NYU TANDON SCHOOL OF ENGINEERING

Latency Focused Function Placement and Request Routing in Multi Access Mobile Edge Computing Networks

Mustafa F. Ozkoc(mfo254)

Chen Li (cl5089)

May 19th

Overview

- **Motivation**
  - Achieving **U**ltra **R**eliable **L**ow **L**atency **C**ommunications (URLLC)
- **Key enablers of 5G networking**
  - **M**obile **E**dge **C**omputing (MEC)
  - **N**etwork **F**unction **V**irtualization (NFV)
  - **S**oftware **D**efined **N**etworks (SDN)
- **Issues, Problems**
  - MEC resources are limited, distributed -> constraints on storage, computational power, bandwidth
  - Varying wireless links
  - Coupled functions
  - Delay Limited Services
- **Possible Solutions**
  - Optimization framework to allocate services and functionalities while satisfying service requirements
    - virtualization of network functions based on physical constraints of the network.
- **Benefits**
  - Proof of achievability of services with tight quality of service (QoS) requirements

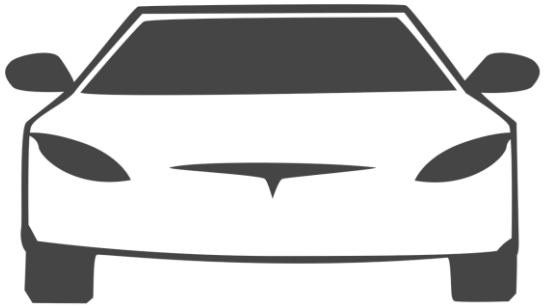
Motivation

Applications requiring ultra reliability and/or low latency

Augmented Reality (AR)

