2023 International Wireless Communications and Mobile Computing (IWCMC) | 979-8-3503-3339-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/IWCMC58020.2023.10182768

# Dependency-aware Microservice Deployment and Resource Allocation in Distributed Edge Networks

Jizhe Zhou, Guangchao Wang, Wei Zhou
China Academy of Information and Communications Technology, Beijing, China
{zhoujizhe, wangguangchao, zhouwei1}@caict.ac.cn

***Abstract*—Microservice deployment strategy gains increasing attentions due to its benefits for reducing the overall computation and transmission delay of applications. Most prior work models the dependencies between microservices of an application as service function chain or directed acyclic graph, and neglects the interaction relationship and the different communication overhead between two interactive microservices. To characterize the features of applications, our work considers the interaction dependency between microservices, and models the communication frequency over each dependency edge. In this paper, we address the joint optimization of microservice deployment and resource allocation decisions to minimize the sum of delay of multiple applications in the distributed edge networks. Since this problem is NP-hard, a coalition game-based algorithm is proposed to obtain the dependency-aware microservice deployment and resource allocation scheme. Numerical results validate the advantages of our approach particularly when microservices have strong interaction.**

***Index Terms*—Microservices deployment, computing network, dependency awareness, multi-access edge computing**

## I. Introduction

Recent years have seen the surge of computation-intensive and delay-sensitive applications, such as augmented reality (AR), interactive gaming, Internet of Vehicles and so on. To meet the computation resource and delay requirement of these applications, Multi-access Edge Computing (MEC) emerges to deploy computation resource in the vicinity of user equipments, e.g., deployed at the base stations or access points in the radio access networks. By offloading computing tasks to MEC, user equipments can reduce the overall delay of computing and transmission compared to local computing or offloading to remote cloud computing platform [1]. In addition, to accelerate development and operation, microservice architecture (MSA) has been widely implemented to cloud platforms. By separating an monolithic application to several loosely-decoupled microservices, MSA allows microservices to be distributedly and independently deployed in the cloud platform while running in coordination with each other [2]. In this sense, combining MSA and distributed edge networks gains increasing attentions to further reduce the delay of applications by leverage the resource-constraint edge nodes. Fig. 1 gives an example of an AR application [3], and its components (e.g., tracker, mapper, etc.) can be deployed as microservices at different edge nodes.

This work was supported in part by National Key R&D Program of China under grant No. 2021YFB2900200 and China Postdoctoral Science Foundation under grant No. 2022M713475.

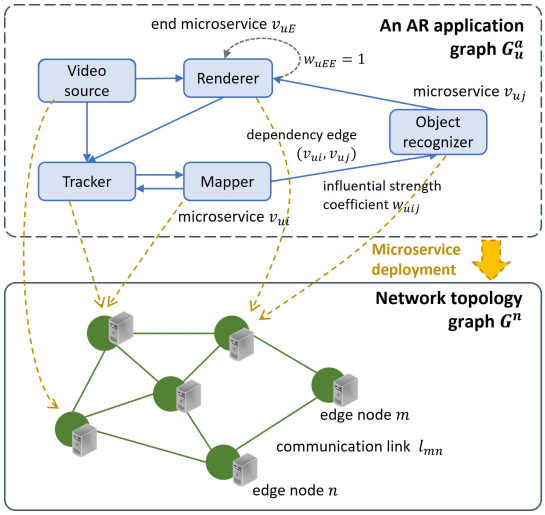


Fig. 1. An example of the dependency-aware microservice deployment for an AR application in the distributed edge networks.

