Data Sharing-Aware Algorithms for Task Allocation in Edge Computing Systems

Sanaz Rabinia
Dept. Computer Science
Wayne State University
Detroit, USA
srabin@wayne.edu

Daniel Grosu[§]
Dept. Computer Science
Wayne State University
Detroit, USA
dgrosu@wayne.edu

***Abstract*—Edge computing allows end-user devices to offload heavy computation to nearby edge servers for reduced latency, maximized profit, and/or minimized energy consumption. Data dependent tasks that analyze locally acquired sensing data are one of the most common candidates for task offloading in edge computing. Thus, the total latency and network load are affected by the total amount of data transferred from end-user devices to the selected edge servers. Most existing solutions for task allocation in edge computing do not consider that some user tasks may operate on the same data items. Making the task allocation algorithm aware of the existing data sharing characteristics of tasks can help reduce network load at a negligible profit loss by allocating more tasks sharing data on the same server.**

In this PhD thesis, we formulate the data sharing-aware task allocation problem that makes decisions on task allocation for maximized profit and minimized network load by considering the data-sharing characteristics of tasks. In addition, because the problem is NP-hard, we design and implement an offline algorithm, which finds a good feasible solution to the problem in polynomial time. We also design and implement online algorithms for task allocation in edge computing that take into account the sharing of data among the tasks offloaded to the same server. We analyze the performance of our offline algorithm against a state-of-the-art baseline that only maximizes profit. We also perform an extensive performance analysis by comparing our online algorithms with their sharing-oblivious counterparts.

***Index Terms*—Edge computing, task allocation, online algorithms, offline algorithms, data sharing.**

I. Introduction

Edge computing is a new computing paradigm offering real-time processing and analysis of data closer to the data sources, such as IoT devices, sensors, or end-user devices. This results in reducing latency, bandwidth usage, and faster response times leading to improved overall system efficiency. However, as the adoption of edge computing systems is growing, dealing with increasing amounts of data transferred in the edge network is a significant challenge. One of the main factors that edge providers need to consider to obtain an efficient deployment on edge servers, is exploiting the sharing of data among tasks that can be offloaded to the same server. Designing efficient algorithms for task allocation that consider data sharing among tasks is essential to improving the performance of the edge systems, which generally have limited computational and memory capacities.

[§]PhD advisor

Edge computing facilitates the operations of nearby resource-limited mobile devices such as smartphones, tablets, autonomous mobile robots, drones, and connected vehicles at lower transmission latency compared to the cloud. In fact, many data-driven applications running on mobile devices need computational support to analyze locally-acquired sensor data (e.g., a video or an image from camera, an audio trace from microphone). Typical tasks include face recognition, image classification, and object tracking. To offload a task, each device must transmit all the data items to be analyzed (e.g., camera frames) to one of the nearby available edge servers. On the other hand, given the possibly large number of end-user devices in the edge system and the even larger number of requests, it is important to ensure the scalability of edge resources with respect to the number of tasks and data being offloaded.

The *objective of this PhD dissertation* is to design, study, and implement offline and online data sharing-aware algorithms for task allocation in edge computing systems. The *central hypothesis* is that taking into account the sharing of data among tasks that can be offloaded on the same server, can minimize the total amount of data offloaded on edge systems and the traffic in the edge network that is associated with data transfers.

II. Significance of Our Research

In this PhD thesis, we formulate the *data sharing-aware task allocation problem* as a bi-objective mixed-integer multilinear program that maximizes the profit derived from executing tasks and minimizes the network load by taking into account the data sharing characteristics of tasks. Because this problem is NP-hard, we design a greedy algorithm, called DSTA (*Data Sharing-Aware Task Allocation*), that finds a feasible solution in polynomial time. Specifically, DSTA considers the task data sharing characteristics expressed as a task-data matrix to decide which tasks to allocate on the edge servers by iteratively selecting a subset of tasks that share the highest amounts of data with the already allocated tasks. In each iteration, DSTA maximizes the profit by prioritizing high-profit/light-workload tasks for allocation to the most suitable edge server.

