Optimization Framework for Resource Management of Mobile Edge Computing Networks

Bin Xiang, Jocelyne Elias, Fabio Martignon, Elisabetta Di Nitto

Outline

Outline

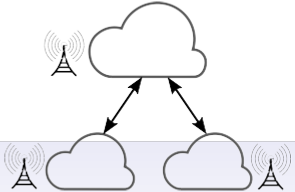
Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network.
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

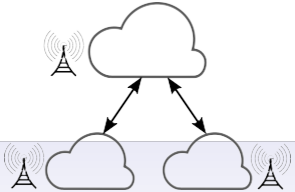
- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network.
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

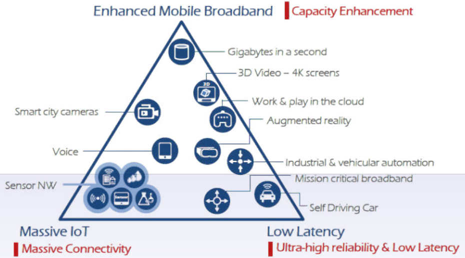


[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network.
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

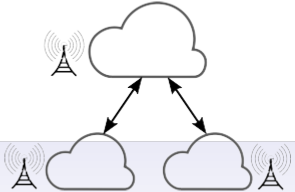


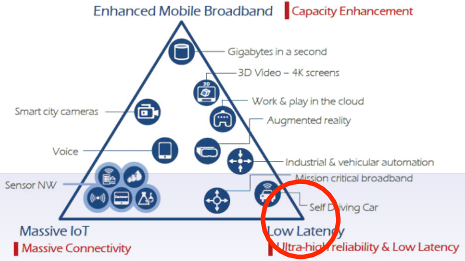


[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network.
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

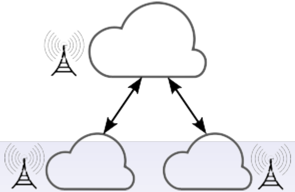


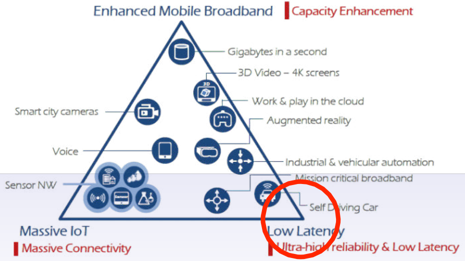


[1]W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network. autonomous driving: $< 1ms$[1]
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

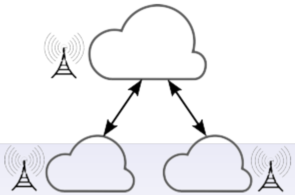


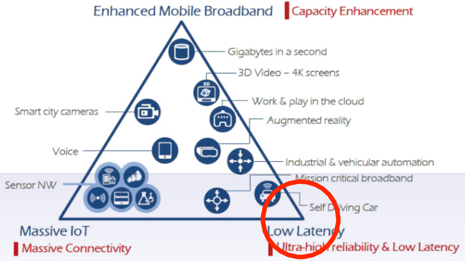


[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network. autonomous driving: $< 1ms$[1]
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.

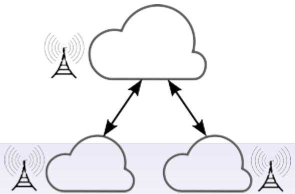


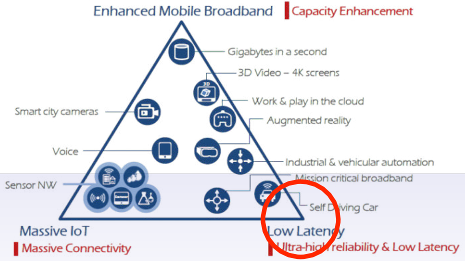


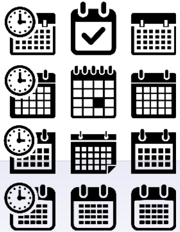
[1] W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network. autonomous driving: $< 1ms$[1]
- MEC make it possible to simultaneously address the stringent latency requirements of critical services and ensure efficient network operation and service delivery.
  - *workload offloading, network planning, network slicing, service placement, etc.*
- Users' demands show certain flexibility in terms of tolerable starting and ending service time.







[1]W. Xiang, K. Zheng, and X. S. Shen, *5G mobile communications*. Springer, 2017.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Proposals:

- leverage the cooperation among interconnected multiple MEC units;
- jointly optimize resources considering multiple aspects of network operations to minimize latency, costs, and maximize profit.

Problems & Proposals

Problems of MEC:

- provides limited computational and storage resources by design;
- does not guarantee the latency requirements of services during peak hours when serving large amount of tasks from users with high demands;
- requires significant investments from both network operators and service providers in terms of deploying, operating and managing edge clouds.

Proposals:

- leverage the cooperation among interconnected multiple MEC units;
- jointly optimize resources considering multiple aspects of network operations to minimize latency, costs, and maximize profit.

State of the Art

- Workload Offloading:[2]
  - They consider the contexts of single MEC or multiple MECs without
  - interconnection We consider multiple MECs with arbitrary topology
- Network Planning:[3]
  - They study nodes placement and the resource configuration
  - We study joint offloading of workloads, slicing and planning of resources
- Request/Resource Scheduling:[4]
  - They focus on the problems without processing or routing
  - aspects We jointly optimize admission, offloading, scheduling and routing

---

[2] C.-F. Liu, M. Bennis, M. Debbah, *et al.*, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, 2019.

[3] A. Santoyo-Gonz´alez and C. Cervello´-Pastor, "Latency-aware cost optimization of the service infrastructure placement in 5G networks," *J. Netw. Comput. Appl.*, vol. 114, pp. 29–37, 2018.

