Age-of-Information and Availability-Aware Service Provisioning in Edge-Enabled Digital Twin Networks

Pranshu Jaiswal and Prakhar Kapisway
Department of Computer Science and Engineering
Indian Institute of Technology Jammu, 181221, India
Email: {2021ucs0107, 2021ucs0106}@iitjammu.ac.in

Samaresh Bera, *Senior Member, IEEE*
Department of Computer Science and Engineering
Indian Institute of Technology Jammu, 181221, India
Email: s.bera.1989@ieee.org

***Abstract*—Digital twin (DT) systems have gained significant traction in edge computing environments, where maintaining real-time synchronization between physical and virtual entities is crucial. Age of Information (AoI) has emerged as a key metric for capturing the freshness of data in such systems. This paper studies the service placement and request routing problem in a digital twin network, where availability and AoI are the primary aspects taken into consideration. The studied problem consists of two phases – master node selection and resource allocation. In the first phase, we formulate an ILP to select master nodes equipped with higher resources for a given edge-enabled digital twin networks. In the second phase, we formulate a multi-constrained optimization problem for service placement and request routing in the digital twin network. We propose a self-tuned greedy approach to solve the formulated NP-hard optimization problem in polynomial time. Simulation results show that the proposed approach yields competitive performance compared to the optimal solution. Furthermore, the results demonstrate the system's efficiency and resource utilization compared to the existing approaches.**

***Index Terms*—Edge network, Digital twin, Resource allocation, Optimization**

I. Introduction

The advent of Industry 4.0 and the proliferation of Internet of Things (IoT) technologies have significantly increased the demand for real-time, intelligent, and scalable systems. Digital Twin (DT), which creates virtual replicas of physical entities, has emerged as a key enabler for such systems, offering real-time monitoring, simulation, and control capabilities [1]. The effectiveness of a digital twin heavily depends on low-latency communication, high service availability, and efficient computation—particularly when deployed in dynamic and large-scale networks. Edge computing has gained traction as a viable solution to meet the stringent latency requirements of digital twin systems by pushing computation and data storage closer to the data sources [2]. However, the resource-constrained nature of edge network poses significant challenges in meeting availability requirements of digital twin applications. Therefore, the highly distributed and heterogeneous nature of edge networks presents significant challenges for service placement and request routing, especially when age-of-information (AoI) and availability are essential, to ensure contextual relevance and minimize communication overhead.

In this paper, we study the problem of AoI and availability-aware service placement and request routing in edge networks for digital twin applications. We model the network as a graph of compute nodes with varying capacities and availability levels, and propose a two-step approach: (i) selection of master nodes based on reachability and hop count, and (ii) optimization of request allocation with respect to delay, resource availability, and utility. To address the inherent complexity of the joint placement and routing problem, we present a greedy algorithm that prioritizes service requests based on a utility-to-cost ratio, considering availability, delay tolerance, and resource requirements. Our solution aims to maximize the overall utility of the system while satisfying all network constraints. The key contributions of this paper are as follows:

- The edge-enabled digital twin network is equipped with two types of nodes – master node and worker node. The master nodes in the network are equipped with higher resources and have higher availability compared to the worker nodes to which the requests are associated with. Therefore, we propose a hop-based integer linear programming (ILP) problem to identify a minimal set of master nodes for centralized processing in the edge network to minimize the CAPEX for network deployments.
- We formulate the problem of AoI and availability-aware service placement and request routing for digital twin applications in edge networks as a constrained optimization problem.
- We propose a self-tuned utility-aware greedy algorithm that efficiently allocates requests to compute nodes while meeting service availability, delay, and resource constraints. Simulation results show that the proposed greedy approach yields competitive perfor-

mance to the optimal solution and outperforms the existing approaches.

The remainder of this paper is organized as follows. Section II discusses the related work. Section III presents the network model and problem statement for master node selection. Section IV presents the delay and availability model, followed by the problem statement for service placement and request routing in digital twin network. Section V outlines our proposed solution approach. Section VI presents the simulation results. Finally, Section VII concludes the paper with some future research directions.

