COIL: A Deep Architecture for Column Generation

Babaki, Charlin and Jena

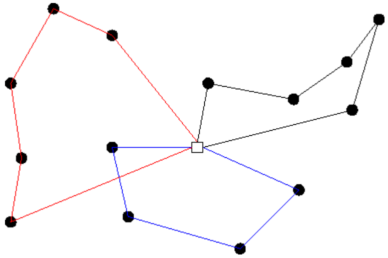
Lecture of Mathieu Lacroix

01/18/2023

Outline

Capacitated VRP

- **Instance:**
  - N customers having a demand d_i for $i \in \{1, \ldots, N\}$ and a location,
  - A depot with a location
  - Set vehicles of capacity c
- **Objective:** Find a set of route of minimum cost s.t.
  - Each customer is in one route,
  - the sum of demands of customers inside a route is no more than c



Extended formulation

Consider a variable x_r for each possible route $r \in R$

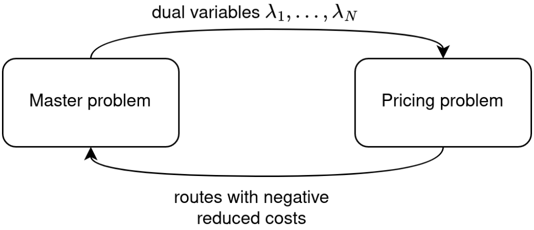
$$\min \quad \sum_{r \in R} c_r x_r \tag{1}$$

$$\sum_{r \in R} e_{ir} x_r \geq 1 \qquad \forall i \in \{1, \ldots, N\} \tag{2}$$

$$x_r \in \{0, 1\} \qquad \forall r \in R. \tag{3}$$

Outline

Column generation



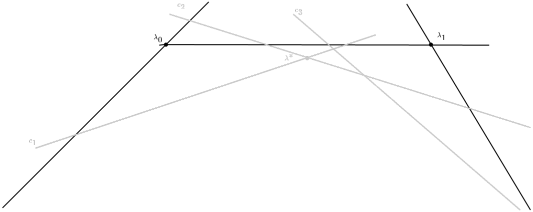
Pricing problem

Compute the route with minimum reduced cost:

$$\hat{r} = \arg\min_{r \in R}(c_r - \sum_{i \in R} \lambda_i)$$

Column generation

Dual viewpoint: cutting plane



Slow convergence

- Several optimal dual solutions at each iteration (ex: segment between λ_0 and λ_1)
- The route returned by the pricing (or separation) depends on the dual solution $\Rightarrow$ Huge number of useless columns are added

Stabilization is needed

Outline

Objective of the paper

- Convergence depends on the column added at each iteration
  $\Rightarrow$ Use ML model to predict the column to add

How does it work?

At each iteration:

- Use ML model at each iteration to predict/choose the "best" solution among dual optimal ones
- give this solution to the pricer and get the returned column

Learn to choose a dual solution such that the returned column is the same as the expert.

Markov Decision Process

CG can be seen as a sequential decision problem (which column to add at each iteration) $\Rightarrow$ MDP

- **State** s_t, z with:
  - s_t representing the RMP at iteration t
  - z representing the CVRP instance
- **Action** a_t: next column to add
- **Policy** probability distribution $\pi_\theta(a_t|s_t, z)$ over actions:
  - conditioned by state s_t, z
  - parametrized by θ

Learning θ

- Imitation learning
- For each instance $j \in J$, an expert solves j with CG:
  - Set of T_j states $s_t^{(j)}$, $z^{(j)}$,
  - Action $a_t^{*(j)}$ performed by the expert ("best" one)
- Learn θ to minimize

$$\mathcal{L}(\theta) = -\sum_{j \in J} \sum_{t=1}^{T_j} \log \pi_\theta(a_t^{*(j)}|s_t^{(j)}, z^{(j)})$$

Expert Data Collection

Consider a set of CVRP instances. For each instance $j \in J$

- Solve CG to optimality to get **a** dual optimal solution λ^*.
- Solve a second time CG where at iteration t:
  - RMP_t is solved to optimality (linear problem)
  - Compute the dual optimal solution $\bar{\lambda}_t$ of RMP_t closest to λ^* (quadratic problem)
  - Call pricing solver with $\bar{\lambda}_t$
  - The returned column is considered as $a_t^{*(j)}$.

