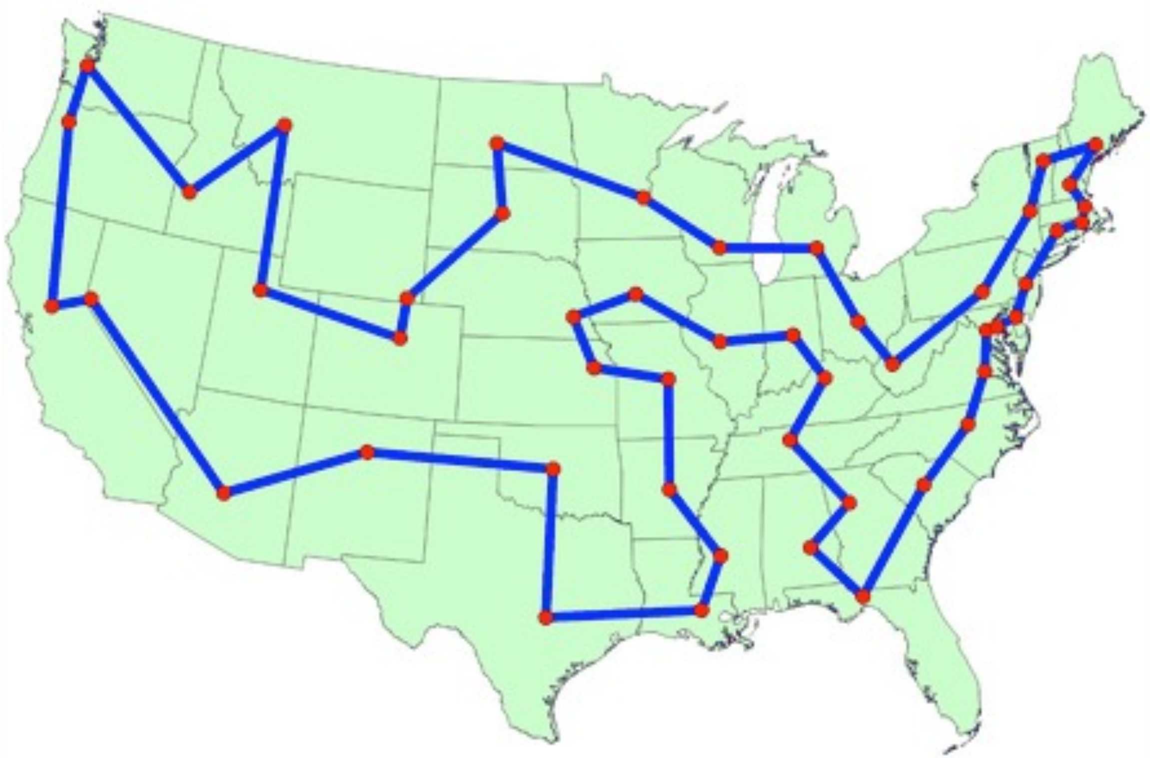
Node 2

- Add cut to tighten relaxation
(if conditions apply)
- Else continue branching on b_2 for example

Example:
Traveling salesman problem

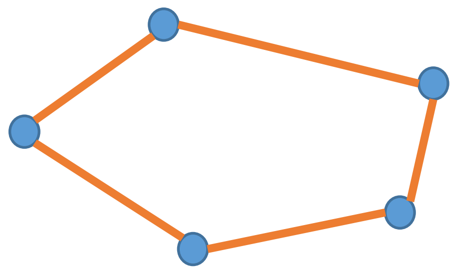
Example: Traveling salesman problem

- **Input:** set of n **points** as (x, y) coordinates
- **Goal:** find the **shortest tour** that **visits every point** and returns to the origin



Example: Traveling salesman problem

- **Mathematical formulation:**



$$\min \sum_{(i,j) \in E} dist_{ij} x_{ij}$$

$$s.t. \quad \sum_{j \neq i} x_{ij} = 2 \qquad \forall i \in \{1, \ldots, N\}$$

Degree 2 constraints

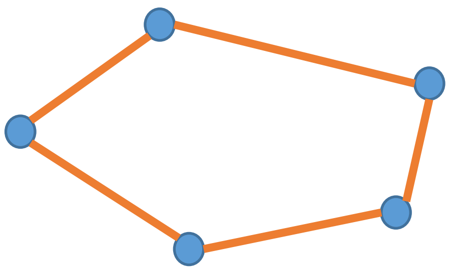
$$\sum_{(i,j) \in \delta(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subsetneq N$$

No cylces in subsets!