II. Related Work

Age of Information (AoI) has become a key metric for evaluating information freshness in edge-enabled digital twin (DT) systems. Recent work has integrated AoI into service provisioning and inference optimization frameworks [1], [3]–[9]. We discuss some of the existing works as follows.

Li et al. [6] introduced an AoI-based resource allocation framework using digital twin network slicing. Their method improves data freshness through dynamic service placement while accounting for latency and system heterogeneity. Similarly, Zhang et al. [1] focused on optimizing inference tasks under AoI constraints. The authors proposed a joint model for service slicing and request routing, achieving lower AoI and higher inference accuracy across distributed edge nodes.

Farhadi et al. [3] proposed approaches for efficient service placement and request scheduling in edge clouds, addressing resource constraints and dynamic demands. The authors proposed a two-time-scale framework that jointly optimizes service placement and request scheduling. Wang et al. [7] proposed an edge-assisted DT to monitor physical objects and environments to determine optimal strategy that can be applied in real-time for different deployment scenarios.

While these studies provide effective AoI-aware strategies, several challenges remain unaddressed as follows: a) limited focus on availability essential for real-world applications; and b) absence of AoI-sensitive frameworks tailored for DT systems. This paper addresses these gaps by proposing an edge-enabled digital twin network that considers AoI minimization, resource constraints, and system availability, enabling more reliable and efficient digital twin services at the edge.

III. Phase I: Master Node Selection

Let there be an edge network $G(N, E)$, where N is the set of nodes and E is the set of edges (links). Each node $n \in N$ has certain CPU and RAM resources available to host a digital twin, denoted by C_n and D_n, respectively. Each link $(i, j) \in E$ is assigned delay denoted by $\lambda_{i,j}$. Due to the softwarized placement of services on the nodes [10], each node $n \in N$ is also associated with an availability denoted by a_n. Considering the edge networks, all nodes cannot be equipped with high resources and availability like a centralized cloud server. Therefore, we select a subset of nodes, called `master nodes`, with higher resources and availability in edge network itself, compared to all other nodes, called `worker nodes`. The set of master nodes and worker nodes is denoted as M and W, where $N = M \cup W$. We consider the hop count and in-degree (reachability) of each node to select the minimum number of master nodes. We define the cost function for each node $i \in N$ as:

$$c_i = \frac{1}{\deg(i) + 1}, \quad \forall i \in N, \tag{1}$$

where $\deg(i)$ is the in-degree of a node $i \in N$. We note that a node with higher in-degree provides higher reachability from other nodes, and thereby reduces the cost if the node is selected as a master node. Furthermore, the denominator in (1) ensures the working of the cost function in presence of a node with zero in-degree.

The objective is to minimize the number of master nodes while considering the associated constraints. Mathematically,

$$\min \sum_{j=1}^{|N|} w_j c_j, \tag{2}$$

subject to

$$\sum_j \beta_{i,j}^h \geq 1, \quad \forall i \in N, i \neq j, \tag{3a}$$

$$\beta_{i,j}^h \leq w_j, \quad \forall i, j \in N, \tag{3b}$$

$$w_j \in \{0, 1\}, \quad \forall j \in N, \tag{3c}$$

where w_j indicates whether the j-th node is selected as a master node (binary variable) denoted in (3c). Equation (3a) ensures that for each node i, there exists at least one master node j within the desired hop count h. Equation (3b) presents the relationship between hop count and number of master node selection, where $\beta_{i,j}^h$ is defined as:

$$\beta_{i,j}^h = \begin{cases} 1, & \text{if a master node is reachable} \\ & \text{from node } j \text{ within hop count } h \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Figure 1 shows a schematic diagram of a digital twin with worker and master nodes, where the shaded nodes are `master` nodes. Both the master and worker nodes are equipped with compute and storage resources.