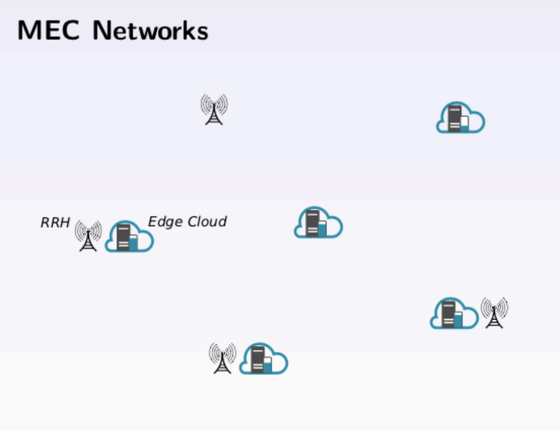
[4] J. Meng, H. Tan, X.-Y. Li, *et al.*, "Online deadline-aware task dispatching and scheduling in edge computing," *IEEE*

Contributions

- Joint Slicing Network and Edge Computing Resources
  - B. Xiang, J. Elias, F. Martignon, E. Di Nitto, "Joint network slicing and mobile edge computing in 5G networks," in *IEEE International Conference on Communications (ICC)*, 2019
- Joint Resource Planning and Slicing for Network and Edge Computing
  - B. Xiang, J. Elias, F. Martignon, E. Di Nitto, "Joint planning of network slicing and mobile edge computing: Models and algorithms," *IEEE Transactions on Cloud Computing, August 2021*
- Resource Calendaring for Mobile Edge Computing
  - B. Xiang, J. Elias, F. Martignon, E. Di Nitto, "Resource calendaring for mobile edge computing in 5G networks," in *IEEE International Conference on Communications (ICC)*, 2021
  - B. Xiang, J. Elias, F. Martignon, E. Di Nitto, "Resource calendaring for mobile edge computing: Centralized and decentralized optimization approaches," *Elsevier Computer Networks, Aug. 2021*
  - B. Xiang, J. Elias, F. Martignon, E. Di Nitto, "A Dataset for Mobile Edge Computing Network Topologies," *Elsevier Data in Brief, November 2021*

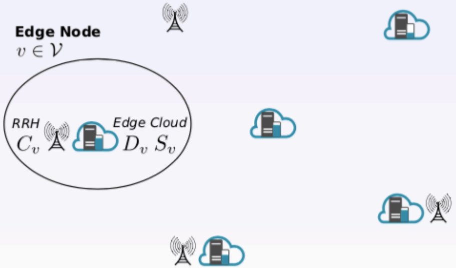
Outline

System Architecture

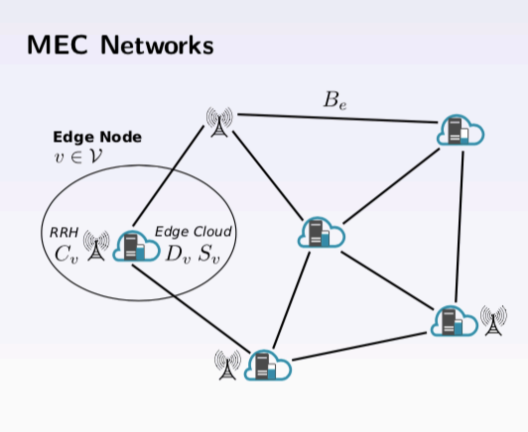


System Architecture

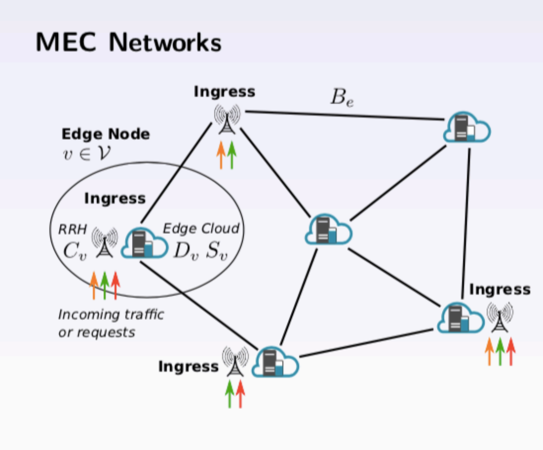
MEC Networks



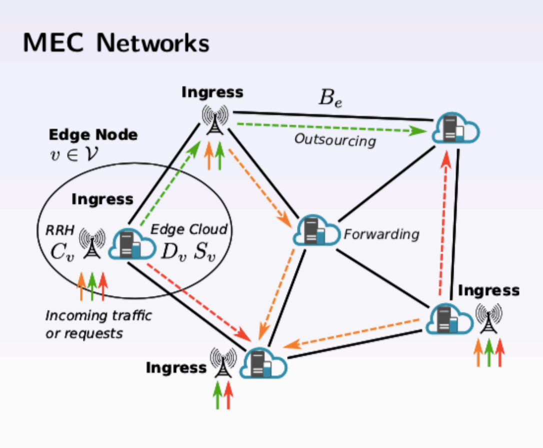
System Architecture



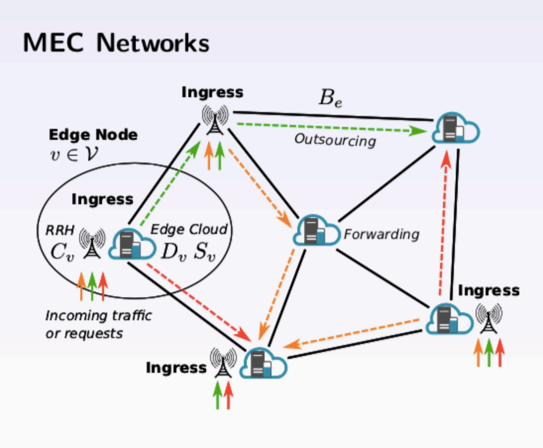
System Architecture



System Architecture



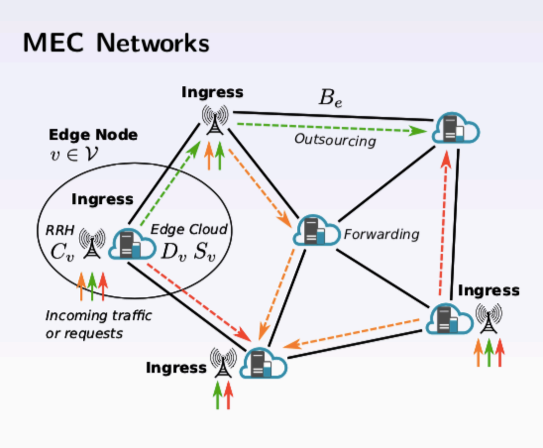
System Architecture



Assumptions

- Traffic is aggregated by types associated with different requirements and can be split and processed on all edge nodes;
- Network, computation and storage resources can be sliced to host traffic segments.
- Network, link and processing latency are modeled based on M/M/1 queueing process.

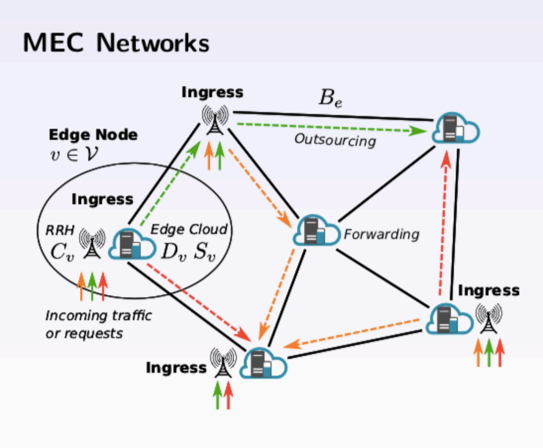
System Architecture



Assumptions

- Traffic is aggregated by types associated with different requirements and can be split and processed on all edge nodes;
- Network, computation and storage resources can be sliced to host traffic segments.
- Network, link and processing latency are modeled based on M/M/1 queueing process.