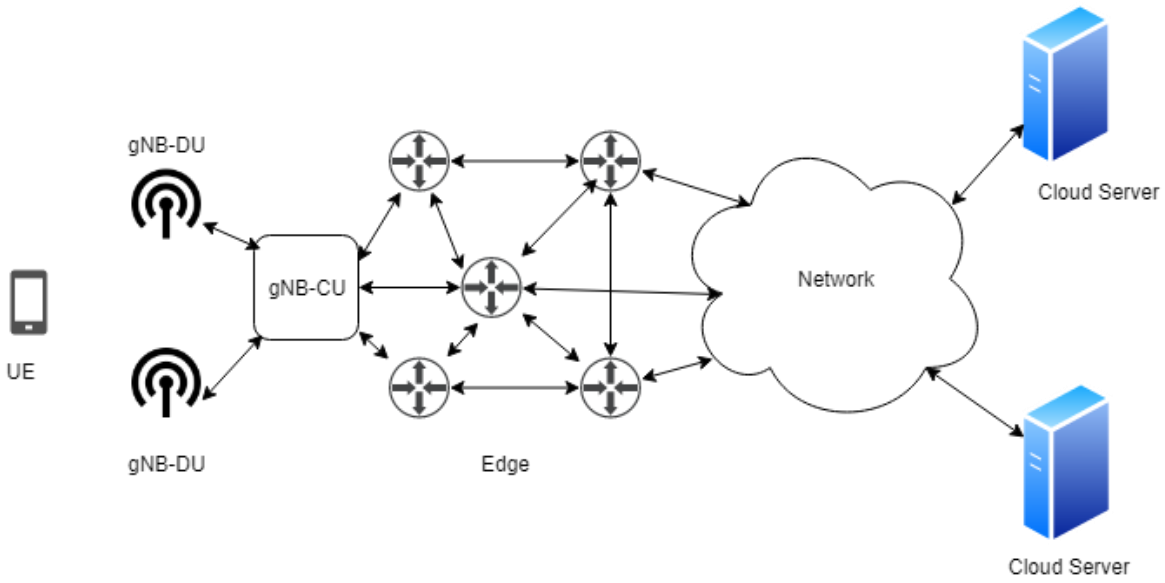
Virtual Reality (VR) Applications

Remote Control



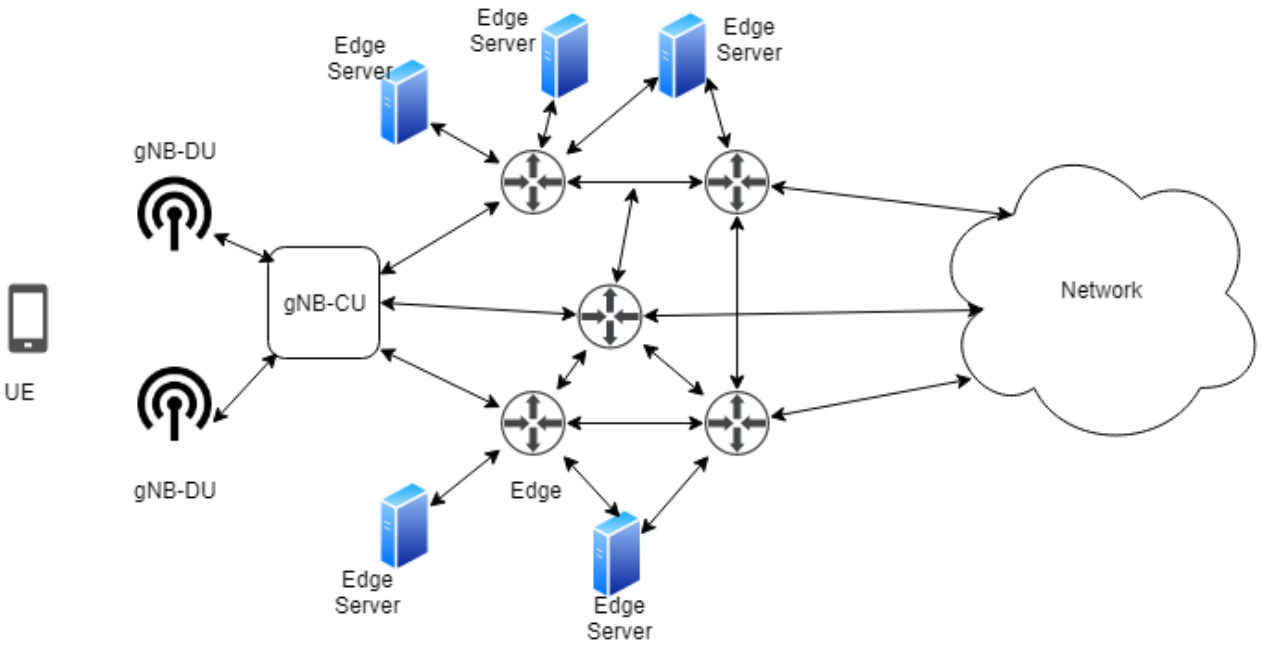
Autonomous Driving

Mobile Edge Computing (MEC)



A service has to go through the core network

- Network congestion is reduced
- Quality of service is improved
- Reduces the network operation cost



A service can be served at the edge

- User mobility and service migration issues
  - Migration cost vs transmission cost
- Stress on control plane functions
- Costly compared to consolidated data center
- Limited resources compared to centralized cloud

Issues and Problems

- MEC resources are limited [1]
  - Computational power constraint
    - Can compute limited number of requests at a given time
  - Bandwidth constraints
    - Connecting links have bandwidth limitations hence can communicate with limited number of user equipment hence limited number of requests can be accepted
  - Storage constraints [2]
    - Can not store every network function, each node can store a subset of functions hence limited number of requests can be satisfied
    - This resource can be shared between applications [3]

[1] P. Mach and Z. Becvar, "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628-1656, 2017.

[2] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, L. Tassiulas, " Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks, arXiv:1901.08946

[3] T. He, H. Khamfroush, S. Wang, T. La Porta and S. Stein, "It's Hard to Share: Joint Service Placement and Request Scheduling in Edge Clouds with Sharable and Non-Sharable Resources," *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, Vienna, 2018, pp. 365-375.