We also take into account the sharing of data among tasks that can be offloaded on the same edge server and formulate the problem of *Online Data Sharing-Aware Task*

Allocation (ODSTA). Managing the sharing of data items among many offloaded tasks on servers, is expected to reduce the total amount of data that needs to be stored at each server, and the traffic in the edge network that is associated with data transfers. We design three online algorithms for solving the ODSTA problem in edge computing systems. We consider an edge computing system composed of a set of heterogeneous servers, where the servers have different capacities for their computational resources (i.e., CPU and memory). Although our online algorithms are inspired by classical *sharing-oblivious* bin packing algorithms such as first-fit, best-fit and worst-fit, they differ from those in that when making allocation decisions they take into account the sharing among the data items stored in the main memory of edge servers.

In summary, this thesis makes the following *contributions*:

- *To the best of our knowledge, our work is the first systematic work to exploit the key intuition that, in edge systems, multiple data-driven tasks from each user device may share some data items*. Taking into account the sharing of data among tasks reduces the traffic in the edge network, the memory usage at the edge servers, and the number of servers needed to execute the tasks, thus, leading to an improved overall performance of the edge system.
- We formulate the data sharing-aware problem as a bi-objective mixed-integer multilinear program that jointly maximizes the task allocation profit and minimizes the network load. *We develop a novel analytical model for capturing the sharing among tasks* and use it to derive the objective functions.
- The formulated data sharing-aware problem is NP-hard. Thus, in order to provide a feasible solution in polynomial time, we *design a greedy algorithm*, called DSTA, that considers the tasks' data-sharing characteristics and iteratively allocates them on edge servers to maximize the profit and minimize the network load.
- We *design three online data sharing-aware algorithms* (DSA) for task allocation in edge computing systems: DSA First-Fit (DSA-FF), DSA Best-Fit (DSA-BF), and DSA Worst-Fit (DSA-WF).
- We compare our proposed DSTA algorithm with a state-of-the-art baseline that only maximizes task execution profit (i.e., P-Greedy). Our results show that DSTA can reduce network data load by about 8x on average at a negligible profit loss compared to P-Greedy.
- We perform an extensive performance analysis by comparing our proposed online algorithms with their sharing-oblivious counterparts. The results show that our algorithms are able to reduce the amount of data transferred in the network by 30.2% to 92.8% and the number of utilized servers by 1% to 82.8% compared to the sharing-oblivious baseline algorithms.

III. Related Work

Task allocation in edge computing has been intensively studied during recent years. Due to the limited computing/energy availability of end-user devices, a significant proportion of related work has focused on offloading task execution to edge servers for lowering end-user energy consumption at a maximum latency requirement [1]–[4]. Other studies have focused on maximizing the quality of service for end-users via task offloading within edge resource constraints [5], [6]. In some cases, edge servers or nearby users may receive some form of profit to provide edge resources for task offloading. Thus, some studies have focused on the topic of finding the best task allocation strategy that maximizes a defined profit in edge systems [7]–[9]. Most of the above studies simply consider the transmission time of data items associated with each offloaded task on the total offloading latency estimation. Some studies provided a more accurate consideration of network packet scheduling for allocating cooperative tasks on edge servers [10]–[12]. However, to the best of our knowledge, *none of the above solutions have considered the fact that multiple tasks from the same user may have to analyze the same data item*. For example, the same camera frame can be used by a task for face recognition and by another task for object detection. Thus, allocating those tasks to different servers without considering that they share data items may lead to the necessity to send the same data item to both servers. On the other hand, allocating those tasks to the same server can help reduce the network load since only one copy of that shared data item needs to be transmitted.

IV. Research Accomplishments

Problem Formulation. In our paper [13], we considered an edge computing system composed of a set $\mathcal{S} = \{S_1, S_2, \ldots, S_M\}$ of M distributed servers, where each server S_j has a limited capacity C_j of computational resources (i.e., CPU cycles). These edge servers serve a set $\mathcal{T} = \{T_1, T_2, \ldots, T_N\}$ of N tasks originating from end-user devices. The set of tasks $\mathcal{T}$ has an associated set of data items, $\mathcal{D} = \{D_1, D_2, \ldots, D_D\}$, that are needed to execute the tasks. We denote the size of data item D_k by d_k, with $k = 1, 2, \ldots, D$. Each task T_i is characterized by a tuple $(r_i, p_i, [A]_{i,*})$, where r_i is the amount of computational resources required by T_i, p_i is the profit for executing T_i, and $[A]_{i,*}$ is the i-th row of the task-data matrix, A. The *task-data matrix* A is a $N \times D$ matrix, where $a_{ik} = d_k$, if task T_i requires data item D_k, and 0, otherwise. The tasks need to be allocated to the servers such that the total profit obtained from executing the tasks is maximized and the total amount of data transferred in the network is minimized. Thus, we formulated the *data sharing-aware task allocation problem* (DSTAP) as a bi-objective mixed-integer multi-linear program:

$$\text{maximize:} \quad \sum_{j=1}^{M} \sum_{i=1}^{N} p_i x_{ij} \tag{1}$$

$$\text{minimize:} \quad \sum_{\mathcal{I} \in \mathcal{P}(\mathcal{T})} (-1)^{(|\mathcal{I}|+1)} \sigma_{\mathcal{I}} \sum_{j=1}^{M} \prod_{i \in \mathcal{I}} x_{ij} \tag{2}$$

subject to:

$$\sum_{i=1}^{N} r_i x_{ij} \leq C_j, \quad \forall j \in \{1, \ldots, M\} \tag{3}$$

$$\sum_{j=1}^{M} x_{ij} \leq 1, \quad \forall i \in \{1, \ldots, N\} \tag{4}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, \forall j \tag{5}$$

The first objective (Equation (1)) is to maximize the total profit. The decision variable x_{ij} is 1, if task T_i is allocated to server S_j, and 0, otherwise. The second objective (Equation (2)) is to minimize the total amount of data offloaded from user devices to the servers, which depends on the decision variables x_{ij} and on the data sharing among tasks. Here, $\mathcal{P}(\mathcal{T})$, is the power set of the set of indices of the tasks in $\mathcal{T}$, and $\mathcal{I}$ is an element of the power set. We define the *sharing parameter*, $\sigma_{\mathcal{I}}$, as the total amount of data shared among the tasks whose indices are in set $\mathcal{I}$. In our paper [13], we give more details on how the sharing parameter is computed and explain how the cost function (2) captures the sharing of data and gives the total amount of data in the network. Constraint (3) ensures that the total allocated computational requests to a server does not exceed the capacity of the server. Constraint (4) ensures that each task is allocated to only one server, while Constraint (5) guarantees the integrality of the decision variables.

DSTA Algorithm. We design an offline greedy algorithm, called DSTA that takes into account the sharing characteristics of the tasks when deciding which tasks to allocate on the edge servers. That is, it iteratively selects a subset of tasks that share the highest amounts of data with the tasks that are already allocated. In each iteration, it establishes a greedy order among these tasks, that is induced by a function that prioritizes high-profit and light-workload tasks for allocation to the most suitable edge server. In our proposed offline algorithm, we defined an *efficiency function* which is used to establish the greedy order among the tasks. The tasks will be considered for allocation in the order provided by this greedy order. The efficiency function is defined by $E_i = \frac{p_i}{\sqrt{\frac{r_i}{\sum_{j=1}^{M} C_j}}}$. The efficiency function for a given task can be viewed as a density measure, computed as the profit obtained from executing the task divided by the square root of the relative size of the request. Here, the relative size of the request is with respect to the total capacity of the servers in the system. This efficiency function allows the algorithm to allocate the tasks in the order of their highest profit density and therefore obtain high values for the total profit gained from executing the tasks.

Online Algorithms. In our recent work [14], we designed three online algorithms (DSA-FF, DSA-BF, and DSA-WF) for task allocation in edge computing systems that take into account the sharing of data among tasks offloaded to the same server. To characterize the sharing of data for the upcoming task T_j hosted on server S_k for the online task allocation problem, we define δ_j^k as the amount of data (memory) shared among the upcoming task T_j and all the tasks already assigned to server S_k, that is, $\delta_j^k = \sum_{l=1}^{D} a_{jl} \cdot [a_{jl} = \Delta_l^k]$, where $[C]$ is the Iverson's bracket and Δ_l^k is entry l of *data allocation vector* Δ^k specifying the data items currently allocated to server S_k. The Iverson bracket $[C]$ evaluates to 1 if the condition C is true, and 0, otherwise. The data allocation vector Δ^k has size D and is defined recursively as follows. If no task is allocated to S_k then $\Delta^k = [0]_D$. Let Δ^k be the vector before task T_j is allocated to server S_k and $\tilde{\Delta}^k$ the vector after T_j is allocated to S_k. Then, $\tilde{\Delta}^k = \Delta^k \boxplus [A]_{j,*}$, where $\boxplus$ is defined as follows,

