Reinforcement Learning for Service Placement and Routing under Delay Constraint

AEP 13

Toulouse, 3th Dec 2024

Orso Forghieri, Erwan Le Pennec [1] ;

Emmanuel Hyon [2] ;

Hind Castel Taleb [3];

Nancy Perrot, Yannick Carlinet [4].

[1]CMAP Ecole Polytechnique

[2]LIP6, Université Paris Nanterre

[3]Telecom SudParis

[4]Orange Gardens

Table of Contents

Problem Statement

ISP WareHouse

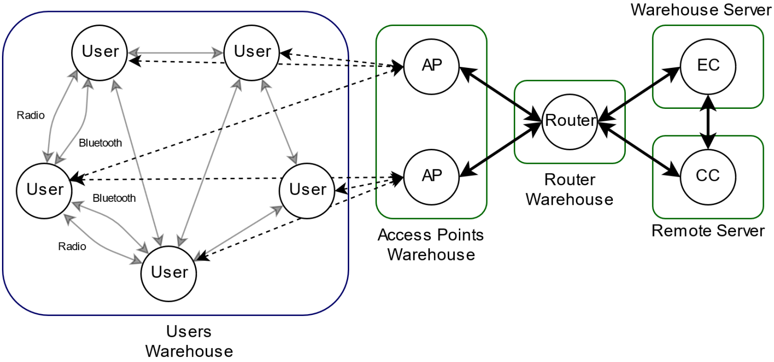

Figure 1: Smart Warehouse instance Scheme

Problem Statement
Edge computing

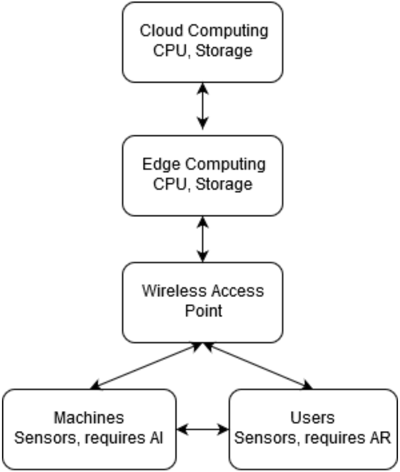

Figure 2: Warehouse network

Customers arrive with:

- A given service to fulfill

The ISP problem: **Where to place the service** ?

1. Cloud (Cloud computing),
2. Edge (Edge computing),
3. WAP.

Service in Cloud: high latency but large computing power

Service in edge: low latency but small computing power

Description of services

A Service

- Several requests with the same destination (User data and server data meet at an intermediate point)
- A request has a known origin (User or server position in the network)

A **demand** d has

- an origin d_o, a destination d_a
- a maximum latency l_d
- a variable quality $\mathbf{q}_d \in \{q_{d,min}, ..., q_{d,max}\}$
- a maximum latency l_d
- a path from origin to destination

Destination and path of demands should be determined.

Description of the network

The network

An undirected graph $\mathcal{G} = (\mathcal{E}, \mathcal{A})$ with the following features:

- $\mathrm{res}^e \in \mathbb{N}$: available resources at a node $e \in \mathcal{E}$ (CPU, storage),
- $\mathrm{capa}^a \in \mathbb{N}^+$: capacity of each arc a,
- lat^a : capacity of each arc a,

Latency on an arc :

Using an arc $a \in \mathcal{A}$ with bandwidth q_a gives a latency

$$\mathrm{lat}^a = \alpha^a \sum_{d \in \mathcal{D}} \mathbf{q}_a + \beta^a$$

- $\alpha^a \in \mathbb{R}^+$ Multiplicative coefficient for the latency
- $\beta^a \in \mathbb{R}^+$ Constant latency of an arc a

Description of the problem

A problem that mixes several sub problems :

- **Placement of service**: Assigning services to nodes with a quantity of resources for each type of service.
  Question: On which resource is the processing placed (local node, edge server, cloud server) ?
- **Routing problems**: The route from source to destination
  Question: Which path to take for a demand?
- **Quality of Service** : Quantity of bandwidth assigned to a demand.
  Question: Which QoS to give the customer ?