Possible Solution

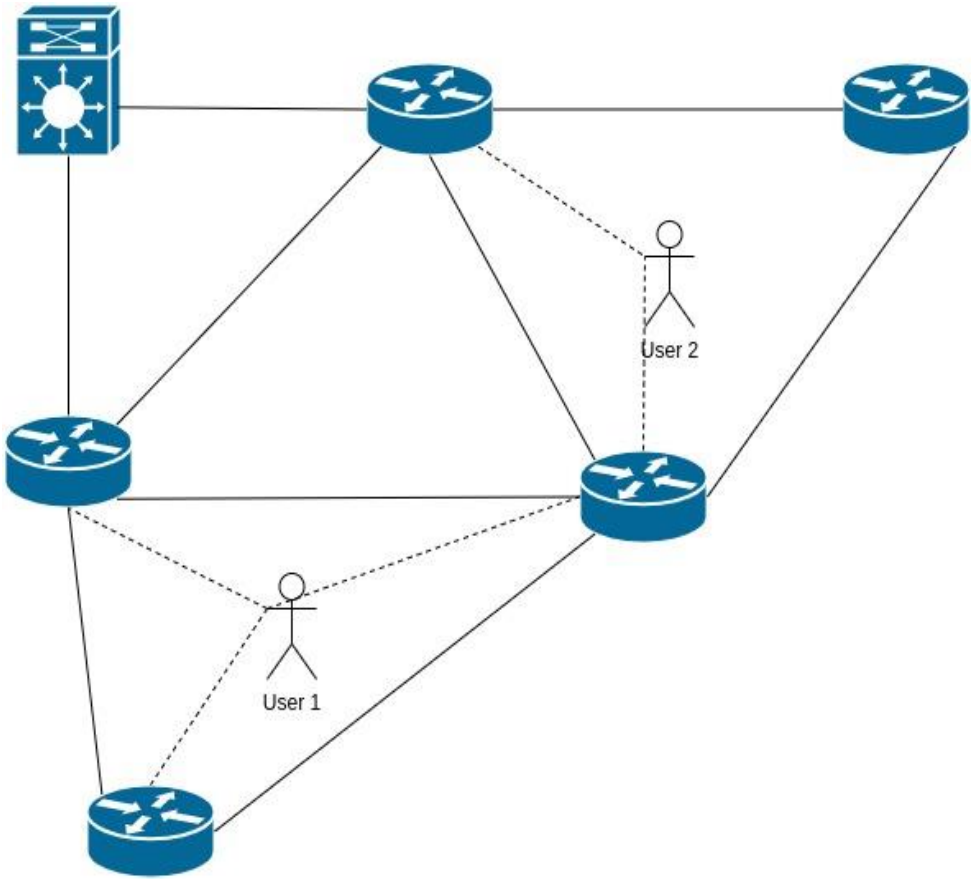
- A service placement solution is required subject to service and physical substrate constraints.
  - Delay sensitive services require careful network planning for data plane and control plane
- What is already in the literature ?
  - [2] Focuses on node physical constraints but neglects QoS constraints and control plane functions
  - [5] Focuses on VNF placements in order to reduce bandwidth but neglects computation constraints
  - [6] Focuses on energy saving by running smallest possible number of computational node but neglects physical constraints of each node and control plane functionality

[2] K. Poularakis, J. Llorca, A. M. Tulino, I. Taylor, L. Tassiulas, " Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks, arXiv:1901.08946

[5] M. Pozza, A. Patel, A. Rao, H. Flinck, S. Tarkoma, " Composing 5G Network Slices by Co-locating VNFs in uslices", *IFIP Networking,* 2019

[6] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros and G. Pavlou, "Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications," in *IEEE Transactions on Network and Service Management*, vol. 15, no. 1, pp. 475-488, March 2018.