$$\Delta^k \boxplus [A]_{j,*} = \begin{cases} \Delta_l^k & \text{if } a_{jl} = 0 \\ a_{jl} & \text{if } a_{jl} \neq 0 \end{cases} \tag{6}$$

Let Δ^k be the vector before task T_j is deallocated from server S_k and $\hat{\Delta}^k$ the vector after T_j is deallocated from S_k. Then, $\hat{\Delta}^k = \Delta^k \boxminus [A]_{j,*}$, where $\boxminus$ is defined as follows,

$$\Delta^k \boxminus [A]_{j,*} = \begin{cases} \Delta_l^k & \text{if } (\Delta_i^k = a_{jl}) \wedge (\exists T_i \in \mathcal{S}_k, i \neq j, a_{il} \neq 0) \\ 0 & \text{if } (\Delta_i^k = a_{jl}) \wedge (\forall T_i \in \mathcal{S}_k, i \neq j, a_{il} = 0) \\ \Delta_l^k & \text{if } a_{jl} = 0 \end{cases} \tag{7}$$

and $\mathcal{S}_k$ is the set of tasks allocated to server S_k.

A task T_j can be assigned to a server S_k if the following constraints are satisfied:

$$\tilde{C}_k^m - \sum_{l=1}^{D} a_{jl} + \delta_j^k \geq 0, \quad \forall T_j \in \mathcal{T}, \forall S_k \in \mathcal{S} \tag{8}$$

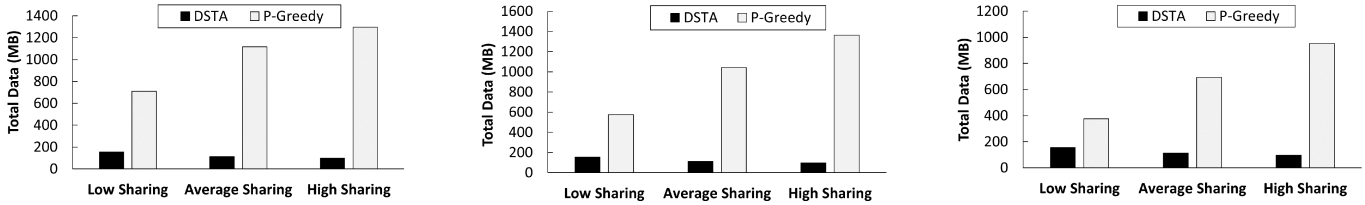
$$\tilde{C}_k^u - r_j^u \geq 0, \quad \forall T_j \in \mathcal{T}, \forall S_k \in \mathcal{S} \tag{9}$$

where $\tilde{C}_k^m$ and $\tilde{C}_k^u$ are the available memory and CPU capacities at server S_k. If there is no task assigned to server S_k, $\tilde{C}_k^m = C_k^m$ and $\tilde{C}_k^u = C_k^u$. Equation (8) is the memory capacity constraint for server S_k, guaranteeing that the available amount of memory of server S_k is not exceeded by allocating the memory needed for task T_j. Equation (9) is the CPU capacity constraint, guaranteeing that the CPU capacity of server S_k is not exceeded by allocating task T_j.

To design DSA-BF, we define an *efficiency metric* that characterizes the scarcity of resources at each server. The classical best fit bin packing algorithm attempts to place each new item into the bin with the maximum load (smallest available space) in which it fits. Thus, if such a bin is found, the new item is placed inside it. To find the maximum load in the case of our edge system, where each server is characterized by two main parameters, CPU capacity and memory size, we define the efficiency metric E_j^k as follows,

$$E_j^k = \begin{cases} \frac{1}{\sqrt{\alpha[\tilde{C}_k^u - r_j] + \beta[\tilde{C}_k^m - (\sum_{l=1}^{N} a_{jl} - \delta_j^k)]}} & \text{if } (\tilde{C}_k^u - r_j \geq 0) \\ & \wedge (\tilde{C}_k^m - (\sum_{l=1}^{N} a_{jl} - \delta_j^k) \geq 0) \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where α and β are the weight parameters for CPU capacity and memory size. The weights are used to tune the algorithm, for example giving more importance to CPU or memory when making the allocation decisions. Finding the server with the maximum load is equivalent to finding the server with the minimum of available resource capacity. Thus, in E_j^k we consider