IV. Phase II: Service Placement and Request Routing

For the given network with master and worker nodes, we focus on the service placement and request routing for incoming service requests denoted by a set R. Each request $r \in R$ has specific CPU and RAM resource requirements, denoted by c_r and d_r, respectively. Furthermore, each

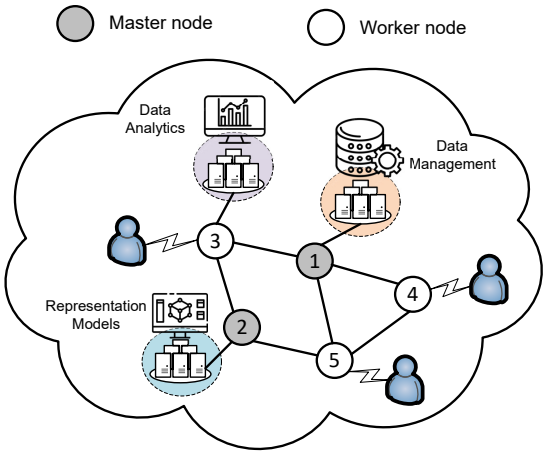


Fig. 1: A digital twin network with master and worker nodes with different services hosted by the nodes

request has a threshold a_r for service availability and is associated with a worker node denoted by $w_r \in W$. Each request $r \in R$ is also associated with a utility U_r, which is obtained by serving the request, and a task size s_r. We consider a generalized utility associated with a request. However, the utility of can be modeled as a function of CPU, RAM, and other networking resources required by the request. To provide the services to the requests, associated service functions need to be placed either at the worker node and/or at the master nodes. Furthermore, request routing needs to happen through the service functions. We discuss the availability and delay models in subsequent sections.

A. Availability Model

A service hosted by a master/worker node is available when both the node and the hosted service are available. A failure of the service can happen at a node due to the hardware failure of the node itself and/or the software failure of the hosted service [10]. Therefore, availability is an important aspect in serving the requests in a digital twin network. We consider that the worker nodes have a lower availability compared to the master node, as the former is placed with limited hardware resources and directly accessible to the users. Moreover, we consider the same availability for all worker nodes unlike master nodes, which are having different availability due to heterogeneous computing and hardware resources. The availability of a node is denoted as a_n, $\forall n \in N$.

B. Delay Model

The total delay associated with task computation includes the delays from local computation on a worker and offloading to a master node:

$$\Delta = \begin{cases} \Delta_{\text{proc}}^{\text{loc}}, & \text{if locally computed,} \\ \Delta_{\text{trans}} + \Delta_{\text{prop}} + \Delta_{\text{que}} + \Delta_{\text{proc}}^{\text{mast}}, & \text{otherwise.} \end{cases}$$

Δ_{trans}, Δ_{prop}, and Δ_{que} represent the transmission, propagation, and queuing delays, respectively. $\Delta_{\text{proc}}^{\text{loc}}$ and $\Delta_{\text{proc}}^{\text{mast}}$ denote the processing delays at the worker node and master node, respectively. We present the modeling of these delays as follows:

1) Processing Delay: The processing delay of request r served by node n is presented as follows:

$$\Delta_{\text{proc},n,r} = \frac{s_r}{c_j}, \forall n \in N, \forall r \in R, \tag{5}$$

where s_r denotes the data size of the request r, and c_n denotes the computational capacity of the node $n \in N$.

2) Transmission Delay: The transmission delay of request r depends on the data-size and the channel capacity. Mathematically,

$$\Delta_{\text{trans},r} = \frac{s_r}{B \log_2(1+\sigma)}, \forall r \in R, \tag{6}$$

where B is the bandwidth of the channel. The symbol σ denotes the signal-to-noise ratio, where the channel is modeled with AWGN channel.

3) Propagation Delay: The propagation delay of request r is modeled as:

$$\Delta_{\text{prop},r} = \sum_{(i,j)\in E} \lambda_{i,j} y_{i,j}^r, \forall (i,j) \in E, \forall r \in R, \tag{7}$$

where $\lambda_{i,j}$ denotes the propagation delay of link (i,j). The binary variable $y_{i,j}^r$ denotes whether the link (i,j) is used to route the offloaded request r to the master nodes.

4) Queuing Delay: The queuing delay of request r depends on the execution and arrival rates at the associated master mode. Mathematically,

$$\Delta_{\text{que},j,r} = \frac{1}{\epsilon_j - \zeta_j}, \forall j \in M, \forall r \in R, \tag{8}$$

where ϵ_j denotes the task execution rate at the master node $j \in M$. The symbol ζ_j denotes the average request allocation rate at the master node $j \in M$. We note that the queuing delay is not applicable when the request is processed at the worker node itself. This is because a single task is generated and received by the worker node at a time. Therefore, total delay experienced by request r is calculated as: $\Delta_{n,r} = \Delta_{\text{trans},r} + \Delta_{\text{prop},r} + \Delta_{\text{que},j,r} + \Delta_{\text{proc},n,r}, \forall r \in R, \forall n \in N, \forall j \in M$.