System Model



Base Stations

- N Base Stations
- C_n : Computational Capacity
- M_n : Memory/Storage Capacity

Services

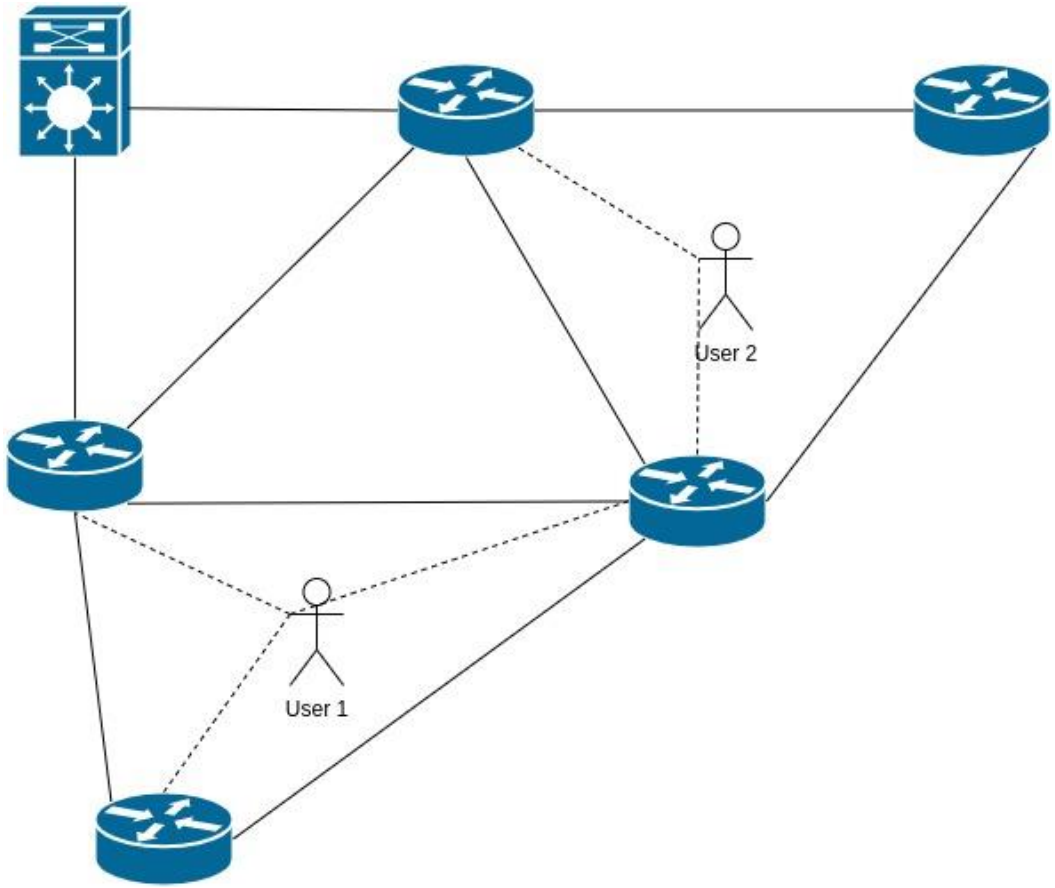
- A library of services in set S are offered
- c_s : capacity requirement
- m_s : memory/storage space requirement
- h_s : traffic volume
- t_s : maximum tolerable latency
- w_s : pricing

Users

- A user u can access to network through nodes N_u a subset of N
- Each user u requests a service s_u from the network

NYU TANDON SCHOOL OF ENGINEERING

Formulation



Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

NYU TANDON SCHOOL OF ENGINEERING

Formulation

Objective Function

Two possible design choices

1. Minimize the network operation & installation cost for given user-service paris.
2. Maximize the number of served user-service paris for a given network configuration.

$$F = \sum_{u \in \mathcal{U}} w_{s_u} \beta_u - \sum_{u \in \mathcal{U}} \sum_{e \in \mathcal{E}} u_{ue} \xi_e y_e$$

Generate the largest possible revenue from a centralized cloud server and a MEC network topology pair.