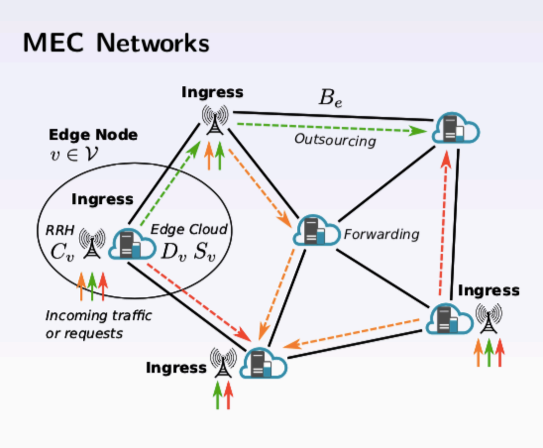
System Architecture



Assumptions

- Traffic is aggregated by types associated with different requirements and can be split and processed on all edge nodes;
- Network, computation and storage resources can be sliced to host traffic segments.
- Network, link and processing latency are modeled based on M/M/1 queueing process.

System Architecture



Assumptions

- Traffic is aggregated by types associated with different requirements and can be split and processed on all edge nodes;
- Network, computation and storage resources can be sliced to host traffic segments.
- Network, link and processing latency are modeled based on M/M/1 queueing process.

System Architecture

Resource management in MEC networks

	Edge Slicing	*Edge Planning*	*Edge Scheduling*
Resource:			
Wireless network (C_v)	√	√	√
Computation (D_v)	√	√	√
Storage (S_v)			√
Link (B_e)	√	√	√
Time domain			
Ingress node	Single	Multiple	Multiple
Topology $G(V, E)$	Hierarchical	Arbitrary	Arbitrary
Operation:			
Slicing	C_v , D_v	C_v , D_v	B_e , D_v , S_v
Offloading	√	√	√
Routing Planning	√	√	√
Admission		√	√
Scheduling			√
Optimization:			
Object	Traffic	Traffic	Reques
Target	Latency	Latency, operation costs	t Profit
Constraints	Capacities, latency	Capacities, latency, budget	Capacities, life cycle
Approach	Centralized	Centralized	Centralized + Decentralized

Outline

Problem Formulation

$\mathcal{P}0$:

$$\min_{c_n, b_{n,v}, q_{n,v}, r_{n,v}} \sum_{n\in\mathcal{N}} \left\{ T_n^{Network} + \max_{v\in\mathcal{V}} \left\{ T_{n,v}^{Processing} + T_{n,v}^{Link} \right\} \right\},$$

s.t. Tolerable latency for each traffic type (τ_n, $n \in \mathcal{N}$),

Wireless network capacity (C_v, $v = ingress$),

Computation capacity of each node (D_v, $v \in \mathcal{V}$),

Link capacity (B_e, $e \in \mathcal{E}$).

Decision variables:

- c_n: Slice of the network capacity C for traffic type $n \in \mathcal{N}$ (*slicing*)
- $b_{n,v}$: Indicator of whether traffic $n \in \mathcal{N}$ is processed on node $v \in \mathcal{V}$ (*offloading*)
- $q_{n,v}$: Percentage of traffic $n \in \mathcal{N}$ processed on node $v \in \mathcal{V}$ (*offloading*)
- $r_{n,v}$: Percentage of computation capacity D_v, $v \in \mathcal{V}$ sliced for traffic $n \in \mathcal{N}$ (*provisioning*)

Problem Formulation

- $\mathrm{P0}$ is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard[5].
  - *Branch and Bound* method can be exploited, but:
    - $\mathrm{P0}$ contains many difficult indicator constraints;
    - Computing time exponentially increases w.r.t. problem size;
- Reformulation + Heuristic + B&B:
  - Transform $\mathrm{P0}$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($\mathrm{P1}$).
  - Propose *Sequential Fixing* and *Greedy* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

Problem Formulation

- $\mathcal{P}0$ is a mixed-integer nonlinear programming (MINLP) problem, which is $\mathcal{NP}$-hard[5].
  - *Branch and Bound* method can be exploited, but:
    - $\mathcal{P}0$ contains many difficult *indicator* constraints;
    - Computing time exponentially increases w.r.t. problem size;
- Reformulation + Heuristic + B&B:
  - Transform $\mathcal{P}0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($\mathcal{P}1$).
  - Propose *Sequential Fixing* and *Greedy* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

Problem Formulation

- $\mathcal{P}0$ is a mixed-integer nonlinear programming (MINLP) problem, which is $\mathcal{NP}$-hard[5].
  - *Branch and Bound* method can be exploited, but:
    - $\mathcal{P}0$ contains many difficult *indicator* constraints;
    - Computing time exponentially increases w.r.t. problem size;
- Reformulation + Heuristic + B&B:
  - Transform $\mathcal{P}0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($\mathcal{P}1$).
  - Propose *Sequential Fixing* and *Greedy* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

Problem Formulation

- $\mathcal{P}0$ is a mixed-integer nonlinear programming (MINLP) problem, which is $\mathcal{NP}$-hard[5].
  - *Branch and Bound* method can be exploited, but:
    - $\mathcal{P}0$ contains many difficult indicator constraints;
    - Computing time exponentially increases w.r.t. problem size;
- Reformulation + Heuristic + B&B:
  - Transform $\mathcal{P}0$ into a mixed-integer quadratically constrained programming (MIQCP) problem $(\mathcal{P}1)$.
  - Propose *Sequential Fixing* and *Greedy* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

Heuristics

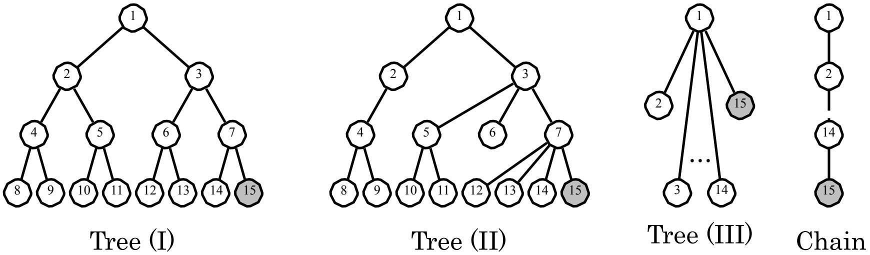
Main procedures of Sequential Fixing:

Algorithm 1 *Sequential Fixing*

1: **Relax** $b_{n,v}$ to continuous $\tilde{b}_{n,v}$ in $\mathcal{P}1$, then **solve** $\tilde{b}^*_{n,v}$;
2: **Rank** nodes ($\mathcal{V}$) by descending $\sum_{n\in\mathcal{N}} \tilde{b}_{n,v}$, and **keep** top $\mathcal{K} \subset \mathcal{V}$;
3: **Rank** traffic types ($\mathcal{N}$) by descending rate and tolerable latency;
4: **Allocate** nodes ($\mathcal{K}$) to types ($\mathcal{N}$) in order and repeatedly;
5: **Set** on/off the corresponding variables $b_{n,v}$ in original $\mathcal{P}1$.