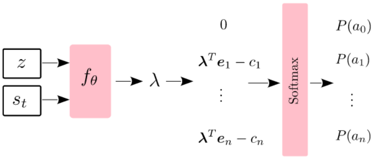
Policy network

ML Model

- State s_t, z as input
- Returns a dual optimal solution of RMP_t

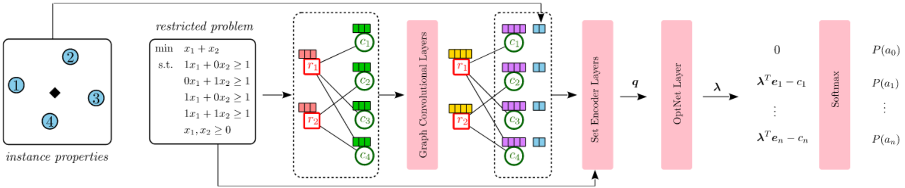
From ML prediction λ to policy

- Compute score $\lambda^T e_r - c_r$ for each route $r \in R$
- Score of 1 for not returning a column
- From score to probabilities: softmax (higher score $\Rightarrow$ higer probability)



Remark: Training can be only made on instances for which columns can be enumerated

ML Model



Representing RMP_t

- Use of a graph neural network (bipartite graph Grasse et al.)
- Solve RMP_t to initialize node features:
  (optimal solution, reduced cost) and (dual sol, slack)

$\Rightarrow$ Computed features c'_i (constraint) for each customer $i \in \{1, \ldots, N\}$

ML Model

Representing the instance properties

- Concatenate c'_i with customer instance features (location, demand) $\Rightarrow c''_i$ for each $i \in \{1, \dots, N\}$
- Use a set encoder layer with self-attention to obtain a vector $q \in \mathbb{R}^N$ (q_i represents customer i)

Finding dual optimal solution

- Use of OptNet Layer to find a dual optimal solution of RMP_t minimizing $\frac{1}{2}\lambda^T Q \lambda + q^T \lambda$ with $Q = 0.001I$
- Remarks/Questions:
  - Needs to solve a quadratic problem
  - $Q \neq \mathbf{0}$ to be differential?
  - Could we set $Q = \mathbf{0}$ after training (to speed-up)?
  - Since Q is small, it tends to return an extreme point (on the contrary to the expert)

Outline

Comparison

300 training, 300 validation and 1652 test instances with 21 customers (random generator)

Methods

- **IPS** (method without learning but stabilization)
- **Baseline**: learn a same q vector for all training instances (no state nor context)
- **MLP** applied independently to each customer i to get q_i
- **COIL-S**: set encoder part
- **COIL-G**: GNN part
- **COIL-GS**: whole pipeline

Computational results

	IPS	*Baseline*	*MLP*	*COIL-S*	*COIL-G*	*COIL-GS*
#Wins	1	278	296	248	436	**695**
Ratio	1.581	1.239	1.211	1.232	1.17	**1.12**

- Wins: number of times the method has minimum number of iterations
- Ratio: number of iterations w.r.t. number of iterations of the expert

Remark: Learning a same q is not so bad. Due to QP not returning an extreme point?

Computational results

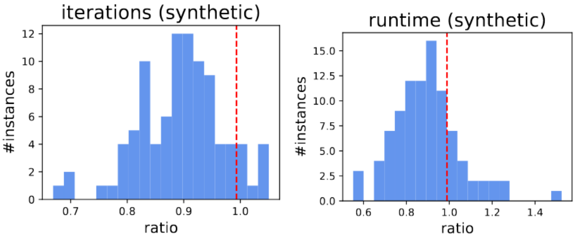
	Loss	*Top-k Accuracy*			
		1	*10*	*100*	*1000*
Baseline	42.37	0.445	0.773	0.933	0.98
MLP	36.134	0.459	0.8	0.947	0.987
COIL-S	34.609	0.461	0.809	0.949	0.988
COIL-G	29.961	0.499	0.836	0.958	0.989
COIL-GS	**25.134**	**0.537**	**0.861**	**0.966**	**0.990**

$top - k$: percentage of time the expert column is in the k columns with highest probability.
Remark: more or less half of the time, the predicted column is not the "good" one.

Computational results

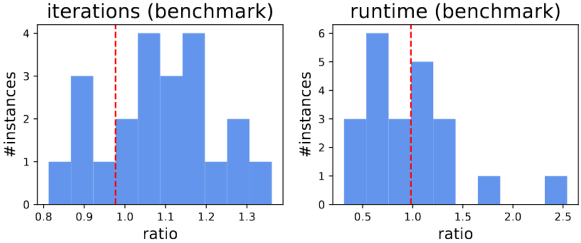
Bigger instances

Comparison with IPS (no learning) on 100 random instances with 50 customers (same generator)



Learning works!

Comparison with IPS (no learning) on benchmark CVRP (up to 100 custormers)



Learning does not work ! :)
Remark: Learning always outperforms unstabilized CG