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

NYU TANDON SCHOOL OF ENGINEERING

Formulation

Constraints

1. A user can be served at most one location or neglected:

$$\sum_{n \in \mathcal{N} \cup \{l\}} x_{nu} \leq 1, \forall u \in \mathcal{U}$$

2. If a user is to be served the user needs to access the network through any of its access nodes, otherwise it shouldn't access to the network and waste resources at access nodes:

$$\sum_{n' \in \mathcal{N}_u} y_{n'u} = \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}$$

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

NYU TANDON SCHOOL OF ENGINEERING

Formulation

Constraints

3. The flow generated by user u should be allocated to links if and only if that user is to be served:

$$u_{ue} \leq \bar{h}_{s_u} \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}, e \in \mathcal{E}$$

4. Each link has a capacity bound. The total flow realized on a link should be less than this capacity:

$$\sum_{u \in \mathcal{U}} u_{ue} \bar{h}_{s_u} \leq c_e, \forall e \in \mathcal{E}$$

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

NYU TANDON SCHOOL OF ENGINEERING

Formulation

Constraints

5. The total delay experienced over the network by a service flow should be less than end to end latency requirement of the service:

$$\sum_{e \in \mathcal{E}} u_{ue} l_e \leq \bar{t}_{s_u} \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}$$

if the service will not be served this constraint is trivially satisfied by the constraint (4)

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

Formulation

Constraints

6. For any node in the network the flow conservation should be satisfied, i.e the summation of total incoming flows and flows generated at that node should be the summation of total outgoing flows and flows destined to that node:

$$\sum_{e \in \mathcal{E}} a_{ev} u_{ue} - \sum_{e \in \mathcal{E}} b_{ev} u_{ue} = (y_{vu} - x_{vu}) \bar{h}_{s_u}, \forall v \in \mathcal{N} \cup \{l\}$$

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where $n' \in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

Indices :

- u = {1,..,U} : Users
- n = {1,..,N, l } : Nodes, where l is the centralized cloud
- e = {1,...,E} : links between nodes, directed

Constants

- c_e : Capacity of link w
- l_e : latency introduced at link e, can be formulated as a function of load in future. $1/(c_e - y_e)$
- a_{ev} = 1 if link e originates at node v; 0, otherwise
- b_{ev} = 1 if link e terminates at node v; 0, otherwise
- W_{su} = price of the service requested by user u

Variables

- x_{nu} : binary variable. 1 if the request of user u is served at node n, 0 otherwise.
- $y_{n'u}$: binary variable. 1 if the user u access the network through node n' where n' $\in N_u$, 0 otherwise
- u_{ue} : binary variable. 1 if the link e is used while serving the user u, 0 otherwise. (can be changed to volume if we want to allow flow splitting.)

maximize

$$F = \sum_{u \in \mathcal{U}} w_{s_u} \beta_u - \sum_{u \in \mathcal{U}} \sum_{e \in \mathcal{E}} u_{ue} \xi_e y_e$$

Subject to

$$\sum_{n \in \mathcal{N} \cup \{l\}} x_{nu} \leq 1, \forall u \in \mathcal{U}$$

$$\sum_{n' \in \mathcal{N}_u} y_{n'u} = \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}$$

$$u_{ue} \leq \bar{h}_{s_u} \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}, e \in \mathcal{E}$$

$$\sum_{u \in \mathcal{U}} u_{ue} \bar{h}_{s_u} \leq c_e, \forall e \in \mathcal{E}$$

$$\sum_{e \in \mathcal{E}} u_{ue} l_e \leq \bar{t}_{s_u} \sum_{n \in \mathcal{N} \cup \{l\}} x_{nu}, \forall u \in \mathcal{U}$$

$$\sum_{e \in \mathcal{E}} a_{ev} u_{ue} - \sum_{e \in \mathcal{E}} b_{ev} u_{ue} = (y_{vu} - x_{vu}) \bar{h}_{s_u}, \forall v \in \mathcal{N} \cup \{l\}$$