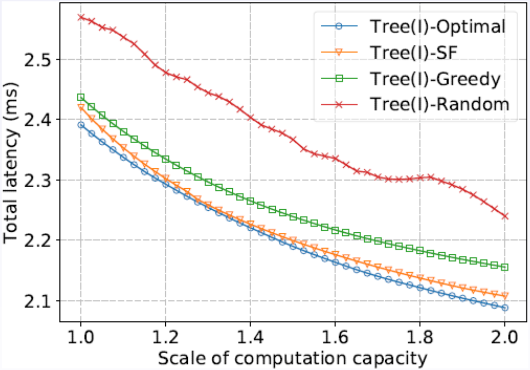
Evaluation

Network Topologies:

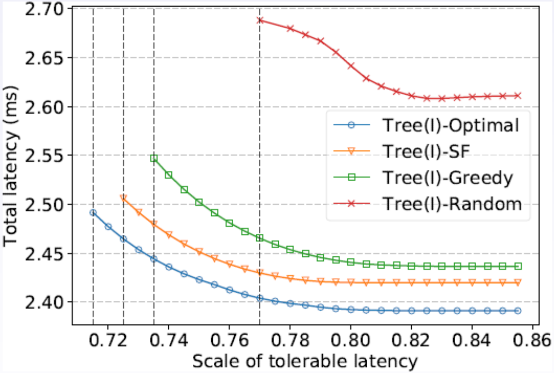


Evaluation

Scaling computation capability D_v, tolerable latency τ_n



(a) Scaling D_v ($|\mathcal{N}| = 3$, $|\mathcal{V}| = 15$)

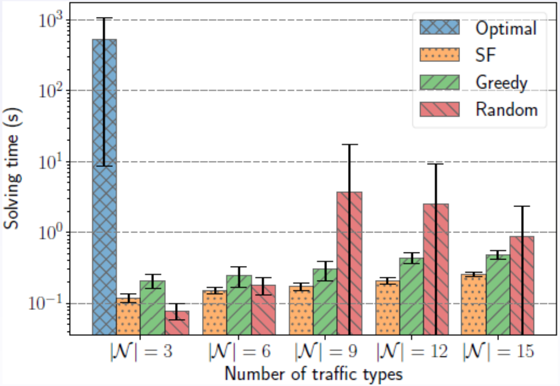


(b) Scaling τ_n ($|\mathcal{N}| = 3$, $|\mathcal{V}| = 15$)

2. capacity 5

Evaluation

Computing time



Outline

Problem Formulation

$\mathcal{P}0$:

$$\min_{\substack{c^{kn}, b_v^{kn}, q_v^{kn}, \\ r_v^{kn}, \delta_v^a, \mathcal{R}_v^{kn}}} \sum_{n \in \mathcal{N}} \max_{k \in \mathcal{K}} \left\{ T_{kn}^{Network} + T_{kn}^{Process+Link} \right\} + w \sum_{v \in \mathcal{V}} J_v^{Operation},$$

s.t. Tolerable latency for each traffic type (τ_n, $n \in \mathcal{N}$),
Wireless network capacity (C_k, $k \in \mathcal{K}$),
Computation capacity and planning budget (D_v, $v \in \mathcal{V}$, P),
Link capacity (B_e, $e \in \mathcal{E}$).

Decision variables:

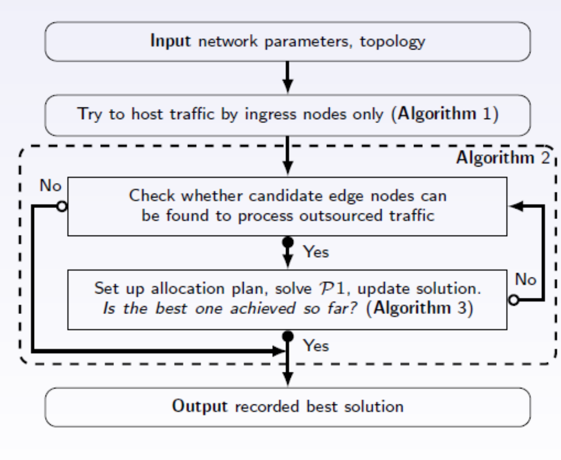
- c^{kn}: Slice of the network capacity for traffic kn (*slicing*)
- b_v^{kn}: Whether traffic kn is processed on node v (*offloading*)
- q_v^{kn}: Percentage of traffic kn processed on node v (*offloading*)
- r_v^{kn}: Percentage of v's computation capacity sliced for processing traffic kn (*provisioning*)
- δ_v^a: Decision for planning computation capacity on node v (**planning**)
- $\mathcal{R}_v^{kn}$: Set of links for routing the traffic piece q_v^{kn} from ingress k to node v (**routing**)

Problem Formulation

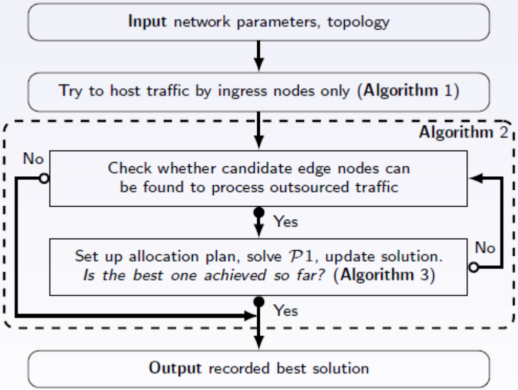
- $\mathcal{P}0$ is a mixed-integer nonlinear programming (**MINLP**) problem, which is $\mathcal{NP}$-hard[5].
  - $\mathcal{P}0$ contains many difficult indicator constraints;
  - **Variables in $\mathcal{P}0$ are "intertwined", e.g., routing and offloading;**
- **Reformulation + Heuristic + B&B:**
  - Transform $\mathcal{P}0$ into a mixed-integer quadratically constrained programming (**MIQCP**) problem ($\mathcal{P}1$).
  - Propose *Neighbor Exploration and Sequential Fixing* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