The key issue of microservice-based edge network is to derive the optimal deployment and resource allocation scheme of microservices to geographically edge nodes with the concerns of dependencies among microservices [4]–[6]. Most prior work models the application using service function chain [7] or directed acyclic graph [8], [9], so that the dependence between a pair of microservices can only be directed from one to the other. However, since microservices may communicate with each other, thus there exists interaction relationship between two microservices (e.g., the interaction between 'tracker' and 'mapper' microservices in Fig. 1), or indirected interaction relationship of a group of microservices if they communicates one to the other in a closed-loop (e.g., microservices of tracker, mapper, object recognizer and renderer have indirected interaction relationship in Fig. 1). Even in [10] that the interaction between two microservices is considered, it assumes that the communication traffic in both directions is the same and contributes equally to the communication overhead. However, in fact, the communication frequency and data transmission are always not coordinated between two microservices, thus, the communication delay in two directions that contributes to the overall delay is not the same. Thus, it is necessary to model the influential strength over each dependency edge to measure the influence from the leading microservice to the following

microservice on the application performance.

This paper proposes the microservice deployment and resource allocation scheme to minimize the overall delay of multiple applications in the distributed edge networks. The main contributions are summarized as follows.

- Different from the prior work, we consider the interaction dependency among microservices and model each application as a weighted directed graph. In the application graph, there exists a dependency edge from the leading microservice to the dependent microservice, which is associated with an influential strength coefficient to estimate how much the former affects the delay performance of the latter.
- To minimize the overall delay of running applications, we formulate the optimization problem of microservice deployment and computation resource allocation in the distributed edge nodes. Since the problem is nonconvex and NP-hard, a coalition game-based algorithm is derived to obtain a suboptimal and stable solution.
- Numerical results are provided to show that the proposed scheme outperforms the greedy algorithm and the influence-unaware microservice deployment scheme particularly when microservices have strong interactions.

The rest of this paper is organized as follows. In Section II, the network model and application model are introduced. The optimization problem is formulated in Section III and then addressed in Section IV. Numerical results are presented in Section V. Finally, conclusions are offered in Section VI.

## II. System Model

We consider a cellular network $\mathcal{G}^n = \{\mathcal{K}, \mathcal{L}\}$ consisting of a set $\mathcal{K} = \{k\}_{k=1}^{K}$ of edge nodes and a set $\mathcal{L}$ of undirect communication links connecting two edges nodes. The edge nodes can be access points co-located with the MEC server. For every link $l_{mn} \in \mathcal{L}$, $m, n \in \mathcal{K}$, the average delivery delay is denoted as $d_{mn}$. Specially, we have $d_{mm} = 0$ for any $m \in \mathcal{K}$. Moreover, the computation capability of each edge node $k \in \mathcal{K}$ is denoted as $C_k$.

### A. Application Model

The set of application in $\mathcal{G}^n$ is denoted as $\mathcal{U} = \{u\}_{u=1}^{U}$. It is assumed that each application can be divided into a set of microservices, each of which can be independently executed in containers and have dependency on parts of the other. Different from the assumptions in the prior work that microservices are processed in sequence and the interactive communication between microservices is not investigated, we also consider various types of dependency relationships including the direct/indirect interaction relationship between microservices.