Heuristic 1
Bottom up approach

Heuristic 2
Top to bottom approach

Basic idea

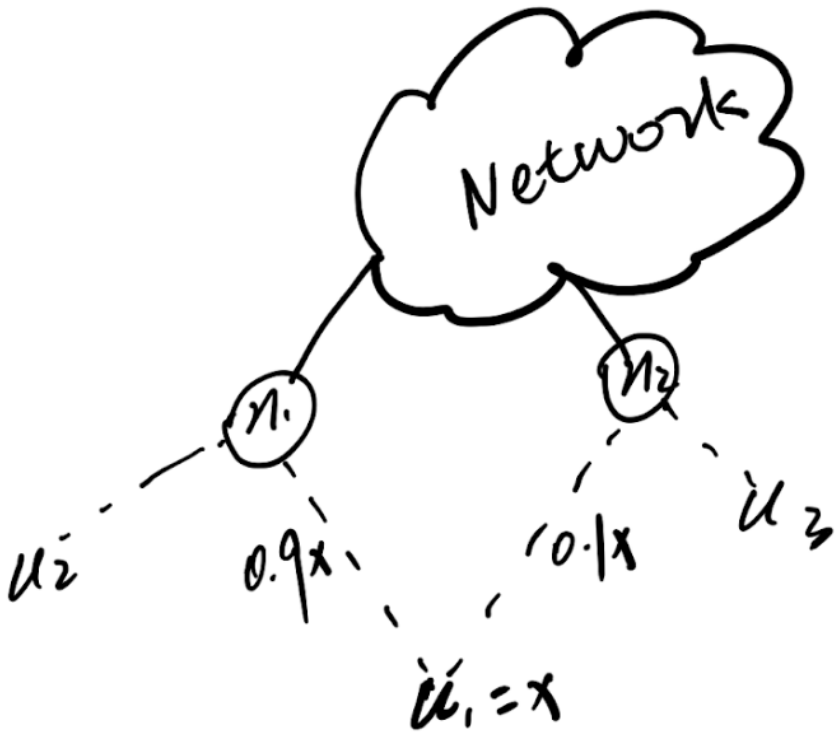
Iteration with sub-mixed integer programming.

More integer variables, more time the algorithm will take. We can make iterative procedures to optimize the objective function with less integer variables.

In our model, we have binary variable: access_indicator, link_indicator and server_indicator. But when we fix link_indicator and access_indicator to binary variables, the server_indicator will be binary automatically.

So we fix link_indicator to be binary, and get ‘binary’ access_indicator iteratively. During the iteration, we add more constraints to the model according to the output of each iteration to gradually get the mixed integer solution.

Algorithm



The most intuitive idea is to fix access_indicator =1 as soon as they become 1 in the iterative procedure.

How about < 1?
Increase 0.9 to 1?
(may cause infeasible solution)

Heuristic 2

Idea:

Try to increase these majority part of flow to one by reducing the minority to zero, which will guarantee the feasible solution.

If majority of u_i came to 1 → Fix to 1.
If not, sum(all flow of u_i) < 1 → drop off

Successive drop off will terminate the iteration.

Algorithm 2 Heuristic 2

C_1={ } constraints to fix variable to 1
C_0={ } constraints to fix variable to 0
terminate_threshold = 2
successive_drop = 0
solve sub-mip.run
$C_1 = C_1 \cup \{(n_0,u_0)\}$ if access_indicator$[n_0,u_0]$=1
while successive_drop < terminate_threshold **do**
  solve sub_mip.run
  $C_1 = C_1 \cup \{(n_i,u_i)\}$ if access_indicator$[n_i,u_i]$=1
  successive_drop = 0
  findmax_exceptOne{access_indicator}
  if $\sum_{n\in N}$access_indicator$[n,u_{max}] < 1$ **then**
    $C_0 = C_0 \cup \{(u_{max})\}$
    successive_drop++
  else
    min_pos = min_pos corresponding to u_{max}
    $C_0 = C_0 \cup \{(\text{min\_pos})\}$
    successive_drop = 0
  end if
end while
$C_0 = C_0 \cup$ {all of the pos whose value $\in (0,1)$}

Two important properties:

Property 1: We just drop off user x if and only if it has the lowest value on the access node of the flow's majority part.