Neighbor Exploration and Sequential Fixing



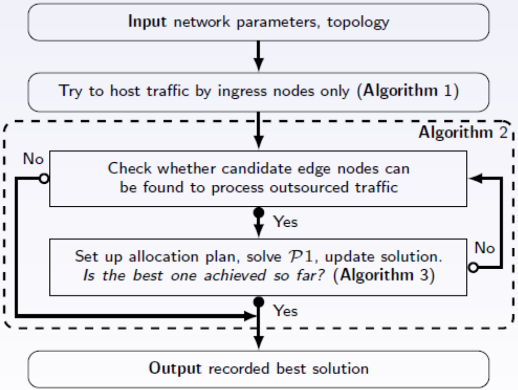
Neighbor Exploration and Sequential Fixing



Algorithm 1 *Attempt of serving traffic with ingress nodes*

$D_k^e = D^m - \sum_{n \in \mathcal{N}} \lambda^{kn}, \ \forall k;$
$\mathcal{K}^u = \{k \in \mathcal{K} \mid D_k^e \leqslant 0\};$ ▷ Unable to host traffic
for $k \in \mathcal{K}^u$ **do**
    Find neighbor ingress nodes to *cover* D_k^e;
    if *found* **then** Add them to Q_k by ascending hop;
Rank $\mathcal{N}$ as $\mathcal{N}_k$ by descending $(\lambda^{kn}, \tau_n), \forall k;$
if $\mathcal{K}^u = \varnothing$ **or** $\bigwedge_{k \in \mathcal{K}^u}(|Q_k| > 1)$ **then**
    Allocate Q_k to $\mathcal{N}_k$ in order and repeatedly, $\forall k$;
    Solve $\mathcal{P}1$ to obtain objective function value $O_{\mathcal{P}1}$;
    if $O_{\mathcal{P}1} > 0$ **then** $O_t = O_{\mathcal{P}1}$;

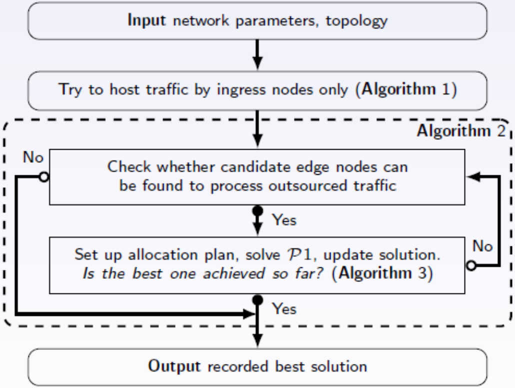
Neighbor Exploration and Sequential Fixing



Algorithm 2 *Priority searching of computation candidates*

1: **Rank** ingress nodes as $\mathcal{K}^s$, $\hat{k} = \mathcal{K}^s(0)$;
2: **while** $|\bigcup_{k \in \mathcal{K}} Q_k| < \left\lfloor \frac{P}{min(L_a)} \right\rfloor$ **and** $\mathcal{K}^s \neq \varnothing$ **do**
3: **Search** candidates $\mathcal{B}$ for $\hat{k}$ from multi-hop neighbors considering estimated computation capacity;
4: Rank $\mathcal{B}$, $v' = \mathcal{B}(0)$;
5: **Spread** v' to help other $\mathcal{K}^s \backslash \{\hat{k}\}$ and update Q_k;
6: **Update** next searching target $\hat{k}$;
7: **if** $\hat{k}$ needs *help* **then continue**;
8: Run (**Algorithm** 3) to obtain O_t;
9: **Return** O_t;

Neighbor Exploration and Sequential Fixing

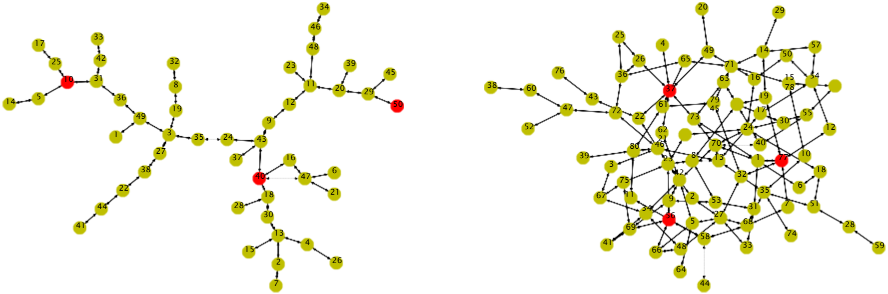


Algorithm 3 *Allocating resources and obtaining solution*

Relax $b_v^{kn}, \delta_v^a, \gamma_e^{kn,v}$ to continuous ones ($\mathcal{P}1 \to \tilde{\mathcal{P}}1$);
Allocate Q_k to $\mathcal{N}_k$ *partially* and solve $\tilde{\mathcal{P}}1$ to obtain $\tilde{b}_v^{kn}$;
if $O_{\tilde{\mathcal{P}}1} > 0$ **then**
  Rank candidates as Q_k^s by descending $\sum_{n\in\mathcal{N}} \tilde{b}_v^{kn}$;
  Revert to the original problem $\mathcal{P}1$;
  if $O_t > 0$ **then** set O_t as $\mathcal{P}1$'s *upper bound*;
  Allocate Q_k^s to $\mathcal{N}_k$ and solve $\mathcal{P}1$;
  if $0 < O_t \& (O_t < O_{\mathcal{P}1} || O_{\mathcal{P}1} < 0) \& \overline{skip}$ **then break**;
  if $0 < O_{\mathcal{P}1} \& (O_{\mathcal{P}1} < O_t || O_t < 0)$ **then** $O_t = O_{\mathcal{P}1}$;
else if $O_t > 0$ & $\overline{skip}$ **then break**;

Evaluation

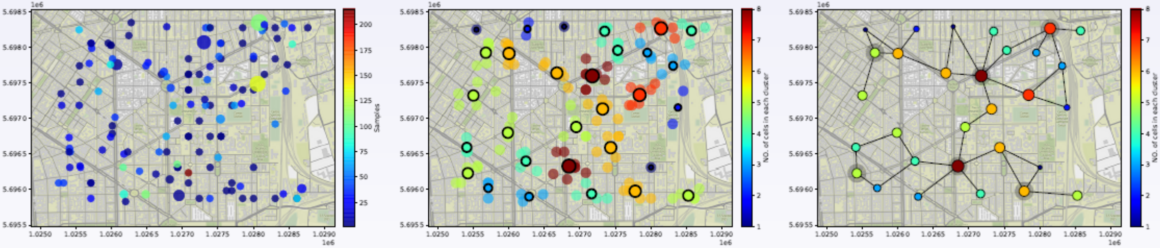
Network Topologies: Random Graphs[6]



(a) 50N50E (b) 80N120E

[6] P. Erdos and A. Renyi, “On Random Graphs I,” *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297, 1959.

