<table>
<thead>
<tr>
<th>Rank</th>
<th>Author</th>
<th>Affiliation</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>Shuaibing Lu</td>
<td>Faculty of Information Technology</td>
<td>Beijing, China</td>
</tr>
<tr>
<td>2nd</td>
<td>Ran Yan</td>
<td>Faculty of Information Technology</td>
<td>Beijing, China</td>
</tr>
<tr>
<td>3rd</td>
<td>Jie Wu</td>
<td>Center for Networked Computing</td>
<td>Philadelphia, USA</td>
</tr>
</tbody>
</table>

Efficient Microservice Deployment with Dependencies in Multi-Access Edge Computing
CONTENTS

1. Introduction
2. Model and Formulation
3. Algorithm Design
4. Experiment and Results
Edge computing is emerging

- The recent advances in edge computing technologies have enabled a wide range of data-intensive and time-critical applications.
- Edge systems → multiple services, a number of users...

Smart City
Automated Driving
AR/VR
Microservice architecture

- The limitations of traditional monolithic applications in terms of scalability and flexibility.
- A lightweight and highly flexible architectural pattern.
Problem

- How to deploy the microservices with dependencies especially under resource constrained edge computing systems still remains a problem?
An example scenario

<table>
<thead>
<tr>
<th>Server</th>
<th>Computing capability</th>
<th>Storage capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>v2</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>v3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>v4</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>v5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>link</th>
<th>$l(v_1,v_2)$</th>
<th>$l(v_2,v_3)$</th>
<th>$l(v_3,v_4)$</th>
<th>$l(v_4,v_5)$</th>
<th>$l(v_5,v_1)$</th>
<th>$l(v_2,v_4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandwidth</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
Challenges

1. How can complex dependencies among microservices be effectively dealt with to improve the overall efficiency?

2. How to balance the trade-off between processing and transmission time for optimal deployment without overwhelming the resource constraints?
CONTENTS

1. Introduction
2. Model and Formulation
3. Algorithm Design
4. Experiment and Results
Model and Formulation

Model

• system model: $S = \{S_h\}$, $S_h = \{M_h, E_h\}$, $M_h = \{m_i^h\}$, $E_h = \{e_{m_i \rightarrow m_j}^h\}$,

$$V = \{v_k\}, \quad L = \{l(v_k, v_q)\}$$

• computation model:

$$d_c(m_i^h) = x(i, k) \cdot q_{m_i^h} / c(v_k)$$
Model and Formulation

Model

- communication model:

\[ d_l \left( e^{h}_{m_i \rightarrow m_j} \right) = y(i, k) \cdot r^{h}_{m_i \rightarrow m_j} / b(v_k, v_q) \]

- Data flow size between microservices
- Communication capability of edge server
Model and Formulation

- **Model**
  - makespan model:

\[
    f(m_j^h) = \max_{\forall i, j} \left\{ f(m_i^h) + d_c(m_j^h) + d_l(e_{m_i \rightarrow m_j}^h) \right\}
\]

- Completion time of the predecessor \( m_i^h \)  
- Computation time of \( m_j^h \)  
- Transmission time for data transferred

\[
    T_h = \max_{\forall j, S_h \in S} \{ f(m_j^h) \} \quad \text{the makespan of service } S_h
\]

\[
    T = \max_{\forall S_h \in S} \{ T_h \} \quad \text{makespan of services in set } S
\]
Model and Formulation

Formulation

P1: minimize \( \forall i, j \) \( T \) \hspace{1cm} (1)

subject to

\[ \sum_{k=1}^{n} x(i, k) = 1, \] \hspace{1cm} (2)

\[ \sum_{i=1}^{n} x(i, k) \leq \phi(v_k), \] \hspace{1cm} (3)

\[ b(v_k, v_q) \leq \tau, \] \hspace{1cm} (4)

\[ x(i, k) \in \{0,1\}, y(i, j) \in \{0,1\}, \forall i, \forall j, \forall k \] \hspace{1cm} (5)

Optimal Microservice Deployment with Dependencies (OMDD problem): OMDD problem is how to find a strategy for microservice in S to minimize P1 under the constraints (2)-(5).
CONTENTS

1. Introduction
2. Background
3. Algorithm Design
4. Experiment and Results
enhanced graph construction

- prioritization among multiple services
Scenario 1: OMDD with no storage constraint

How to balance the computing and communication resources?

- **Step 1**: find the server $v_0$ with the highest computing capability
  - Sort $V$ in descending order according to the server’s computing capability $c(v_k)$.