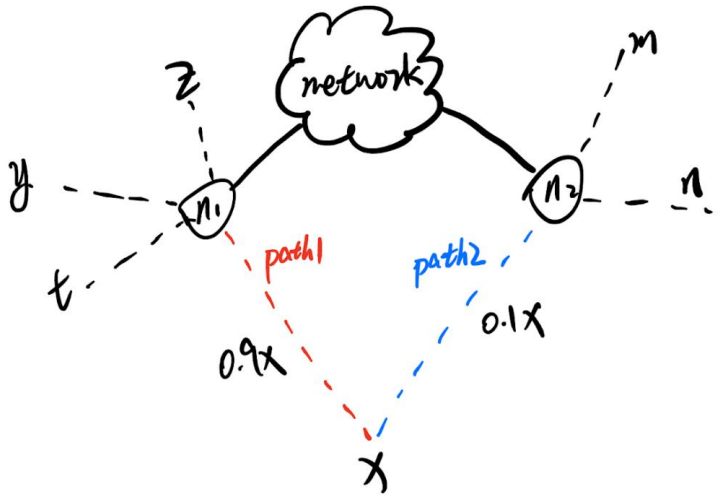
Property 2:For these user who is not drop off. If the shortest path can not accommodate this user, it will choose the second shortest path. If the second shortest path can not do either, then the third …

Sketch proof on two splits:

Split happens when the available shortest path(SP) can not fully accommodate one user.

In our model:

Each node can serve at least one service. Due to each node can serve all services, so service type does not matter for node. The value of each service matters. →Only the Higher value flow effects the lower value flow



Two cases:

Path1 is the SP:

X.v = min{V_{n1}}

Decrease 0.1x to 0, then due to X.v = min{V_n1}, 0.9x can not increase(not infect y,z,t).

sum{flow of user x}<1 drop off in the future iteration. →**property 1**

Path 2 is the SP:

X.v = min{V_{n2}}

if X.v = max{V_{n1}}

decrease 0.1x to 0 will have 0.9x→ (0.9+0.1)x →**property 2**

if X.v = min{V_{n1}}

decrease 0.1x to 0 will not infect y,z,t .Then 0.9x will be kept and drop off in the future iteration. → **property 1**

else(y.v<z.v<x.v<t.v)

decrease 0.1x to 0 will lead to 0.9x→(0.9x+wx)

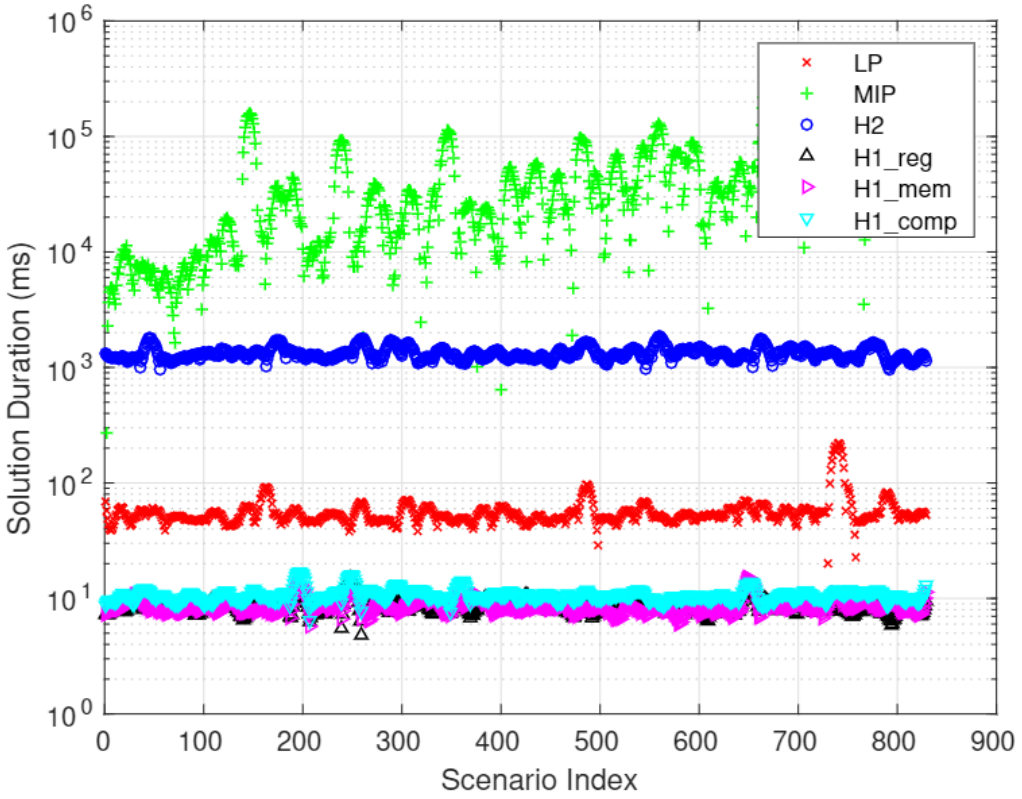
First result: 0.9x→ (0.9+0.1)x →**property 2**