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in E$$

Example: Traveling salesman problem (TSP)

- **Mathematical formulation:**



$$min \sum_{(i,j) \in E} dist_{ij} x_{ij}$$

$$s.t. \quad \sum_{j \neq i} x_{ij} = 2 \qquad \forall i \in \{1, \ldots, N\}$$

Degree 2 constraints

Exponentially many!

$$\sum_{(i,j) \in \delta(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subsetneq N$$

No cylces in subsets!

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in E$$

Example: Traveling salesman problem (TSP)

- **Idea:**

  - Start with **relaxation** (no subset constraints)
    => perform standard **integer Branch & Bound**
  - Every time we find an integer solution, check if there is a subset constraint that is **violated**
    - if yes => **add the constraint** to the **branch** (cut!)
    - if no => solution is **optimal**

Example: Traveling salesman problem (TSP)

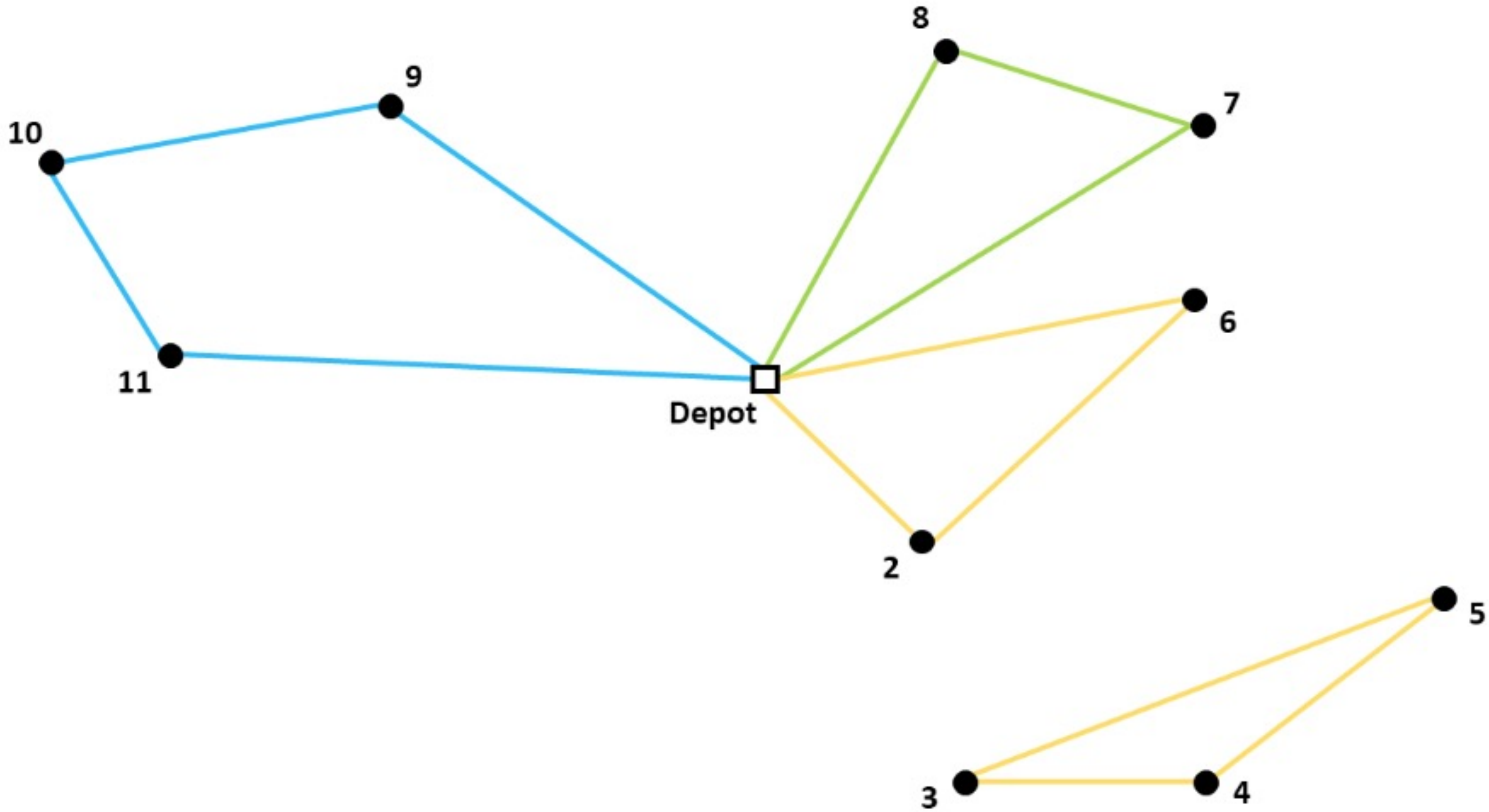
- TSP_VRP.ipynb (or TSP_VRP.py)
- First code full MILP model
- Then code a function that detects a (shortest) cycle, given an integer solution to the TSP:

  - Start at any node
  - Degree 2 constraints imply: only cycles are possible
    => go along the neighbors until you're back at the start
  - If len(cycle) < n then we found a violation

- Run Gurobi using callbacks (Branch & Cut)

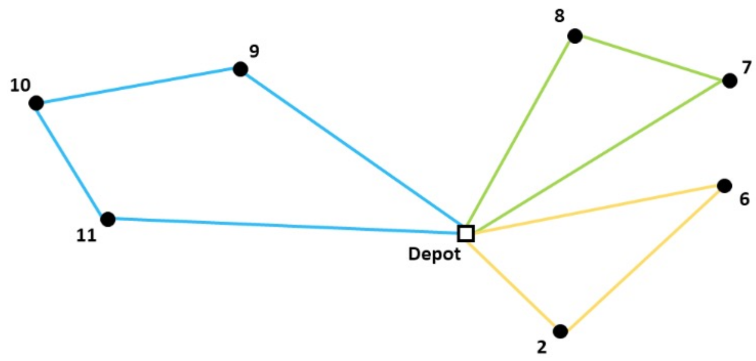
Extension to the Vehicle Routing Problem (VRP)

Extension: Vehicle routing problem (VRP)



Extension: Vehicle routing problem (VRP)

- **Mathematical formulation:**



$$\min \sum_{(i,j)\in E} dist_{ij}x_{ij}$$

$$s.t. \quad \sum_{j\neq i} x_{ij} = 2 \qquad \forall i \in \{1,\ldots,N\} \setminus \text{depot}$$

Degree 2 constraints

$$\sum_{(i,j)\in\delta(S)} x_{ij} \le |S| - \left\lceil \frac{|S|}{Q} \right\rceil \qquad \forall S \subsetneq N$$

No cylces in subsets! + capacity constraint

$$x_{ij} \in \{0,1\} \qquad \forall i,j \in E$$

TRANSP-OR

EPFL

Extension: Vehicle routing problem (VRP)

- **Idea:**
  - Start with **relaxation** (no subset constraints)
    => perform standard **integer Branch & Bound**
  - Every time we find an integer solution,

    1. Check if there is a **cycle not connected to the depot**
       if yes, eliminate with subset constraint $(\ldots \leq |S| - 1)$
    2. Else, check if any route violates **capacity constraint**
       if yes, eliminate with constraint $(\ldots \leq |S| - \frac{|S|}{Q})$
       else, solution is **optimal**