Optimization Framework for Resource Management of Mobile Edge Computing Networks

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

1. Introduction
2. System Architecture
3. Joint Slicing Network and Edge Computing Resources
4. Joint Resource Planning and Slicing for Network and Edge Computing
5. Resource Calendaring for Mobile Edge Computing
6. Conclusion
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.
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.

---

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.

---

Reference:

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.
Motivation

- Mobile Edge Computing (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network. autonomous driving: < 1ms
- 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.

---

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 makes 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.

---

Motivation

- *Mobile Edge Computing* (MEC) provides an IT service environment and cloud-computing capabilities at the edge of mobile network. autonomous driving: $< 1ms$
- 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.
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:**
  - They consider the contexts of single MEC or multiple MECs without interconnection.
  - We consider multiple MECs with arbitrary topology.

- **Network Planning:**
  - They study nodes placement and the resource configuration.
  - We study joint offloading of workloads, slicing and planning of resources.

- **Request/Resource Scheduling:**
  - They focus on the problems without processing or routing aspects.
  - We jointly optimize admission, offloading, scheduling and routing.

---


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

- Joint Resource Planning and Slicing for Network and Edge Computing

- Resource Calendarizing for Mobile Edge Computing
Outline

1. Introduction
2. System Architecture
3. Joint Slicing Network and Edge Computing Resources
4. Joint Resource Planning and Slicing for Network and Edge Computing
5. Resource Calendaring for Mobile Edge Computing
6. Conclusion
System Architecture

MEC Networks

RRH  Edge Cloud
System Architecture

MEC Networks

Edge Node
\( v \in V \)

RRH
\( C_v \)

Edge Cloud
\( D_v S_v \)
System Architecture

MEC Networks

Edge Node
$v \in V$

RRH
$C_v$

Edge Cloud
$D_v$

Resource Management of MEC Networks
System Architecture

MEC Networks

Edge Node $v \in V$

Ingress

$B_e$

RRH $C_v$

Edge Cloud $D_v, S_v$

Incoming traffic or requests

Ingress
System Architecture

MEC Networks

Ingress

Edge Node

$\forall v \in V$

RRH $C_v$

Outsourcing

Incoming traffic or requests

Resource Management of MEC Networks
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.
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.
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.
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

<table>
<thead>
<tr>
<th>Resource:</th>
<th>Edge Slicing</th>
<th>Edge Planning</th>
<th>Edge Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wireless network ((C_V))</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Computation ((D_V))</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage ((S_V))</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Link ((B_e))</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time domain</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ingress node</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Topology (G(V, E))</td>
<td>Single Hierarchical</td>
<td>Multiple Arbitrary</td>
<td>Multiple Arbitrary</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation:</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Slicing</td>
<td>(C_V, D_V)</td>
<td>(C_V, D_V)</td>
<td>(B_e, D_V, S_V)</td>
</tr>
<tr>
<td>Offloading</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Routing Planning</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Admission</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Scheduling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Optimization:</th>
<th>Traffic</th>
<th>Traffic</th>
<th>Reques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Target Constraints Approach</td>
<td>Capacities, latency</td>
<td>Capacities, latency, budget</td>
<td>Capacities, life cycle</td>
</tr>
<tr>
<td>Traffic Latency</td>
<td>Centralized</td>
<td>Centralized</td>
<td>Centralized + Decentralized</td>
</tr>
</tbody>
</table>
Outline

1. Introduction
2. System Architecture
3. Joint Slicing Network and Edge Computing Resources
4. Joint Resource Planning and Slicing for Network and Edge Computing
5. Resource Calendaring for Mobile Edge Computing
6. Conclusion
Problem Formulation

\[ \mathcal{P}0: \]

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

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

Wireless network capacity \( (C_v, v = \text{ingress}) \),

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

- $P_0$ is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard\(^5\).

  * Branch and Bound method can be exploited, but:
    * $P_0$ contains many difficult indicator constraints;
    * Computing time exponentially increases w.r.t. problem size;

  Reformulation + Heuristic + B&B:

  * Transform $P_0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($P_1$).

  Propose *Sequential Fixing* and *Greedy* to accelerate B&B.

---

Problem Formulation

- $P_0$ is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard$^5$.
  