(a) Low Demand (b) Average Demand (c) High Demand

Fig. 1: DSTA vs. P-Greedy: Total amount of data on the network (a-c) for various combinations of workload demand and data sharing characteristics (exponential distribution) across offloaded tasks.

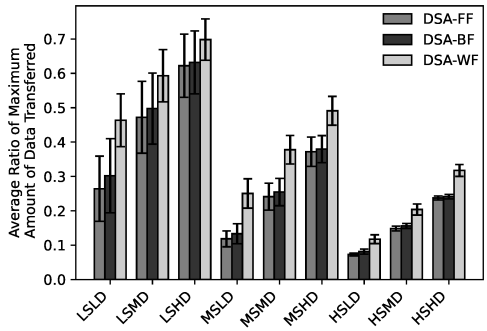


Fig. 2: Online algorithms: Relative maximum amount of data transferred.

the available capacity for CPU and memory i.e., $\tilde{C}_k^u - r_j$ and $\tilde{C}_k^m - (\sum_{l=1}^{N} a_{jl} - \delta_j^k)$, respectively. It is important to mention that the efficiency metric E_j^k takes into account the sharing of memory among tasks that are offloaded to the same server S_k.

V. Experimental Analysis

Experimental Setup. To characterize the relationship between server's total capacity (in GIPS) and the total amount of computational request, we define a demand ratio, ρ as follows: $\rho = \frac{\sum_{i=1}^{N} r_i}{\sum_{j=1}^{M} C_j}, \quad 0 \leq \rho \leq 1$. Based on this ratio, we consider three different cases for computational requests by tasks: (i) *low demand* (LD) ($\rho \ll 1$); (ii) *medium demand* (MD) ($\rho \approx 1/2$); and (iii) *high demand* (HD) ($\rho \approx 1$).

We utilize the Erdös-Rényi random graph model [15] to create the bipartite graph that represents the data sharing pattern of the generated taskset. The adjacency matrix of this graph is used as the task-data matrix A. In the Erdös-Rényi model, for a graph with n vertices and m edges, the probability of generating each edge is given by: $p^m(1-p)^{\binom{n}{2}-m}$. The parameter $p \in [0, 1]$, called the *sharing degree* here, characterizes the sharing among tasks. We generate instances for three different cases: (*i*) *low sharing* (LS) ($0 \leq p \leq 0.3$); (*ii*) *medium sharing* (MS) ($0.3 < p \leq 0.6$); and (*iii*) *high sharing* (HS) ($0.6 < p \leq 1$).

DSTA: Network Data Load Analysis. Figures 1a-1c, show the total data size in the network corresponding to the allocation obtained by DSTA and P-Greedy (a state of the art algorithm that maximizes the total profit only). In all three demand cases, the total size of data corresponding to the allocation obtained by DSTA is much smaller than that corresponding to P-Greedy. In Figure 1a, for low demand case, the data size corresponding to P-Greedy is 4.59, 9.82, 13.32 times greater than that of DSTA for low sharing, average sharing, and high sharing scenarios, respectively. In Figure 1b, for average demand case, the data size corresponding to P-Greedy is 3.73, 9.16, and 14 times greater than that of DSTA for low sharing, average sharing, and high sharing cases, respectively.

Online algorithms: Maximum amount of data transferred in the edge network. For each of the online algorithms we recorded the amount of data transferred in the edge network at each task arrival and termination and determine the maximum among those values. Then, we determine the ratio of the maximum amount of data transferred in the case of the proposed data sharing-aware algorithms and that of the sharing-oblivious algorithms. Figure 2, shows this ratio for all nine types of instances with different levels of sharing and demand. The ratio decreases as the sharing of data items among the tasks increases. For example, in the case of LSLD, DSA-FF leads to 74% less data transferred in the network than FF (corresponding to a ratio of 0.26), while in the case of HSLD, DSA-FF leads to 92.8% less data transferred than FF. A similar behavior is exhibited by DSA-BF which obtains a little bit smaller reduction in the data transferred for those cases (about 70% for LSLD and 92% for HSLD).