Network Topologies: A Real Network Scenario



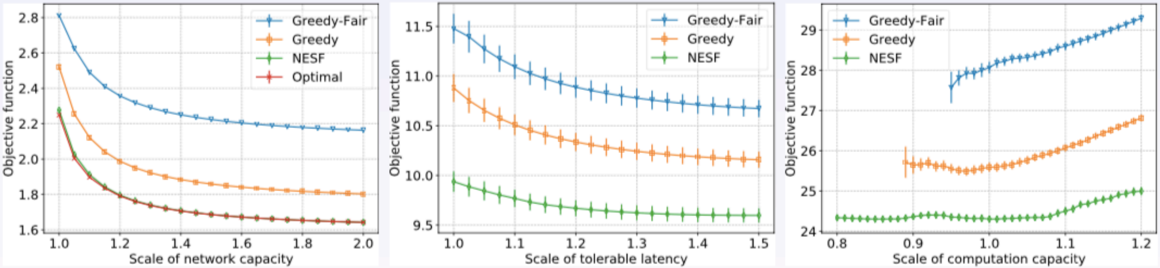
(a) Vodafone LTE cells[7] (b) Cell clusters (c) Topology on clusters

Figure: Città Studi topology with 30 nodes, 35 edges and 6 ingress nodes (marked with gray shadow).

[7] https://www.opencellid.org/

Evaluation

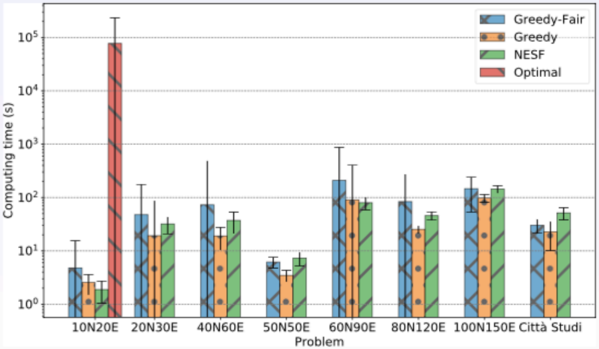
Scaling network capacity, tolerable latency and computation L_3



(a) Network capacity (10N20E) (b) Tolerable latency (80N120E) (c) Computation L_3 (Città Studi)

Evaluation

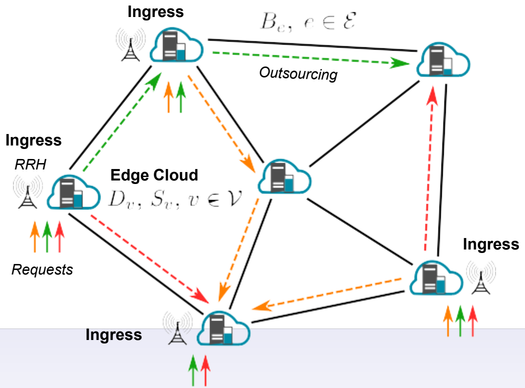
Computing time



Outline

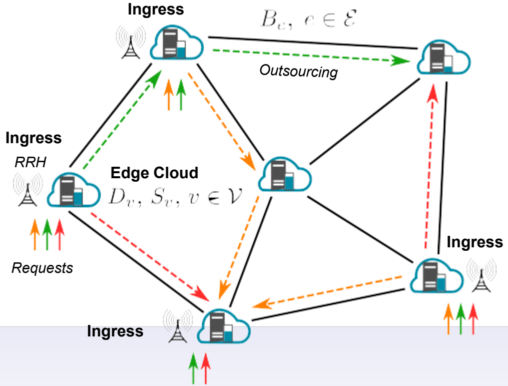
System Model

MEC Network & Request



System Model

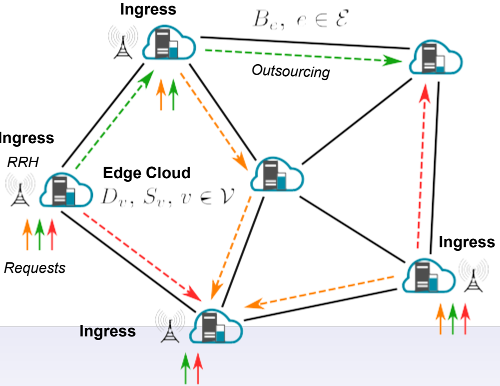
MEC Network & Request



Request

An aggregated communication-computation demand (e.g., web, video, game, etc.,) requiring bandwidth, storage and computation resources of the network.

MEC Network & Request



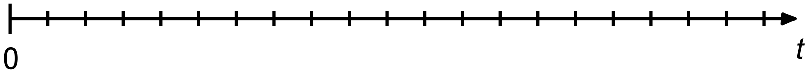
Request

An aggregated communication-computation demand (e.g., web, video, game, etc.,) requiring bandwidth, storage and computation resources of the network.

Parameter	Definition
s^k	Source node of request $k \in K$ Average
λ^k	arrival rate of request k Processing
η^k	density of request k Storage required
m^k	to serve request k Revenue gained
μ^k	from serving request k