To thoroughly illustrate the characteristics of microservice-based application, the application $u \in \mathcal{U}$ is denoted as a weighted directed graph $\mathcal{G}_u^a = \{\mathcal{V}_u, \mathcal{E}_u\}$, in which $\mathcal{V}_u = \{v_{ui}\}_{i=1}^{V_u}$ is the set of microservices and $\mathcal{E}_u$ is the set of dependency edges between some pairs of microservices. If there exist an edge $(v_{ui}, v_{uj}) \in \mathcal{E}_u$ for microservices $v_{ui}, v_{uj} \in \mathcal{V}_u$, it means that microservice $v_{uj}$ depends on the output of microservice $v_{ui}$. If we have both $(v_{ui}, v_{uj}) \in \mathcal{E}_u$ and $(v_{uj}, v_{ui}) \in \mathcal{E}_u$ for $v_{ui}, v_{uj} \in \mathcal{V}_u$, it means that microservices $v_{ui}$ and $v_{uj}$ have direct interaction with each other. Further to enable indirect interaction between microservices, a cycle that connects multiple microservices is used, i.e., if edges $(v_{ui}, v_{uj}), (v_{uj}, v_{uh}), (v_{uh}, v_{ui}) \in \mathcal{E}_u$, then, microservices $v_{ui}, v_{uj}, v_{uh} \in \mathcal{V}_u$ form a cycle of $\mathcal{G}_u^a$ and have interaction relationship with each other. Each microservice $v_{ui}$ of application $u$ is associated with the computation requirement $m_{ui}$ of its computation task and output size $o_{uij}$ to microservice $v_{uj}$ if we have $(v_{ui}, v_{uj}) \in \mathcal{E}_u$. The microservice which turns out the final output of one complete execution of a specific application is referred to as the end microservice. The end microservice of application $u$ is denoted as $v_{uE}$. In order to precisely denote the finish of computation by the end microservice $v_{uE}$, we assume that the end microservice $v_{uE}$ has a cycle $(v_{uE}, v_{uE}) \in \mathcal{E}$ to itself, which is referred to as the end edge of application $u$.

Moreover, each dependency edge $(v_{ui}, v_{uj}) \in \mathcal{E}_u$ is associated with an influential strength coefficient $w_{uij} \in \mathbb{R}^+$ to show how the performance of microservice $v_{ui}$ affects its following microservice $v_{uj}$. The influential strength coefficient can be estimated by the communication frequency, the traffic over the link and other statistic parameters associated with the dependency edge. In this paper, $w_{uij}$ for $(v_{ui}, v_{uj}) \in \mathcal{E}_u$ is defined as the communication frequency, that is, the average times of microservice $v_{ui}$ sending its output to microservice $v_{uj}$ with respect to a complete execution. The larger $w_{uij}$, the more frequent of microservice $v_{ui}$ to communicate with and send its output to microservice $v_{uj}$. Apparently, since the end microservice $v_{uE}$ can achieve a complete execution after finishing its computation task, thus, the influential strength $w_{uEE}$ of the end edge $(v_{uE}, v_{uE})$ is 1.

### B. Microservice Deployment

To enable microservice deployment, we define a microservice deploying variable $x_{uik} \in \{0, 1\}$ for microservice $v_{ui} \in \mathcal{V}_u$ and edge node $k \in \mathcal{K}$ to denote if microservice $v_{ui}$ is placed at edge node $k$ ($x_{uik} = 1$) or not ($x_{uik} = 0$). For simplicity, it is assumed that a microservice is placed only at one edge node, which can be expressed as the constraint

$$\sum_{k \in \mathcal{K}} x_{uik} = 1, \text{ for all } v_{ui} \in \mathcal{V}_u, u \in \mathcal{U}. \tag{1}$$

We also define a computation resource allocation variable $y_{uik} \in [0, 1]$ to denote the fraction of computation resources of edge node $k$ allocated to microservice $v_{ui} \in \mathcal{V}_u$. Then, this imposes the resource capacity constraint

$$\sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{V}_u} x_{uik} y_{uik} \leq 1, \text{ for all } k \in \mathcal{K}. \tag{2}$$

In addition, an edge node $k \in \mathcal{K}$ only allocates its computation resources to a microservice $v_{ui}$ if microservice $v_{ui}$ is deployed at it, defined as

$$(1 - x_{uik})y_{uik} \leq 0. \tag{3}$$

For output transmission, if there exist a dependency edge between microservice $v_{ui}$ and $v_{uj}$ and they are both deployed, the former sends its output via the shortest path between two deployed edge nodes.

## III. Problem Formulation

In this section, the delay of microservice-based application is introduced. Then, the optimization problem is formulated in terms of the microservice deployment and resource allocation decisions.