VI. Conclusion and Future Work

In this thesis, we designed several online and offline algorithms for task allocation in edge computing systems that take into account the sharing of data among tasks offloaded to the same server. Taking into account the sharing of data among tasks reduces the traffic in the edge network, the memory usage at the edge servers, and the number of servers needed to execute the tasks, thus, leading to an improved overall performance of the edge system. In our future work, we plan to explore further optimizations of these algorithms to improve their performance and also determine their theoretical performance guarantees.

Acknowledgment

This research was supported in part by the US National Science Foundation under Grants no. CCF-2118202 and CNS-2231523.

REFERENCES

[1] X. Chen and G. Liu, "Joint optimization of task offloading and resource allocation via deep reinforcement learning for augmented reality in mobile edge network," in *Proc. IEEE Int. Conf. on Edge Computing (EDGE)*, 2020, pp. 76–82.

[2] X. Huang, L. He, and W. Zhang, "Vehicle speed aware computing task offloading and resource allocation based on multi-agent reinforcement learning in a vehicular edge computing network," in *Proc. IEEE Int. Conf. on Edge Computing (EDGE)*, 2020, pp. 1–8.

[3] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Trans. on Communications*, vol. 65, no. 8, pp. 3571–3584, 2017.

[4] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "Toffee: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Trans. on Cloud Computing*, pp. 1–1, 2019.

[5] C. Avasalcai, C. Tsigkanos, and S. Dustdar, "Decentralized resource auctioning for latency-sensitive edge computing," in *Proc. IEEE Int. Conf. on Edge Computing (EDGE)*, 2019, pp. 72–76.

[6] H. Badri, T. Bahreini, D. Grosu, and K. Yang, "Risk-aware application placement in mobile edge computing systems: A learning-based optimization approach," in *Proc. IEEE Int. Conf. on Edge Computing (EDGE)*, 2020, pp. 83–90.

[7] D. Zhang, Y. Ma, C. Zheng, Y. Zhang, X. S. Hu, and D. Wang, "Cooperative-competitive task allocation in edge computing for delay-sensitive social sensing," in *Proc. IEEE/ACM Symp. on Edge Computing (SEC)*, 2018, pp. 243–259.

[8] D. Y. Zhang and D. Wang, "An integrated top-down and bottom-up task allocation approach in social sensing based edge computing systems," in *Proc. IEEE Conf. Comp. Comm. (INFOCOM)*, 2019, pp. 766–774.

[9] A. Kiani and N. Ansari, "Toward hierarchical mobile edge computing: An auction-based profit maximization approach," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2082–2091, 2017.

[10] Q. Chen, Z. Zheng, C. Hu, D. Wang, and F. Liu, "Data-driven task allocation for multi-task transfer learning on the edge," in *Proc. 39th IEEE Int. Conf. Distrib. Comp. Syst. (ICDCS)*, 2019, pp. 1040–1050.

[11] Y. Sahni, J. Cao, and L. Yang, "Data-aware task allocation for achieving low latency in collaborative edge computing," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3512–3524, 2019.

[12] H. Xing, L. Liu, J. Xu, and A. Nallanathan, "Joint task assignment and wireless resource allocation for cooperative mobile-edge computing," in *Proc. IEEE Int. Conf. on Communications (ICC)*, 2018, pp. 1–6.

[13] S. Rabinia, H. Mehryar, M. Brocanelli, and D. Grosu, "Data sharing-aware task allocation in edge computing systems," in *Proc. IEEE Int. Conf. on Edge Computing*, 2021, pp. 60–67.

[14] S. Rabinia and D. Grosu, "Data sharing-aware online algorithms for task allocation in edge computing systems," in *to be submitted to IEEE Int. Conf. on Edge Computing*, 2024.

[15] P. Erdös and A. Rényi, "On random graphs i," *Publicationes Mathematicae Debrecen*, vol. 6, p. 290, 1959.