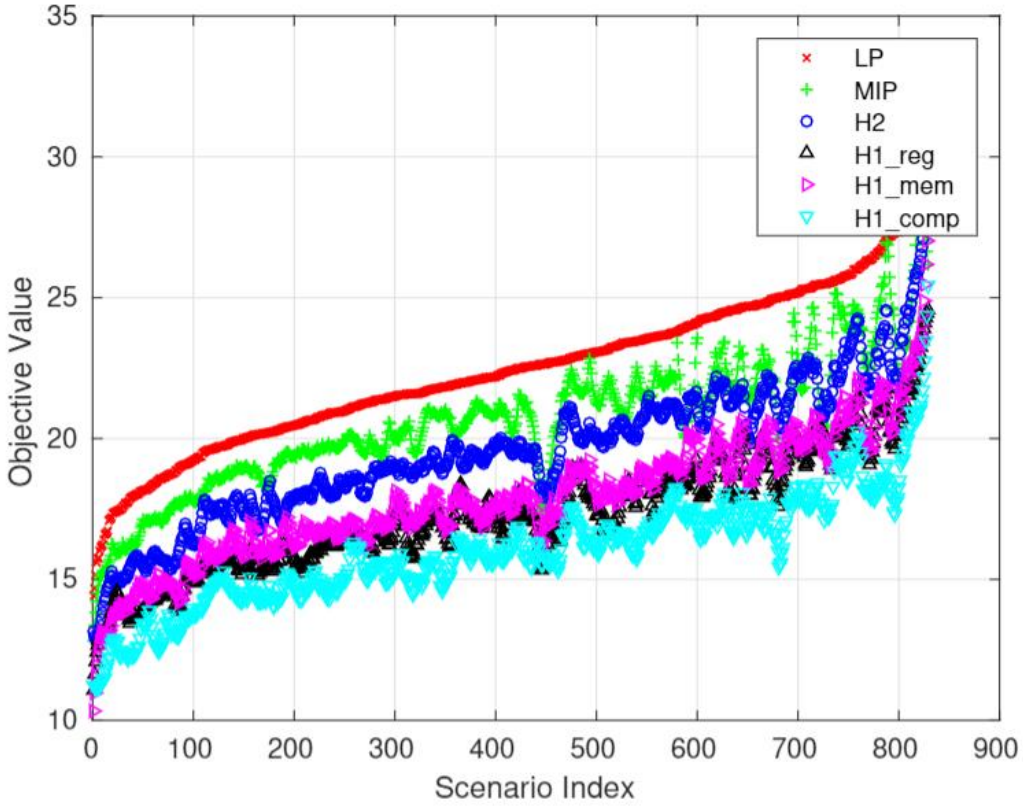
Second result: w<0.1(after remove y and z on node1→

X.v = min{V_{n1}}) →(0.9x+wx)<1→

drop off→**property 1**

Experiments





Summary

Pros:

- Achieve good approximation for mixed integer programming with much less computation.
- Solution duration is much more stable than mip's.
- Adjustable(parameters, terminate condition)

Cons:

- Still mixed integer programming in each iteration(LP in future work)

Thank you!

Q&A