### A. Delay of Microservice

The delay of the microservice-based application consists of two parts, i.e., the computation delay of microservices and the transmission delay of their outputs. Since the communication frequency from one microservice to another is different and dependent on the specific dependency edge, the delay of a microservice is defined as the accumulative weighted computing delay and transmission delay over all its dependency edges to distinct following microservices. In this sense, the overall delay of microservices describes the upper bound of the execution delay of per request for an application. To elaborate the dependency relationship of microservices, a dependency matrix $A_u = [a_{uij}] \in \{0,1\}^{V_u \times V_u}$ is denoted, where $a_{uij} = 1$ if edge $(v_{ui}, v_{uj}) \in \mathcal{E}_u$. Note that the dependency matrix $A_u$ is an asymmetric matrix. Then, the delay of microservice $v_{ui}$ is calculated by the weighted accumulation of computation delay $t_{ui}^{com}$ and transmission delay $t_{uij}^{tr}$ of its output to microservice $v_{uj}$ with $a_{uij} = 1$ as

$$t_{ui} = \sum_{j \in \mathcal{V}_u} a_{uij} w_{uij} \left( t_{ui}^{com} + t_{uij}^{tr} \right), \text{ for all } v_{ui} \in \mathcal{V}_u, \tag{4}$$

where we have

$$t_{ui}^{com} = \sum_{k \in \mathcal{K}} x_{uik} \frac{m_{ui}}{y_{uik} C_k}, \tag{5}$$

and

$$t_{uij}^{tr} = \sum_{k \in \mathcal{K}} \sum_{h \in \mathcal{K}} x_{uik} x_{ujh} o_{uij} \sum_{l_{mn} \in L_{hk}^s} d_{mn}. \tag{6}$$

Here, $L_{kh}^s$ is the set of communication links over the shortest path from edge node $k$ to edge node $h$. Specifically, for the end microservice $v_{uE}$, since we do not consider the transmission of the final output and $d_{mm} = 0$ for any edge node $m \in \mathcal{K}$, then, we have $t_{uEE}^{tr} = 0$ no matter where the end microservice $v_{uE}$ is deployed. Thus, we only consider the computation delay of the end microservice $v_{uE}$.

In this sense, the delay of an application is defined as the sum of the weighted accumulative delay of all microservices as

$$t_u = \sum_{i \in \mathcal{V}_u} t_{ui}, \text{ for all } u \in \mathcal{U}. \tag{7}$$

### B. Delay Minimization

The objective of our problem is to minimize the total delay of applications with respect to microservice deploying decisions $\boldsymbol{x} = \{\boldsymbol{x}_u\}_{u \in \mathcal{U}}$ where $\boldsymbol{x}_u = [x_{uik}] \in \{0,1\}^{V_u \times K}$, and computation resource allocation decisions $\boldsymbol{y} = \{\boldsymbol{y}_u\}_{u \in \mathcal{U}}$ where $\boldsymbol{y}_u = [y_{uik}] \in [0,1]^{V_u \times K}$. According to the delay of application in (7) and constraints in (1)-(3), the optimization problem is defined as

$$\begin{aligned} \min_{\boldsymbol{x}, \boldsymbol{y}} \ & T = \sum_{u \in \mathcal{U}} t_u \\ \text{s.t. } & (1), (2), (3). \end{aligned} \tag{8}$$

## IV. Coalition-based Algorithm of Microservice Deployment and Resource Allocation

The optimization problem (8) is nonconvex and NP-hard to address. In order to obtain the optimal strategy of microservices deployment and resource allocation, a coalition game model is introduced to strengthen the collaboration among microservices for delay reduction.