C. Problem Statement

The objective is to maximize the total utility by serving the requests while considering the associated constraints. Mathematically, we formulate the optimization problem as follows:

$$\max \sum_{r\in R} \sum_{n\in N} U_r x_{r,n}, \tag{9}$$

subject to

$$a_r x_{r,n} \leq a_n, \quad \forall r \in R, \forall n \in N, \tag{10a}$$

$$\sum_{r \in R} c_r x_{r,n} \leq C_n, \quad \forall n \in N, \tag{10b}$$

$$\sum_{r \in R} d_r x_{r,n} \leq D_n, \quad \forall n \in N, \tag{10c}$$

$$\sum_{n \in N} x_{r,n} \leq 1, \quad \forall r \in R, \tag{10d}$$

$$\Delta_{n,r} x_{r,n} \leq \Delta_r, \quad \forall r \in R, \forall n \in N, \tag{10e}$$

$$x_{r,n} = 0, \quad \forall n \in W \setminus \{w_r\}, \tag{10f}$$

$$x_{r,n} \in \{0,1\}, \quad \forall r \in R, \forall n \in N, \tag{10g}$$

$$y^r_{i,j} \in \{0,1\}, \quad \forall r \in R, \forall (i,j) \in E, \tag{10h}$$

$$\sum_{(i,j) \in E} y^r_{i,j} - \sum_{(j,i) \in E} y^r_{j,i} = \begin{cases} x_{r,n}, \text{if } i = w_r, \\ -x_{r,n}, \text{if } i = n, \\ 0, \text{otherwise.} \end{cases} \tag{10i}$$

Equation (9) represents the objective, which is to maximize the total utility. Equation (10a) denotes the availability constraint, where the availability requirement of a request $r \in R$ should be fulfilled by the compute node $n \in N$. The compute and storage constraints are captured in (10b) and (10c), respectively. Equation (10d) denotes that the request r is served by at most one node, either worker node or a master node. The constraint on delay requirement is captured in (10e). A request cannot be served by a worker node except the one associated with the request, as presented in (10f). In the optimization problem, two types of binary decision variables are present – $x_{r,n}$ for request admission and $y^r_{i,j}$ for request routing. $x_{r,n}$ is equal to 1 if request r is served by node n, else 0, and it is presented in (10g). Similarly, $y^r_{i,j}$ is equal to 1 if request r is routed through link $(i,j) \in E$, and it is presented in (10h). Finally, Equation (10i) considers the flow conservation rule associated with request routing. The formulated optimization problem is NP-hard.

V. Solution Approach

To solve the optimization problem in (9) in polynomial time, we propose a utility-to-cost ratio-based greedy approach for serving incoming requests.

A. Maximum Utility-to-Cost Ratio First

This approach prioritizes requests based on utility-to-cost ratio, where we design the following cost function:

$$\Phi_r = \alpha a_r + \beta \cot^{-1}\left(\frac{\Delta_{\max} - \Delta_r}{\Delta_{\max}}\right) + \gamma \frac{d_r}{d_{\max}} + \delta \frac{c_r}{c_{\max}},$$

where α, β, γ, and δ are the predefined constants tuned for a given network deployment and request distribution, which is discussed in the subsequent section. $\Delta_{\max}$, $d_{\max}$, and $c_{\max}$ denote the maximum delay, storage and compute requirements among all the requests.