  * Branch and Bound method can be exploited, but:
  * $P_0$ contains many difficult indicator constraints;
  * Computing time exponentially increases w.r.t. problem size;

Reformulation + Heuristic + B&B:

- Transform $P_0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($P_1$).

Propose Sequential Fixing and Greedy to accelerate B&B.

---

Problem Formulation

- $P_0$ is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard\(^5\).
  
  Branch and Bound method can be exploited, but:
  
  - $P_0$ contains many difficult indicator constraints;
    Computing time exponentially increases w.r.t. problem size;
  -...

Reformulation + Heuristic + B&B:

- Transform $P_0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($P_1$).

Propose **Sequential Fixing** and **Greedy** to accelerate B&B.

---

Problem Formulation

- P0 is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard\(^5\).
  
  *Branch and Bound* method can be exploited, but:
  
  * P0 contains many difficult indicator constraints;
  
  Computing time exponentially increases w.r.t. problem size;

Reformulation + Heuristic + B&B:

- Transform P0 into a mixed-integer quadratically constrained programming (MIQCP) problem (P1).

Propose **Sequential Fixing** and **Greedy** to accelerate B&B.

---

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 $\mathcal{P}1$. 

Resource Management of MEC Networks
Network Topologies:
Evaluation

Scaling computation capability $D_v$, tolerable latency $\tau_n$

(a) Scaling $D_v$ ($|N| = 3$, $|V| = 15$)

(b) Scaling $\tau_n$ ($|N| = 3$, $|V| = 15$)
Evaluation

Computing time

![Graph showing computing time for different numbers of traffic types with Optimal, SF, Greedy, and Random strategies.](image-url)
Problem Formulation

\[ \mathcal{P} 0: \]

\[
\min_{c_{kn}, b_{kn}, q_{vn}, r_{vn}, \delta_{vn}, R_{vn}} \sum_{n \in \mathcal{N}} \max_{k \in \mathcal{K}} \left\{ T_{kn}^{Network} + T_{kn}^{Process + Link} \right\} + w \sum_{v \in \mathcal{V}} J_{v}^{Operation},
\]

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

\begin{align*}
\text{Wireless network capacity} & (C_k, k \in \mathcal{K}), \\
\text{Computation capacity and planning budget} & (D_v, v \in \mathcal{V}, P), \\
\text{Link capacity} & (B_e, e \in \mathcal{E}).
\end{align*}

Decision variables:

- \(c_{kn}\): Slice of the network capacity for traffic \(kn\) (slicing)
- \(b_{vn}\): Whether traffic \(kn\) is processed on node \(v\) (offloading)
- \(q_{vn}^{kn}\): Percentage of traffic \(kn\) processed on node \(v\) (offloading)
- \(r_{vn}^{kn}\): Percentage of \(v\)'s computation capacity sliced for processing traffic \(kn\) (provisioning)
- \(\delta_{vn}\): Decision for planning computation capacity on node \(v\) (planning)
- \(R_{vn}^{kn}\): Set of links for routing the traffic piece \(q_{vn}^{kn}\) 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.

---

Heuristics

Neighbor Exploration and Sequential Fixing

1. **Input** network parameters, topology

2. Try to host traffic by ingress nodes only (Algorithm 1)

3. Check whether candidate edge nodes can be found to process outsourced traffic

   - **Yes**
     - Set up allocation plan, solve $P_1$, update solution. *Is the best one achieved so far?* (Algorithm 3)

   - **No**

4. **Output** recorded best solution
Heuristics

Neighbor Exploration and Sequential Fixing

**Algorithm 1** Attempt of serving traffic with ingress nodes

1: $D_k^e = D^m - \sum_{n \in N} \lambda^{kn}, \forall k$;
2: $K^u = \{k \in K | D_k^e \leq 0\}; \triangleright$ Unable to host traffic
3: for $k \in K^u$ do
4: Find neighbor ingress nodes to cover $D_k^e$;
5: if found then Add them to $Q_k$ by ascending hop;
6: Rank $N$ as $N_k$ by descending $(\lambda^{kn}, \tau_n), \forall k$;
7: if $K^u = \emptyset$ or $\bigwedge_{k \in K^u} (|Q_k| > 1)$ then
8: Allocate $Q_k$ to $N_k$ in order and repeatedly, $\forall k$;
9: Solve $P1$ to obtain objective function value $O_{P1}$;
10: if $O_{P1} > 0$ then $O_t = O_{P1}$;
Heuristics