Roughly speaking: Assignment problem coupled with multicommodity flow (NP Hard).

An underlying question: is deepRL an efficient method that scales up ?

Literature review

- Use case descriptions [Orange Lab 2020, Premsankar 2018]
- Service assignment
  - Survey : [Ait Salah 2020]
  - RL for Service assignment [Frohlich (Gelembe) 2021]
- Network Flows
  - Linear Latency [Bonami 2017]
  - Non Linear Latency [BenAmeur 2006]
  - Multicommodity Flow [Ahija 2014]
- Resource assignment
  - Survey [Benhamiche 2019]
- RL for Combinatorial Optimization
  - Seminal Paper on TSP [Belo 2016]
  - Survey [Mazyavkina 2021]
- DeepRL for multicommodity flow
  - For a CDN [Wang 2022]

Table of Contents

Decision Variables

Decision variables:

- Service placement:
  $y_S^e \in \{0, 1\}$ (1 if S is on node e, 0 otherwise)
- Demands routing:
  x_d^a (1 if request d uses edge a, 0 otherwise)
- Quality of request ($\sim$ quantity of data) which will transit through the network
  $q_d \in \{q_{d,min}, ..., q_{d,max}\}$ the quality of demand d.

Objective function :

$$\max \sum_d q_d$$

Mathematic Program

$$\max \sum_{S, d \in S} q_d \qquad \textit{(maximize quality)}$$

$$\text{s. t.} \sum_{f \in \delta(e)} \left(\mathbf{x}_d^{(e,f)} - \mathbf{x}_d^{(f,e)} \right) = \mathbf{1}_{e=d_o} - \mathbf{y}_S^e \,, \ \forall e, \forall d \in S, \forall S \quad \textit{(routing)}$$

$$\sum_{S \in \mathcal{S}} \mathbf{y}_S^e r_S \leq \mathrm{res}^e \,, \quad \forall e \in \mathcal{E} \qquad \textit{(Resources on nodes)}$$

$$\sum_{S \in \mathcal{S}} \sum_{d \in S} \mathbf{q}_d \mathbf{x}_d^a \leq \mathrm{capa}^a \,, \quad \forall a \in \mathcal{A} \qquad \textit{(Edges capacities)}$$

$$\sum_{a \in \mathcal{A}} \mathbf{x}_d^a \left(\alpha^a \sum_{S' \in \mathcal{S}} \sum_{d' \in S'} \mathbf{q}_{d'} \mathbf{x}_{d'}^a + \beta^a \right) \leq l_d \,, \quad \forall d \in S, \forall S \in \mathcal{S}$$

$$\textit{(Demands latencies)}$$

$$\sum_{e \in \mathcal{E}} \mathbf{y}_S^e \geq 1 \,, \quad \forall S \in \mathcal{S} \qquad \textit{(Services placement)}$$

$$\mathbf{q}_d \in \{q_{d,\min}, \ldots, q_{d,\max}\}, \ \mathbf{y}_S^e \in \{0,1\}, \ \mathbf{x}_d^a \in \{0,1\}$$

Resolution of the mathematic program

Solving

- This problem is non-linear but can be linearised by introducing new variables without loss of generality.
- We obtain an Integer Linear Program
- ILP can be solved using a Solver (cplex Here)
- Solver provides the optimal solution.
- BUT do not scale.

Improvements ?

- Continuous relaxation and rounding
- Decomposition: Service placement (solved by local search) and Multi commodity flow (with Linear Progamming)

LP-Path formulation

Path formulation Principle

Pre-generate all possible paths $P \in \mathcal{P}_o$ from origins.

We introduce:

$$\lambda_P^d \in \{0, 1\}$$

which describes whether the request d uses the path P.

We reformulate the previous problem with λ.

Interest: Less variables and pre computation of the paths.

Approximation

Generate a limited number of paths and find an approximate solution.