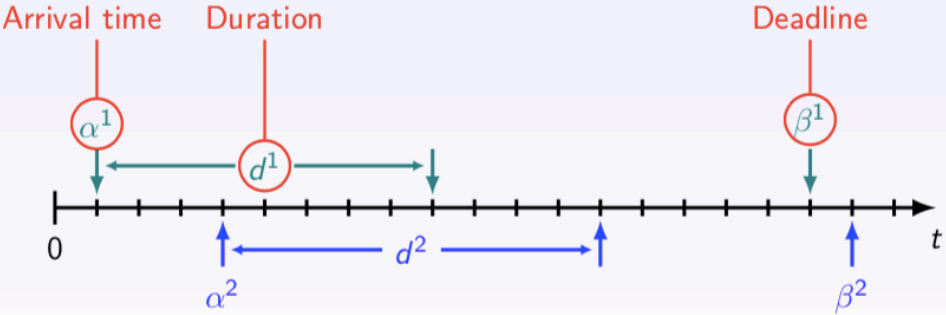
System Model

Request Calendaring Example



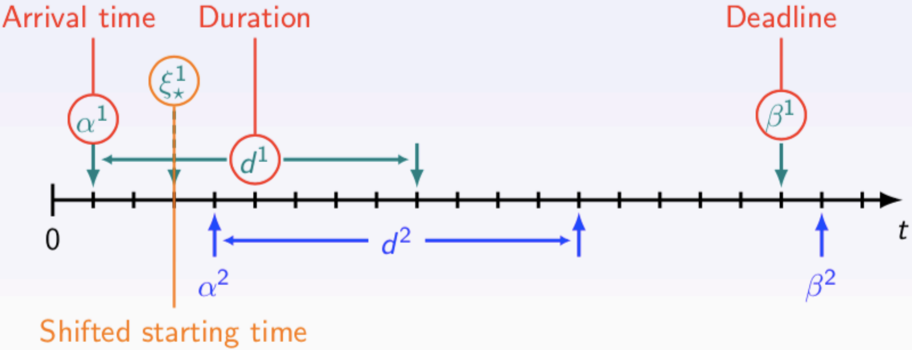
Request Calendaring Example

Request Calendaring Example



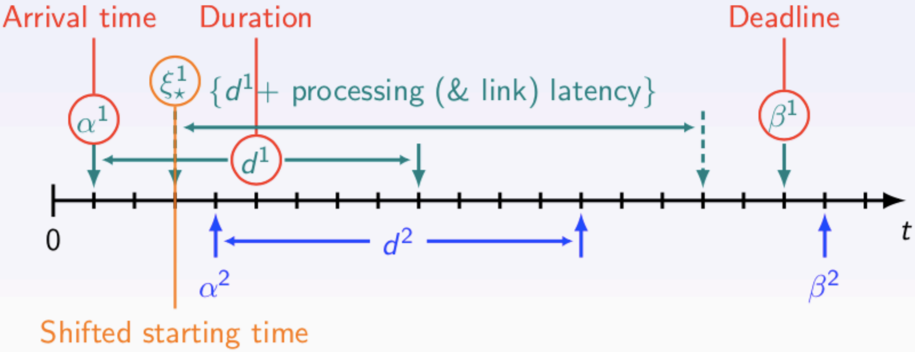
System Model

Request Calendaring Example



System Model

Request Calendaring Example



Problem Formulation

$\mathcal{P}0$:

$$\max_{\substack{z^{kt},\, q^{kv},\, r^{kvt},\\ \rho^{kvt},\, p_e^{kvt}}} \sum_{t\in\mathcal{T}} \sum_{k\in\mathcal{K}} \left\{ \mu^k \boxed{z^{kt}} - \sum_{v\in\mathcal{V}} \left\{ \boxed{r^{kvt}} D_v \theta_v + \boxed{\rho^{kvt}} m^k \phi_v + \sum_{e\in\mathcal{E}} \boxed{p_e^{kvt}} B_e \psi_e \right\} \right\},$$

s.t. Life cycle of requests (α^k, β^k, d^k, $k \in \mathcal{K}$),

Processing latency and storage provisioning (D_v, S_v, $v \in \mathcal{V}$),

Network routing and link latency (B_e, $e \in \mathcal{E}$).

Decision variables:

- z^{kt}: Whether request k starts at time slot $t \in \mathcal{T}$ **(Scheduling)**
- q^{kv}: Fraction of request k processed on node v *(Offloading)*
- r^{kvt}: Fraction of node v's computation capacity sliced to k at t *(Provisioning)*
- ρ^{kvt}: Whether node v processes request k at t **(Provisioning)**
- p_e^{kvt}: Fraction of link e's bandwidth sliced to request q^{kv} at t *(Provisioning & Routing)*

Problem Formulation

- $\mathcal{P}0$ is a mixed-integer nonlinear programming (MINLP) problem, which is $\mathcal{NP}$-hard[5].
  - $\mathcal{P}0$ contains many difficult indicator constraints;
  - Variables in $\mathcal{P}0$ are "intertwined", e.g., routing and
  - offloading;

    Constraints for the life cycle of requests;
- Reformulation + Heuristic + B&B:
  - Transform $\mathcal{P}0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($\mathcal{P}1$).
  - Propose *Sequential Fixing and Scheduling* to accelerate B&B.

[5] R. Kannan and C. L. Monma, "On the computational complexity of integer programming problems," in *Optimization and Operations Research*, Springer, 1978, pp. 161–172.

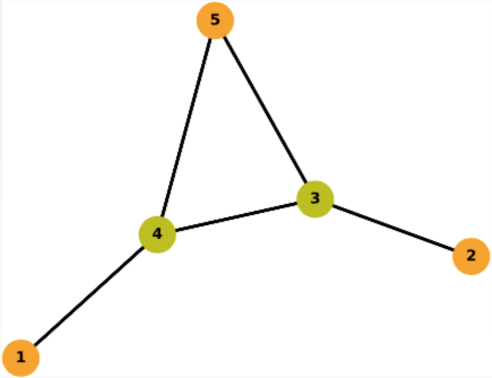
Heuristics

Algorithm 1 *Sequential fixing and scheduling*

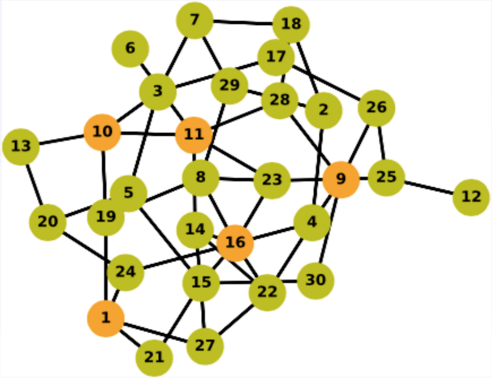
Sort $\mathcal{K}$ in descending order by $\frac{\mu^k}{d^k \lambda^k m^k \eta^k}$, $k \in \mathcal{K}$;
for $k \in \mathcal{K}$ **do**
    Find candidate nodes set $\mathcal{Q}^k$ to compute k, considering **request overlap**;
    if $\mathcal{Q}^k \neq \varnothing$ **then**
        for $\mathcal{V}_i \in \mathcal{Q}^k$ **do**
            Set $b^{kv} = 1$, $\forall v \in \mathcal{V}_i$; **Fix** route (γ_e^{kv}) using *Dijkstra*;
            Optimize $\mathcal{P}1$ to get profit $\mathcal{O}$ and solution $\mathcal{S}$;
            if $\mathcal{O} \geqslant 0$ **then break**;
    if $\mathcal{O} \geqslant \mathcal{O}^\star$ & $\mathcal{Q}^k \neq \varnothing$ **then**
        Update $\mathcal{O}^\star \leftarrow \mathcal{O}$, $\mathcal{S}^\star \leftarrow \mathcal{S}$;
        Admit k and allocate resources based on $\mathcal{S}^\star$;
    else **Reject** k;

Evaluation

Network Topologies: Random Graphs[6]