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 $|\bigcup_{k \in \mathcal{K}} Q_k| < \left\lfloor \frac{P}{\min(L_s)} \right\rfloor$ and $\mathcal{K}^s \neq \emptyset$ do
   3. Search candidates $B$ for $\hat{k}$ from multi-hop neighbors considering estimated computation capacity;
   4. Rank $B$, $v' = B(0)$;
   5. Spread $v'$ to help other $\mathcal{K}^s \setminus \{\hat{k}\}$ 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 $O_t$;
Heuristics

Neighbor Exploration and Sequential Fixing

Input network parameters, topology

Try to host traffic by ingress nodes only (Algorithm 1)

Check whether candidate edge nodes can be found to process outsourced traffic

Set up allocation plan, solve $P_1$, update solution. Is the best one achieved so far? (Algorithm 3)

Output recorded best solution

Algorithm 3 Allocating resources and obtaining solution

1: Relax $b_{v}^{kn}, \delta_{v}^{a}, \gamma_{e}^{kn,v}$ to continuous ones ($P_1 \rightarrow \tilde{P}_1$);
2: Allocate $Q_k$ to $N_k$ partially and solve $\tilde{P}_1$ to obtain $\tilde{b}_{v}^{kn}$;
3: if $O_{\tilde{P}_1} > 0$ then
4:   Rank candidates as $Q_s^k$ by descending $\sum_{n \in N} \tilde{b}_{v}^{kn}$;
5:   Revert to the original problem $P_1$;
6:   if $O_t > 0$ then set $O_t$ as $P_1$’s upper bound;
7: Allocating $Q_s^k$ to $N_k$ and solve $P_1$;
8:   if $0 < O_t \& (O_t < O_{P_1} \& \& O_{P_1} < 0)$ then skip then break;
9:   if $0 < O_{P_1} \& (O_{P_1} < O_t \& \& O_t < 0)$ then $O_t = O_{P_1}$;
10: else if $O_t > 0 \& \& \text{skip}$ then break;
Network Topologies: Random Graphs

Evaluation

(a) 50N50E

(b) 80N120E

---

Evaluation

Network Topologies: A Real Network Scenario

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

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

7https://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

![Bar chart showing computing time for different problems and algorithms. The chart compares Greedy-Fair, Greedy, NESF, and Optimal algorithms. The x-axis represents different problem sizes (e.g., 10N20E, 20N30E), and the y-axis shows computing time in seconds (log scale). The chart includes error bars for each data point.]
Outline

1. Introduction
2. System Architecture
3. Joint Slicing Network and Edge Computing Resources
4. Joint Resource Planning and Slicing for Network and Edge Computing
5. Resource Calendaring for Mobile Edge Computing
6. Conclusion and Future Work
System Model

MEC Network & Request

Resource Management of MEC Networks
System Model

MEC Network & Request

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

MEC Network & Request

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s^k$</td>
<td>Source node of request $k \in K$ Average arrival rate of request $k$</td>
</tr>
<tr>
<td>$\lambda^k$</td>
<td>Processing density of request $k$</td>
</tr>
<tr>
<td>$\eta^k$</td>
<td>Storage required to serve request $k$</td>
</tr>
<tr>
<td>$m^k$</td>
<td>Revenue gained from serving request $k$</td>
</tr>
<tr>
<td>$\mu^k$</td>
<td></td>
</tr>
</tbody>
</table>

Request
System Model

Request Calendaring Example
System Model

Request Calendaring Example

Arrival time $\alpha^1$, Duration $d^1$, Deadline $\beta^1$
System Model

Request Calendaring Example

Arrival time $\alpha^1$, Duration $d^1$, Deadline $\beta^1$

Arrival time $\alpha^2$, Duration $d^2$, Deadline $\beta^2$
Request Calendaring Example

- Arrival time: $\alpha^1$, $\xi^1_x$
- Duration: $d^1$
- Deadline: $\beta^1$
- Shifted starting time: $\alpha^2$, $d^2$, $\beta^2$
System Model