Approximation: consider the kth shortest path for each demand.

$$\sum_{d \in S, S \in \mathcal{S}} \mathbf{q}_{d,k}^* \leq \sum_{d \in S, S \in \mathcal{S}} \mathbf{q}_d^*$$

LP-Path modelling

$$\max \sum_{S \in \mathcal{S}, d \in S} \mathbf{q}_d \qquad \text{Max quality}$$

$$\text{s. t.} \sum_{S \in \mathcal{S}} \mathbf{y}_S^e r_S \leq res^e \qquad \text{Resources on nodes}$$

$$\sum_{S \in \mathcal{S}, d' \in S} \sum_{P' \in \mathcal{P}^{d'}} \delta_{P'}^a \lambda_{P'} \mathbf{q}_{d'} \leq \text{capa}^a \qquad \text{Edges capacities}$$

$$\lambda_P \sum_{a \in P} \left(\alpha^a \sum_{S' \in \mathcal{S}} \sum_{d' \in S} \sum_{P' \in \mathcal{P}^{d'}} \delta_{P'}^a \lambda_{P'} \mathbf{q}_{d'} + \beta^a \right) \leq l^d \qquad \text{Demand latencies}$$

$$\sum_{e \in \rceil} \mathbf{y}_S^e = 1 \qquad \text{Service placement}$$

$$\mathbf{q}_d \in \{1, ..., q_{d,max}\}, \mathbf{y}_S^e \in [0,1], \mathbf{x}_d^a \in \{0,1\}$$

Table of Contents

Local search heuristic

Solution description

A solution is a vector (e_S, q_d, SP_d) where

- e_S describes the *service placement* for each service ($\in [1, |\mathcal{E}|]^S$).
- q_d describes the *quantity* for each demand ($\in [1, Q]^D$).
- SP_d describes the *shortest path* chosen (index of the path) by demand ($\in [1, MAXPCC]^D$).

Greedy Algorithm

We assume that there is a feasible solution.

- Find the feasible solution (minimal quantity and all services on cloud)
- Repeat while improvement exists
  - Increases the flows greedily
  - If possible decrases the load of most loaded arc (by changing path)
  - If possible change the placement of services

Evolutionary approach

Using CMAES

Same solution considered as above (e_S, q_d, SP_d).

CMAES description

- Consider a population of solutions.
- New candidates are generated by sampling a multivariate normal distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$
  - Recombination (crossing) means select a new mean value.
  - Mutation add a random vector perturbation with zero mean.
  - dependencies between the variables are represented by covariances

The covariance matrix adaptation (CMA) method updates the covariance matrix and the means to find the optimal solution

- We use the free CMAES solver (BBOB).
- Ceiling: at each iteration we round the continuous solution obtained to get integers for all components.

Table of Contents

RL for CO the framework

[Belo 2017] uses Deep RL to solve TSP, there is a larger use of RL for CO since.

Deep RL can be seen as an heuristic to upgrade or build solutions of CO problems.

Principle, the underlying a MDP

- State Space: the set of feasible solutions
- Action Space: the set of possible actions that modifies a solution (also the set of actions which build a neighbourhood).
- Transition are deterministic and describes the way to pass from a solution to a neighbour
- Reward: an estimate (often “ad hoc”) of the improvement induced by the new policy.

Advantage of using a Neural Network: prediction of non-seen solutions.

Description of the RL model

State space: (e_S, SP_d, q_d)

- Current node where service is placed (index)
- Current index of the Shortest Path
- Quantity of each request
- Additional information on the graph (latencies, arc loading...)

Action space:

- Explicit choice of placement node
- Explicit choice of Shortest Past (index of Path)
- modification of quantity $q_d \pm 1$.

Reward:

- -10 if action impossible
- $\max(0, q_{actual} - q_{\text{best}})$ otherwise

where q_{best} is the maximum quantity reached before

Solving

Use actor critic methods to solve the problem: optimize a vector of parameters θ that defines the policy by a SGD

$$\nabla_\theta J(\pi_\theta) = \mathbf{E}_{\pi_\theta}\Big(\sum_t \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)\Big)$$

with $A(s,a) = \sum_{t=t'}^{H} \gamma^{t-t'} r(s_t, a_t) - b(s_t)$.