### A. Coalition Game Model

In this paper, we treat the set of edge nodes as the set of coalitions, which is denoted as $G = \{G_1, ...G_K\}$. Each coalition $G_k$ is composed of the set of microservices deployed on edge node $k \in \mathcal{K}$. On account of the constraint of microservice deployment, the coalitions satisfy the conditions $G_k \cap G_h = \emptyset$, for all $G_k, G_h \in G$, and $\cup_{k=1}^{K} G_k = \{\mathcal{V}_u\}_{u=1}^{U}$. If more microservices are placed on one edge node $k$, the computing resources allocated to each microservice decreases accordingly, resulting in the increase of its computation delay. On contrary, if we distributedly place each microservice on different edge nodes, this neglects the different requirements of computation resources and communication frequency of microservices, which may degrade the overall delay performance. Thus, there exits a optimal coalition structure $G^*$ associated with the cooperation of microservices, which is equivalent to the optimal microservice deployment and resource allocation scheme. In the coalition game, the set of players, represented by the set $\mathcal{V} = \{\mathcal{V}_u\}_{u=1}^{U}$ of microservices of all applications, iteratively explore and update coalition structure $G$ to improve the utility. Given a coalition structure $G$, the transmission delay between microservices are determined. Thus, the computation resource allocation is optimized for all microservices assigned with each coalition $G_k \in G$, respectively. Here, the utility of a coalition $G_k$ is defined as the minus of the accumulative delay of all microservices deployed on edge node $k$ as

$$U(G_k) = - \sum_{v_{ui} \in G_k} t_{ui} \tag{9}$$

where $G_k = \{v_{ui} | x_{uik} = 1, \forall i \in \mathcal{V}_u, u \in \mathcal{U}\}$. In addition, the utility of the coalition structure $G$ is defined as the sum of the utility of all coalitions, which is equivalent to the minus of the overall delay $T$ of all microservices as

$$U(G) = \sum_{k \in \mathcal{K}} U(G_k) = -T \tag{10}$$

Let $\mathbb{G}$ denote the set of feasible solutions of coalition structure. Accordingly, the problem to find the coalition structure $G \in \mathbb{G}$ with the maximal utility $U(G)$ is equal to the optimization problem (8). Then, we define a coalition game for microservices deployment and resource allocation according to [11].

*Definition 1 (Coalition Game for Microservices Deployment and Resource Allocation):* The coalition game for microservices deployment and resource allocation is defined as $(\mathcal{V}, G, U)$, where $\mathcal{V}$ is the set of players, $G \in \mathbb{G}$ is coalition structure and $U$ is the utility of a certain coalition partition.

### B. Coalition Game-based Algorithm

To transform the coalition structure, it is necessary to have a well defined preference order for players and make them join or leave a coalition based on its own preference. Let $\succ$ define the preference relation of two collections of coalitions $G$ and $G'$ for microservices $\mathcal{V}$, so that $G' \succ G$ implies that microservices of $\mathcal{V}$ prefer the partition $G'$ than partition $G$. In essence, microservice $v_{ui}$ decides to join or leave a coalition based on the utility the whole partition. Assume that currently coalition struction is $G$ and microservice $v_{ui}$ is a member of coalition $G_h$, then, the switch rule of microservice $v_{ui}$ to leave $G_h$ and join another $G_k \neq G_h$ is defined as

$$G' \succ G \Leftrightarrow U(G') > U(G), \tag{11}$$

where $G' = G/\{G_k, G_h\} \cup \{G_{k'}, G_{h'}\}$, $G_{k'} = \{G_k \cup \{v_{ui}\}\}$, $G_{h'} = \{G_h/\{v_{ui}\}\}$. The preference (11) indicates that microservice $v_{ui}$ switches to another coalition if the overall utility of coalition structure increases. According to the conditions above, the change of coalition structure is enabled by the join-and-leave actions of microservices.