- **Step 2**: deploy the microservices
  - deploy all microservices on the server $v_0$

**Theorem**: OMDD-US is an optimal solution for solving P1 under the constraints (7), (9)-(11).
Scenario 2: OMDD with no communication constraint

how to balance the computing and storage resources?

- **Step 1**
  - Sort $V$ in descending order according to the server’s computing capability $c(v_k)$.

- **Step 2**
  - Sort microservices in descending order of required computing capability $q_{m_i}$.

- **Step 3**
  - Deploy the microservices sorted in descending order to servers with stronger computing capabilities.

**Theorem**: OMDD-UB is an optimal solution on solving P1 under the constraints (7)-(8), (10).
Scenario 3: OMDD with constraints (7)-(10)

- **Definition 1 (main path):** The main path $p_i$ refers to the path with the maximum weight $\text{arg max}\{w(p_i)\}$ of $\hat{I}$.

- **Definition 2 (preferred server):** Let $v^\circ$ indicate the preferred server of $V$, where $v^\circ = \max_{v_k \in V} \xi(v_k)\{v_k | v_k \in V\}$. Here, $\xi(v_k)$ is the priority value of $v_k$ with the sum of the computing capacity and the maximum bandwidth that is connected in $g$.

- **Definition 3 (maximum cut):** Let $C_{p_i}$ indicate the maximum cut of path $p_i$ in $\hat{I}$ which constructs by $|\phi_{v^\circ}|$ microservices with the largest weights combination.

**Theorem:** The main path will become the critical path when server computation capacities and inter-server bandwidths are equal, where $\{c(v_i) = b(v_k,v_q) | \forall v_i \in V, \forall (v_k,v_q) \in L\}$. 
Scenario 3: OMDD with constraints (7)-(10)

- **Step 1: a preliminary deployment method based on main path embedding**
  - find the main path with the maximum weight.
  - Prioritize placement on preferred server $v^\circ$.
  - Compare the number of microservices on the main path with the capacity of the servers; if the server capacity is less than the number of microservices on the main path, perform partitioning.

- **Step 2: improved simulated annealing algorithm**
  - Through multiple iterations, the simulated annealing algorithm gradually converges to optimal solutions, thereby improving the quality and effectiveness of the deployment strategy.
Experiment and Results

- Basic Setting
  - Hardware: Windows 10 with an Intel(R) Xeon(R) Silver 4210R CPU @ 2.40GHz, NVIDIA RTX5000 GPU, and 32GB memory.
  - Dataset: China Telecom Shanghai Company (3,233 base station locations and their corresponding user connections in June 2014.)
  - Range: Randomly selected subsets of locations containing 6, 20, and 50 base stations. Each service was abstracted as a DAG and generated with the number of 18, 50, and 120 microservices, respectively.
Experiment and Results

- **Two Comparison algorithms**
  - **Simulated Annealing-only (SA):** Traditional annealing algorithm, generating random initial values.
  - **Q-Learning (QL):** States are composed of the allocation status of a series of services. Each service can be assigned to different servers (edge nodes or cloud) or remain unassigned, and the action space contains $\sum_{h=1}^{[S]} |M_H| \times |V|$ actions. We select the deployment on the corresponding server according to the Q table.
Experiment and Results

Fig. 2. 10 servers and 25 microservices.

Fig. 3. 20 servers and 50 microservices.

Fig. 4. 50 servers and 120 microservices.
Conclusion

- We investigate the microservice deployment problem with dependencies by formulating to minimize the makespan of multiple services under the resources constraints in multiaccess edge computing, and we theoretically analyze and demonstrate the complexity of this problem through the proof of NP-hardness.

- We propose three microservice deployment strategies by offering flexibility and adaptability for various application scenarios. We initially consider two straightforward scenarios: one with unlimited storage resources under the bandwidth constraint, and the other with unlimited bandwidth resources under the storage constraint. For each of these two scenarios, we introduce a novel enhanced graph construction method and design two optimal solutions.
Conclusion

- We then consider a more complicated scenario with resource limitations on storage, computing, and communication. We produce a feasible solution by introducing an effective embedding method based on the novel definitions of the main path and preferred server, which are extracted based on the topology features of the services and edge environment, respectively. Based on that, we propose an updating method by introducing an improved simulated annealing strategy, and we analyze the complexity.

- We conducted extensive experiments to compare our strategies with several baselines based on the China Telecom Shanghai Company dataset, which was constructed by the geographic information of 3,233 base stations. The results are shown from different perspectives to provide conclusions.
THANK YOU