Two algorithms

- PPO : big exploration, not very stable
- A2C : more exploitation, explores less

Take advantage of the exiting solvers (*stablebaseline)* that run on GPU

Table of Contents

Instances description

Instances tree shapes

- Instances that imitate a connected warehouse
- Tree-shaped like graphs
- Each user is linked with 4 neighbours (graph is not a tree)

Size	**Nodes**	**Services**	**Demands**
Tiny	6	4	8
Small	12	10	20
Medium	22	20	40
Large	32	30	40

Table 1: Characteristics of different network sizes

Results

Tiny instances

Method	Computation time	Optimality Gap	Objective
ILP exact	1s	100%	686
ILP with 10 SP	1s	100%	686
RL PPO	14.2s	85%	580
RL A2C	12.3s	69%	474
Heuristic	0.1s	77%	532
CMAES	13.7s	37%	258

Table 2: Solving time and performance on Tree_6_4 instance.

Results

Small instances

Method	Computation time	Gap	Objective
ILP exact	3s	100%	700
ILP with 10 SPP	1s	100%	700
RL PPO	10mn	45%	280
RL A2C	10mn	50%	350
Heuristic	1s	95%	676

Table 3: Solving time and performance on Tree_12_10 instance.

Results

Medium

Method	Computation time	Gap	Objective
ILP exact	9.2s	100%	(1400)
ILP with 10 SPP	—	—	—
RL PPO	18.7mn	51%	(718)
RL A2C	7mn30	46%	(650)
Heuristic	1s	98%	(1376)

Table 4: Solving time and performance on Tree_22_20 instance.

Table of Contents

Conclusion

We compare three methods to solve a service placement problem coupled with a Multi commodity flow.

- DeepRL performance is quite low but scale for large instance
- CMAES methods has average performance but do not scale with parameters e_S, q_D, SP_d when dimension is greater than 30.
- Greedy heuristic have a very good performance for very short time.

Perspective

Some comments about Deep RL

- Consider implementation carefully (GNN, Convolutional and encoders)
- Distinguish the state space and the Neural Network inputs
  - Information needed to feed the NN is not part of the state space.
  - Do not let the Neural Network learn what can be easily computed
- Pre-training is a key
  - In [Belo 2017] only the pre-trained algorithms work well.
  - Transfer learning is not obvious (especially it does not fit to every instances of a given OC problem).

One major challenge:

- Action space is naturally hierarchical (quantities depends on routing that depends on placement)

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. (1993).
Network flows: theory, algorithms, and applications.
Prentice-Hall, Inc., USA.

Ait Salaht, F., Desprez, F., and Lebre, A. (2020).
An overview of service placement problem in fog and edge computing.
ACM Computing Surveys (CSUR), 53(3):1–35.

Bello, I., Pham, H., Le, Q. V., Norouzi, M., and Bengio, S. (2016).
Neural combinatorial optimization with reinforcement learning.
arXiv preprint arXiv:1611.09940.

Ben-Ameur, W. and Ouorou, A. (2006).
Mathematical models of the delay constrained routing problem.
Algorithmic Operations Research, 1(2).

Benhamiche, A., da Silva Coelho, W., and Perrot, N. (2019).
Routing and resource assignment problems in future 5g radio access networks.
In *International Network Optimization Conference.*

Bonami, P., Mazauric, D., and Vaxès, Y. (2017).
Maximum flow under proportional delay constraint.
Theoretical Computer Science, 689:58–66.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. (2021).
Reinforcement learning for combinatorial optimization: A survey.
Computers & Operations Research, 134:105400.

OrangeLab (2020).
Dedicat 6g dynamic coverage extension and distributed intelligence for human centric applications with assured security, privacy and trust: from 5g to 6g.

Premsankar, G., Di Francesco, M., and Taleb, T. (2018).
Edge computing for the internet of things: A case study.
IEEE Internet of Things Journal, 5(2):1275–1284.

Svorobej, S., Bendechache, M., Griesinger, F., and Domaschka, J. (2020).
Orchestration from the Cloud to the Edge, pages 61–77.
Springer International Publishing.