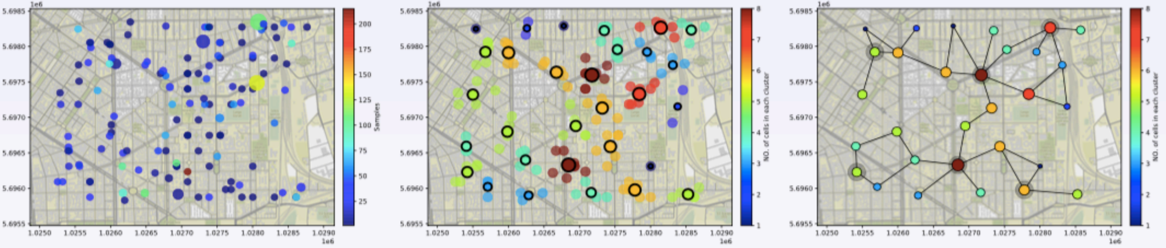
(a) 5N5E3R



(b) 30N50E30R

[6] P. Erdos and A. Renyi, "On Random Graphs I," *Publicationes Mathematicae Debrecen*, vol. 6, pp. 290–297,

Network Topologies: A Real Network Scenario



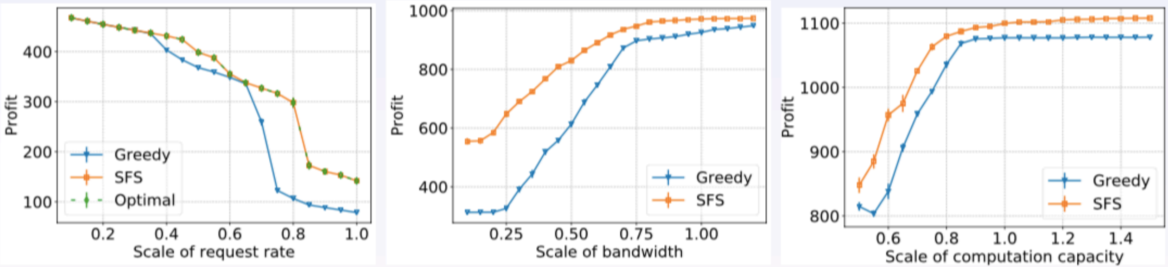
(a) Vodafone LTE cells[7] (b) Cell clusters (c) Topology on clusters

Figure: Città Studi topology with 30 nodes, 35 edges and 6 ingress nodes (marked with gray shadow).

[7] https://www.opencellid.org/

Evaluation

Scaling request rate λ^k, link bandwidth B_e, and computation capacity D_v



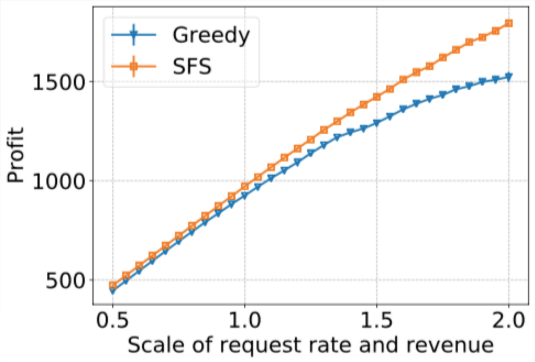
(a) Profit - λ^k, (5N5E3R)

(b) Profit - B_e, (30N50E30R)

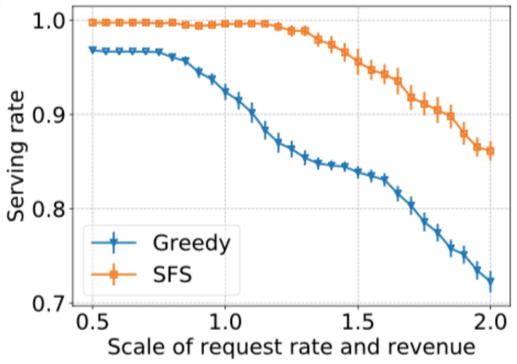
(c) Profit - D_v, (CittàStudi30R)

Evaluation

Scaling simultaneously (request rate λ^k and revenue μ^k)



(a) Profit - (λ^k, μ^k), (30N50E30R)



(b) Serving rate - (λ^k, μ^k), (30N50E30R)

Outline

Conclusion

Conclusion

- We proposed an optimization framework for resource management in MEC networks.
- We investigated three aspects: slicing, planning, and scheduling of MEC network resources to serve aggregated mobile traffic and user requests with different QoS requirements.
- The framework was targeted at reducing the total latency, saving network operation costs and improving profit of both mobile operators and service providers.
- It jointly optimized edge resources in terms of communication, computation and storage under constraints of latency, capacity, budget and request's life cycle, which are NP-hard.
- To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.

Conclusion

- We proposed an optimization framework for resource management in MEC networks.
- We investigated three aspects: slicing, planning, and scheduling of MEC network resources to serve aggregated mobile traffic and user requests with different QoS requirements.
- The framework was targeted at reducing the total latency, saving network operation costs and improving profit of both mobile operators and service providers.
- It jointly optimized edge resources in terms of communication, computation and storage under constraints of latency, capacity, budget and request's life cycle, which are $\mathcal{NP}$-hard.
- To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.

Conclusion

Conclusion

- We proposed an optimization framework for resource management in MEC networks.
- We investigated three aspects: slicing, planning, and scheduling of MEC network resources to serve aggregated mobile traffic and user requests with different QoS requirements.
- The framework was targeted at reducing the total latency, saving network operation costs and improving profit of both mobile operators and service providers.
- It jointly optimized edge resources in terms of communication, computation and storage under constraints of latency, capacity, budget and request's life cycle, which are NP-hard.
- To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.

Conclusion

Conclusion

- We proposed an optimization framework for resource management in MEC networks.
- We investigated three aspects: slicing, planning, and scheduling of MEC network resources to serve aggregated mobile traffic and user requests with different QoS requirements.
- The framework was targeted at reducing the total latency, saving network operation costs and improving profit of both mobile operators and service providers.
- It jointly optimized edge resources in terms of communication, computation and storage under constraints of latency, capacity, budget and request's life cycle, which are NP-hard.
- To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.

Conclusion

Conclusion

- We proposed an optimization framework for resource management in MEC networks.
- We investigated three aspects: slicing, planning, and scheduling of MEC network resources to serve aggregated mobile traffic and user requests with different QoS requirements.
- The framework was targeted at reducing the total latency, saving network operation costs and improving profit of both mobile operators and service providers.
- It jointly optimized edge resources in terms of communication, computation and storage under constraints of latency, capacity, budget and request's life cycle, which are NP-hard.
- To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.