Request Calendaring Example

Arrival time $\xi^1$ 

Duration $d^1$ 

Deadline $\beta^1$

Shifted starting time $\alpha^2$

$t$
Problem Formulation

\[ \mathcal{P}_0: \]
\[
\max_{z^{kt}, q^{kv}, r^{kvt}, \rho^{kvt}, p^{kvt}} \sum_{t \in T} \sum_{k \in K} \left\{ \mu^k z^{kt} - \sum_{v \in V} \left\{ r^{kvt} D_v \theta_v + \rho^{kvt} m^k \phi_v + \sum_{e \in E} p^{kvt}_e B_e \psi_e \right\} \right\},
\]

\text{s.t.} \quad \text{Life cycle of requests} \; (\alpha^k, \beta^k, d^k, k \in K),
\text{Processing latency and storage provisioning} \; (D_v, S_v, v \in V),
\text{Network routing and link latency} \; (B_e, e \in E).

**Decision variables:**
- \( z^{kt} \): Whether request \( k \) starts at time slot \( t \in T \) \text{ (Scheduling)}
- \( q^{kv} \): Fraction of request \( k \) processed on node \( v \) \text{ (Offloading)}
- \( r^{kvt} \): Fraction of node \( v \)'s computation capacity sliced to \( k \) at \( t \) \text{ (Provisioning)}
- \( \rho^{kvt} \): Whether node \( v \) processes request \( k \) at \( t \) \text{ (Provisioning)}
- \( p^{kvt}_e \): Fraction of link \( e \)'s bandwidth sliced to request \( q^{kv} \) at \( t \) \text{ (Provisioning & Routing)}
Problem Formulation

- $P_0$ is a mixed-integer nonlinear programming (MINLP) problem, which is NP-hard$^5$.
  - $P_0$ contains many difficult indicator constraints;
  - Variables in $P_0$ are “intertwined”, e.g., routing and offloading;
    Constraints for the life cycle of requests;
- Reformulation + Heuristic + B&B:
  - Transform $P_0$ into a mixed-integer quadratically constrained programming (MIQCP) problem ($P_1$).
  - Propose *Sequential Fixing and Scheduling* to accelerate B&B.

---

Heuristics

Algorithm 1 Sequential fixing and scheduling

1: Sort $\mathcal{K}$ in descending order by $\frac{\mu_k}{d_k}^{\chi_k}$, $k \in \mathcal{K}$;
2: for $k \in \mathcal{K}$ do
3:   Find candidate nodes set $Q^k$ to compute $k$, considering request overlap;
4:   if $Q^k \neq \emptyset$ then
5:     for $\mathcal{V}_i \in Q^k$ do
6:       Set $b^{kv} = 1$, $\forall v \in \mathcal{V}_i$; Fix route ($\gamma^{kv}_e$) using Dijkstra;
7:       Optimize $P1$ to get profit $\mathcal{O}$ and solution $S$;
8:       if $\mathcal{O} \geq 0$ then break;
9:     if $\mathcal{O} \geq \mathcal{O}^* \& Q^k \neq \emptyset$ then
10:        Update $\mathcal{O}^* \leftarrow \mathcal{O}$, $S^* \leftarrow S$;
11:        Admit $k$ and allocate resources based on $S^*$;
12:     else Reject $k$;
Evaluation

Network Topologies: Random Graphs

(a) 5N5E3R

(b) 30N50E30R

---

Evaluation

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 $\lambda^k$, link bandwidth $B_e$, and computation capacity $D_v$

(a) Profit - $\lambda^k$, (5N5E3R)

(b) Profit - $B_e$, (30N50E30R)

(c) Profit - $D_v$, (CittàStudì30R)
Evaluation

Scaling simultaneously (request rate $\lambda^k$ and revenue $\mu^k$)

(a) Profit - $(\lambda^k, \mu^k)$, (30N50E30R)

(b) Serving rate - $(\lambda^k, \mu^k)$, (30N50E30R)
Outline

1. Introduction
2. System Architecture
3. Joint Slicing Network and Edge Computing Resources
4. Joint Resource Planning and Slicing for Network and Edge Computing
5. Resource Calendaring for Mobile Edge Computing
6. 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 N P -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 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 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 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 NP-hard.

To tackle them efficiently, centralized and decentralized approaches were designed to provide approximate resource allocation solutions in short computing time.