CIVIL-557

Decision aid methodologies in transportation

Lab 5: Heuristics: Column Generation

Tom Haering

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

19.03.2024

Overview

- Lab 1 & 2 : Gurobi, VRP
- Lab 3: Labeling Algorithm, SPP
- Lab 4: Heuristics

- Lab 5: Coding a heuristic for the VRP that uses the labeling algorithm to solve an SPP to iteratively add columns to a master problem that we solve with Gurobi 💯💯🔥🔥

Overview

- Lab 3: Labeling algorithm to solve **shortest path problem (SPP)** given **rewards** for all customers as input

- Today: SPP is the **subproblem** when solving the vehicle routing problem (VRP) using **branch and price**

  or:

  when solving the VRP using a **column generation** heuristic

Overview

- Branch & Price:

    1. Take **VRP MILP** (set partitioning formulation)

$$
\begin{aligned}
\mathit{Minimize} \sum_{r\in\Omega} c_r\lambda_r & \\
s.t. \sum_{r\in\Omega} a_{ir}\lambda_r = 1 & \qquad \forall i \in N, \\
\sum_{r\in\Omega} \lambda_r = |K|, & \\
\lambda_r \in \{0,1\} & \qquad \forall r \in \Omega.
\end{aligned}
$$

Overview

- Branch & Price:

  2. Start with some **dummy variable** routes:

  Example:

$$\lambda_i = \text{route from } \textbf{depot to customer } \boldsymbol{i} \textbf{ and back}$$

$$cost_i = 2 * \text{distance(depot, } i)$$

$$a_{ij} = 1 \text{ if } i = j, 0 \text{ else}$$

$$\Omega^0 = \{\lambda_1, \ldots, \lambda_n\} \text{ set of all dummy routes}$$

Overview

- Branch & Price:

  3. Solve the **relaxed** restricted **master problem**:

$$
\begin{aligned}
&Minimize \sum_{r\in\Omega} c_r \lambda_r \\
&s.t. \sum_{r\in\Omega} a_{ir}\lambda_r = 1 && \forall i \in N, && (\varphi_i) \\
&\sum_{r\in\Omega} \lambda_r = |K|, \\
&\lambda_r \in [0,1] && \forall r \in \Omega^0
\end{aligned}
$$

Overview

- Branch & Price:

4. Read values of the **dual variables** (reduced cost of visiting a customer) corresponding to constraints (φ_i):

$$reward_i = \varphi_i$$

5. **Solve** the **SPP** with these rewards
   - Compute shortest path = route r
   - Compute cost c_r = total length of route r
   - Compute incidence vector a_{ir} of route r
   - Compute the reduced cost of the solution: total cost minus the collected rewards

$$c_r - \sum_i a_{ir}\varphi_i$$

if < 0, adding route r to MP decreases total cost

Overview

- Branch & Price:

6. **Add the route** to the master problem

7. **Resolve the master problem**, read dual variables, solve subproblem with new rewards, etc.

8. **Repeat until reduced costs** of SPP are **zero** (no more possible improvements)

9. Resolve the master problem as **MILP**

Solution = **column generation heuristic**

$\Rightarrow$ not guaranteed to be optimal

Overview

- Branch & Price:

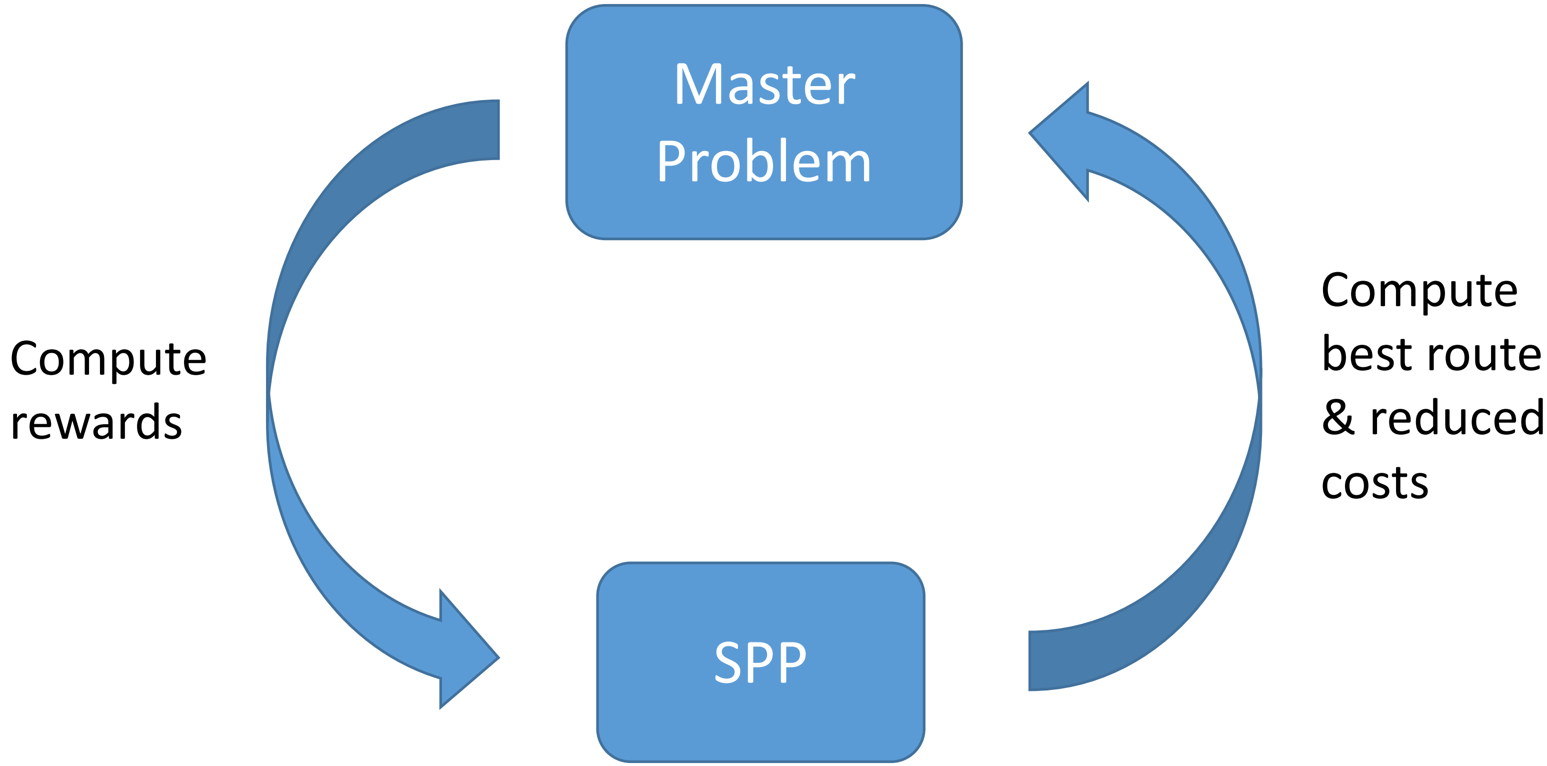
In **Branch & Price** step 9 is instead:

9. Resolve the master problem as a **relaxation.**
   We might get some routes with fractional values.

   $\Rightarrow$ **Branch** on fractional variables
   $\Rightarrow$ Repeat process for both child nodes
   $\Rightarrow$ Continue until no more fractional values in solution
   $\Rightarrow$ Guaranteed to be **optimal**

Overview

- Column generation:



Today's lab

- In this exercise class we will do only the **column generation heuristic**
- Start from code for solving the SPP with **loads**, vehicle **capacity** and **time window** constraints from week 3
- Extend it to fit in a column generation procedure