Assume that the current coalition structure is $G = \{G_1, ..., G_k\}$. At iterate $n$, if preference (11) is satisfied, a microservice leaves its current coalition and join a more preferable new coalition. If the preference (11) is not satisfied, there is still a chance of a microservice to leave its current coalition and join a new coalition. The probability of this chance at iterate $n$ is referred to as the acceptance probability $p(T_n) = \exp((U(G') - U(G))/T_n)$, where $T_n = T_0/\log(n)$, and $T_0 > 0$ is the initial temperature in stimulated annealing. Given a coalition structure $G$, the transmission delay between microservices is determined due to the fixed deploying variables $\boldsymbol{x}$. Then, the computation resource allocation can be solved independently at each coalition $G_k$ to minimize the accumulative computing delay of all microservices $v_{ui} \in G_k$. With fixed $\boldsymbol{x}$, the optimization problem of computation resource allocation is convex to variables $\boldsymbol{y}$. Thus, it can be solved by using KKT conditions for the resource allocation problem in each coalition. The overall algorithm is summarized in Algorithm 1. Algorithm 1 stops if converges to a Nash-stable solution such that any microservices cannot switch coalitions to improve the overall utility.

**Algorithm 1** Dependency-aware Microservice Deployment and Resource Allocation

**Require:** Initial coalition structure $G$, let $n = 1$.

**repeat**
  Uniformly randomly choose a microservice $v_{ui}$, and its current assigned coalition is $G_h$. Randomly choose another coalition $G_k \neq G_h$, $G_k \in G$. Let $G_{k'} = \{G_k \cup \{v_{ui}\}\}$, $G_{h'} = \{G_h/\{v_{ui}\}\}$. The new coalition structure is denoted as $G' = G/\{G_k, G_h\} \cup \{G_{k'}, G_{h'}\}$.
  **if** $G' \succ G$ **then**
    Microservice $v_{ui}$ leave $G_h$ and join $G_k$, making new coalitions $G'_k$ and $G'_h$. Computation resource allocation is optimized in new coalition $G'_k$ and $G'_h$ using KKT conditions. The coalition structure of microservices $\mathcal{V}$ is transferred to $G'$.
    Let $G = G'$.
  **else**
    **if** rand $< p(T_n)$ **then**
      Go to Step 4 and Step 5.
    **end if**
  **end if**
  Let $n = n + 1$.
**until** Convenges to Nash-stable solution

## V. Numerical Results

In this section, we provide the numerical results concerning a fully-connected network topology with $|\mathcal{K}| = 6$ edge nodes and $|\mathcal{L}| = 15$ communication links. Each edge node has a computation capability $C_k = 5 \times 10^9$ CPU cycles/s. The average delivery delay $d_{mn}$ over $l_{mn} \in \mathcal{L}$ follows an uniform distribution in the interval $[1, 5]$ s/GB. The number of application requests is $|\mathcal{U}| = 3$, each of which contains $|\mathcal{V}_u| = 9$ microservices. For each microservice $v_{ui}$, the computation requirement $m_{ui}$ follows an uniform distribution in the interval $[1, 3] \times 10^9$ CPU cycles. It is assumed that the output size $o_{uij}$ over dependency edge $(v_{ui}, v_{uj})$ is linear to the computation requirement $m_{ui}$, defined as $o_{uij} = \alpha m_{ui}$ with output size coefficient $\alpha \in (0, 1]$ bit/CPU cycles. Here, we have $\alpha = 0.8$ unless explicitly stated. Moreover, the influential strength coefficient $w_{uij}$ over dependency edge $(v_{ui}, v_{uj})$ follows an uniform distribution in the interval $[0.5, 2]$.

To verify the delay performance for different levels of interaction among microservices of applications, we evaluate the delay performance in terms of *the number of cycles* formed by microservices of an application, and the number of microservices per cycle of an application, which we referred to as *size of cycle*. According to Section II, microservices forming a cycle of an application $\mathcal{G}_u^a$ have interaction relationship with each other. The number of cycles is to indicate the overall level of interaction of an application. The more number of cycles of an application, the more microservices probably have interaction relationship with the other, thereby, the stronger interaction happens among microservices. When the number of cycles is 0, the application graph is a directed linear topology.

