The Capacitated Vehicle Routing Problem

Tomáš Turek

October 7, 2024

Linear program - variables

For $i \neq j \in V$ we declare two variables:

$$x_{i,j} = \begin{cases} 1 & \text{if a truck goes through the edge } ij \\ 0 & \text{otherwise} \end{cases}$$

$$[0, Q] \ni y_{i,j} = \text{amount of goods transferred between } i \text{ and } j$$

And connect them by constraint:

$$y_{i,j} \leq Q \cdot x_{i,j}$$

Lastly the number of trucks: $k \in [0, n]$, where $n = |V|$. So the optimization function is as follows.

$$\min \sum_{i \in V} \sum_{j \in V, j \neq i} c_{ij} x_{ij}.$$

Linear program - constraints

For each node one truck is entering and leaving and for depot it is the number of trucks.

$$\forall i \in V \setminus \{d\} : \sum_{j \in V, j \neq i} x_{ij} = 1$$

$$\forall i \in V \setminus \{d\} : \sum_{j \in V, j \neq i} x_{ji} = 1$$

$$\sum_{j \in V, j \neq d} x_{dj} = k$$

$$\sum_{j \in V, j \neq d} x_{jd} = k$$

Also count the amount of good that is given to the customer.

$$\forall i \in V \setminus \{d\} : \sum_{j \in V, j \neq i} y_{ji} - \sum_{j \in V, j \neq i} y_{ij} = q_i$$

Approximation algorithm

```
Require: x, y from LP and instance of the problem.
Ensure: Integer solution.
current = d
capacity = Q
while There exist not visited customer. do
    Filter neighbors based on x, capacity and if they were visited or not.
    if No such neighbors exists. then
        Choose them without considering x. And give them equal chances.
    end if
    Choose one of the neighbors based on their x values. Set it as current and decrease capacity.
    Update integer result.
    if We ended in depot. then
        Reset capacity to Q.
    end if
end while
```

Used solvers

The main program is parser written in Rust. Firstly it will create linear program in Gurobi format and then it is parsed to the NEOS Server (also possible to run locally).
After the solution file is created parse it and find the approximation via the same program.

```
cargo run [ilp|lp|apx] path/to/vrp_file [|path/to/sol_file]-r
```

Note that also automatization script is provided and so is python checker for each solution.

Results

Instance	LP relaxation	ILP solution	Approximation	Upper bound	Lower bound
E-n13-k4	217	247	294	247	247
E-n22-k4	307	375	441	375	375
E-n23-k3	471	569	629	569	569
E-n30-k3	418	505	532	534	534
E-n31-k7	330	379	461	379	379
E-n33-k4	717	838	1064	835	835
E-n51-k5	474	525	703	521	521
E-n76-k7	598	–	997	682	682
E-n76-k8	640	–	1082	735	735
E-n76-k10	714	–	1231	830	830
E-n76-k14	858	–	1501	1021	1021
E-n101-k8	736	–	1307	817	–
E-n101-k14	939	–	1698	1071	–

Table: Results using our solution and known bounds.