Today's lab

- Exercise 1: Initialize **master problem**

$$
\begin{aligned}
Minimize \sum_{r\in\Omega} c_r \lambda_r & \\
s.t. \sum_{r\in\Omega} a_{ir}\lambda_r = 1 & \qquad \forall i \in N, \\
\sum_{r\in\Omega} \lambda_r = |K|, & \\
\lambda_r \in [0,1] & \qquad \forall r \in \Omega^0
\end{aligned}
$$

Today's lab

- Exercise 1: Initialize master problem

$$\begin{aligned}
&Minimize \sum_{r\in\Omega} c_r \lambda_r \\
&s.t. \sum_{r\in\Omega} a_{ir}\lambda_r \geq 1 && \forall i \in N, \\
&\cancel{\sum_{r\in\Omega} \lambda_r = |K|,} && \textbf{Not necessary} \\
&\lambda_r \in [0,1] && \forall r \in \Omega^0
\end{aligned}$$

Today's lab

- Exercise 1: Initialize master problem

```
def initialize_master(n):
    # Create a new model
    m = gp.Model("Master Problem")

    # Create variables for dummy routes (create a dicitionary of variables)

    # Set objective

    # Add constraint that every costumer needs to be visited by at least one
    # vehicle (store constraints in a dictionary as you would for variables):

    # Optimize model
    m.setParam("OutputFlag", 0)
    m.optimize()

    # read dual variables of costumer constraints (create a dictionary called
    # "reward" with customers as keys)

    # set the reward for the depot to be 0

    return m, reward, visit, lambda_vars
```

Today's lab

- Exercise 1: Initialize master problem

```
# creating gurobi variables
m.addVar(lb, ub, vtype=GRB.CONTINUOUS / GRB.BINARY, name)

# setting objective
m.setObjective(objective expression, GRB.MINIMIZE / GRB.MAXIMIZE)

# adding constraints
m.addConstr(variable >= value)

# sums
gp.quicksum(variable[i] for i in range(number))

# solution value of variables
variable.x

# dual values of constraints
constraint.Pi
```

Today's lab

- Exercise 2: Finish the column generation implementation

```
# initialize master problem
master, first_reward, visit, lambda_vars = initialize_master(n)

iteration = 0
reward = first_reward
red_cost = -100000
tours = dict()
```

Today's lab


```python
# while there exist positive reduced costs, generate new columns
while red_cost < -1e-13:
    iteration += 1
    if iteration % 10 == 0:
        print(f"iteration = {iteration}, reduced costs = {red_cost}")
    l, red_cost = labeling_algorithm(reward)

    # generate tour
    tour = []
    while l.parent:
        tour.append(l.customer)
        l = l.parent
    tour.append(Start_id)
    tour.reverse()
    tour.append(Start_id)
    tours[iteration] = tour
    tour_length = total_distance(tour, distance)

    # create new column a[:, r] for the new found route

    # add the route as new variable to the master problem

    # update customer constraints:

    # resolve master problem
    master.optimize()
    if iteration % 10 == 0:
        print("obj function = ", master.ObjVal)
    # read dual variables of costumer constraints (rewards)
```

Today's lab


```
# count how many times an element appears in a list
List.count(element)

# adding a new variable to a gurobi model (including objective coefficient)
model.addVar(lb, ub, obj=coeff, vtype, name)

# update the coefficient of a variable in a constraint
model.chgCoeff(constraint, variable, coefficient)
```

Today's lab

- Exercise 3: Dealing with cycles

```
# Feasibilty check
def feasible(l, i):
    """Returns if extending from label l to node i is feasible"""
    #Is it feasible to go to node i from label l?
    if l.customer == i:
        return False
    if distance[(l.customer,i)] +l.time > late[i]:
        return False
    if l.load + demand[i] > Q:
        return False
    return True
```

```
# check if a customer was already visited:

# go through all parent labels and check if
# any of them are at that customer

l.parent = parent label (label visited previously
                         in the path)
l.customer = node at current label l
```