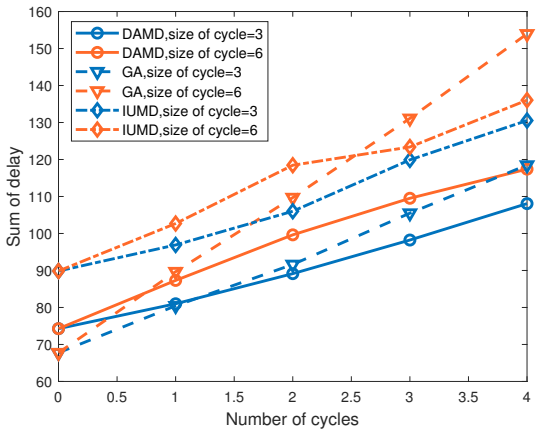


Fig. 2. Sum of delay versus different number of cycles of an application: the blue lines depict the delay performance when there are 3 microservices forming a cycle; the orange lines depict the delay performance when there are 6 microservices forming a cycle.

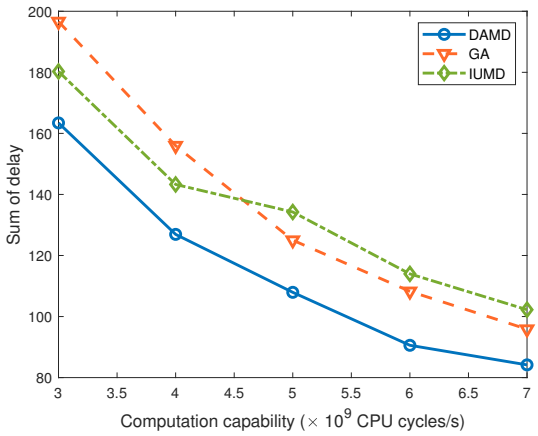


Fig. 3. Sum of delay versus different computation capability of edge nodes.

Similarly, the more of the size of cycles also indicates that more microservices interact with each other. We compare the delay performance with Greedy Algorithm (GA) and Influence-Unaware Microservice Deployment (IUMD). For each non-deployed microservice, GA optimizes deploying and resource allocation decisions to minimize the overall delay of all microservices that already deployed in the network. For IUMD, it optimizes the microservice deployment and resource allocation similar with Algorithm 1 while does not concern with the various influential strength coefficient $W = \{W_u\}_{u\in\mathcal{U}}$ where $W_u = [w_{uij}] \in \{0,1\}^{V_u \times K}$.

First, we evaluate the proposed algorithm in Algorithm 1, referred to as DAMD, versus the number of cycles of each application $\mathcal{G}_u^a$ in Fig. 2. The blue lines depicts the delay performance as the number of microservices forming a cycle (i.e., size of cycle) is 3, and the orange lines depicts the delay performance as the size of cycle is 6. It is shown that DAMD is able to leverage the influential strength coefficient

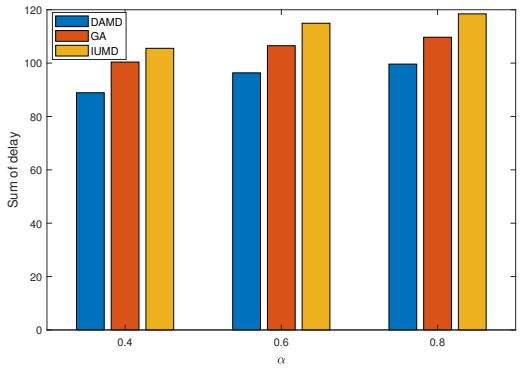


Fig. 4. Sum of delay versus different output size coefficient $\alpha$.

$W$ to get lower sum of delay of applications compared to IUMD. When the number of cycles per application is 0, which means that the application graph is a directed linear topology, it is observed that GA gets slightly lower delay compared to DAMD. It is because that when there are less interactions among microservices, GA does an ergodic search for the optimal deployment and resource allocation decisions for each microservice in sequence, while DAMD converges to a suboptimal solution. As the number of cycles increase from 1 to 4, more microservices have interaction with others, then, DAMD is able to find better deployment scheme and gets decreasing sum of delay compared to GA. Moreover, as the size of cycle increase from 3 to 6, DAMD gets much lower sum of delay compared to GA.

Fig. 3 shows the delay performance under different computation capability $C_k$ of edge nodes. Here, we set the number of cycles per application is 3, and the size of cycle is 6. It is observed that DAMD is better to use limited computation resources compared to GA, thus obtains 16.9%-18.6% less delay when computation capability $C_k \leq 4 \times 10^9$ CPU cycles/s. As computation capability increases, IUMD works worse on delay reduction as for its unconcern of influential strength between dependent microservices.

Finally, we evaluate the delay performance versus different output size coefficient $\alpha$ in Fig. 4. As $\alpha$ increase from 0.4 to 0.8, more amount of outputs are communicated between two dependent microservices. Under different amount of data transmission between microservices, it is observed that DAMD gets at least 15.8% and 9.2% lower sum of delay compared to IUMD and GA, respectively.

## VI. Conclusion

In this paper, the joint microservice deployment and resource allocation problem is studied, in which the interaction among microservices and the influence strength between two dependent microservices are considered. To solve this problem, a coalition game-based algorithm is proposed to minimize the overall delay of multiple applications. Numerical results show that our proposed algorithm outperforms greedy algorithm and influence-unaware microservice deployment scheme

when microservices have strong interaction relationship between each other. Future directions are to consider more performance indicators (energy, reliability), and derive the scheme with respect to the multi-objective optimization.

## REFERENCES


[1] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757-6779, 2017.

[2] B. Sonkoly, J. Czentye, M. Szalay, B. Németh and L. Toka, "Survey on Placement Methods in the Edge and Beyond," *IEEE Commun. Surveys Tut.*, vol. 23, no. 4, pp. 2590-2629, Fourthquarter 2021.

[3] A. Al-Shuwaili and O. Simeone, "Energy-Efficient Resource Allocation for Mobile Edge Computing-Based Augmented Reality Applications," *IEEE Wireless Commun. Lett.*, vol. 6, no. 3, pp. 398-401, June 2017.

[4] L. Chen et al., "IoT Microservice Deployment in Edge-Cloud Hybrid Environment Using Reinforcement Learning," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12610-12622, 15 Aug. 2021.

[5] A. Samanta and J. Tang, "Dyme: Dynamic Microservice Scheduling in Edge Computing Enabled IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6164-6174, July 2020.

[6] T. Bahreini and D. Grosu, "Efficient Algorithms for Multi-Component Application Placement in Mobile Edge Computing," *IEEE Trans. Cloud Comput.*, vol. 10, no. 4, pp. 2550-2563, Dec. 2022.

[7] H. Jin, Y. Jin, H. Lu, C. Zhao and M. Peng, "NFV and SFC: A Case Study of Optimization for Virtual Mobility Management," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2318-2332, Oct. 2018.

[8] J. Chen, Y. Yang, C. Wang, H. Zhang, C. Qiu and X. Wang, "Multi-task Offloading Strategy Optimization Based on Directed Acyclic Graphs for Edge Computing," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9367-9378, June, 2022.

[9] H. Sedghani, F. Filippini and D. Ardagna, "A Randomized Greedy Method for AI Applications Component Placement and Resource Selection in Computing Continua," in *2021 IEEE International Conference on Joint Cloud Computing (JCC)*, United Kingdom, pp. 65-70, 2021.

[10] W. Lv et al., "Microservice Deployment in Edge Computing Based on Deep Q Learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2968-2978, Nov. 2022.

[11] Z. Han, D. Niyato,W. Saad, and A. Hjorungnes, *Game Theory in Wireless and Communication Networks: Theory, Models, and Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2011.