Algorithm 1 Proposed utility-based greedy algorithm

Inputs: Set of worker nodes: W
Set of master nodes: M
Set of all nodes: N, each node $n \in N$ with a_n, C_n, and D_n;
Set of Requests: R, each request $r \in R$ with a_r, c_r, d_r, U_r, s_r, and associated worker node w_r;

Output: Binary allocation of requests to worker and master nodes

```
R̂ ← Sort requests in descending order of U_r/Φ_r, ∀r ∈ R
for each request r ∈ R̂ do
    for each node n ∈ {w_r ∪ M} do
        flag ← CHECK_FEASIBILITY(r, n)
        if flag then
            x_{r,n} = 1
        else
            x_{r,n} = 0
function CHECK_FEASIBILITY(r, n)
    Check availability: a_r ≤ a_n
    Check compute: Σ_{r∈R} c_r ≤ C_n
    Check storage: Σ_{r∈R} d_r ≤ D_n
    Check delay: Δ_n ≤ Δ_r
    if All constraints satisfied then
        return TRUE
    else
        return FALSE
```

The time complexity of the proposed approach is $O(|R|\log|R| + |R||N|)$, where $|R|$ be the number of requests and $|N|$ be the number of nodes. Where the first part $O(|R|\log|R|)$ is due to the sorting the requests, and the second part $O(|R||N|)$ is for serving the requests.

B. Tuning the Predefined Constants

We use the Bayesian optimization approach [11] and utilize the Gaussian process (GP) upper confidence bound (UCB) to tune the constants. The GP is used to construct a posterior of the objective function and UCB is used as the acquisition function. Mathematically, the acquisition function of the proposed greedy approach is denoted by [12]: $\text{UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \kappa\iota(\mathbf{x})$, where $\mathbf{x}$ is a vector with 4 dimensions, each representing one constant. $\mu(\mathbf{x})$ denotes the posterior mean and $\iota(\mathbf{x})$ denotes the variance at $\mathbf{x}$, and κ is a system parameter that controls the importance of exploration over exploitation. In each iteration, the Bayesian optimization approach finds out the $\mathbf{x}$ for which $\text{UCB}(\mathbf{x})$ is maximized. It then determines where to sample next from the 4-dimensional search-space $[0,1]^4$. We note that methods such as gradient-decent cannot be used because we do not have a closed-form expression of the objective function that can be used to maximize the total reward.

We use RayTune [13] to frame this tuning problem. In RayTune, we set the number of samples to explore as 50

and the number of trials as 40. For each trial, we generate a total of 50 service requests. The constants are tuned given the network topology, networking resources, and request-specific requirements, which are reported in Sec. VI. We choose the sample that yields the maximum mean utility over all the trials.

VI. Performance Evaluation

We simulate the digital twin in a softwarized platform with the parameters presented in Table I.

TABLE I: Simulation parameters

Parameter	**Value**
Network topology	AttMpls [14]
Total number of nodes	25
Hop-count threshold	3
Compute resource of master nodes	[500, 1000]
Storage resource of master nodes	[500, 1000]
Compute resource of worker nodes	[200, 400]
Storage resource of worker nodes	[200, 500]
Availability of master nodes	[0.9, 1.0]
Availability of worker nodes	0.8
Propagation delay of a link	[0.05, 0.2]
Request arrival distribution	Poisson process
Compute requirement of requests	[60, 150]
Storage requirement of requests	[150, 450]
Availability requirement of requests	[0.75, 0.99]
Delay requirement of requests	[3, 10]
Utility of requests	[5, 10]
Self-tuned [α, β]	[0.375, 0.951]
Self-tuned [γ, δ]	[0.599, 0.732]

We compare the proposed greedy approach with optimal solution, where we solve the optimization problem using `IBM CPLEX` to get the optimal solution. We use the tuned value of the predefined constants to calculate the utility-to-cost ratio of a request. Furthermore, we consider two other greedy approaches – without considering availability requirements, such as in [7], called `U-WoAVL` and random allocation, called `U-Random`. In `U-WoAVL`, the requests are allocated based on their utilities similar to the proposed approach while considering the associated constraints except the availability. Whereas requests are allocated in random order in case of `U-Random` considering the associated constraints and utilities.Henceforth, we represent the optimal solution, proposed greedy approach, greedy without availability, and greedy with random allocation by `ILP`, `Proposed`, `U-WoAVL`, and `U-Random`, respectively. We consider the following performance metrics – total utility, percentage of admitted requests, percentage of resource utilization, and computation time. The simulation results are discussed in subsequent sections.

A. Results and Discussion

1) Total Utility: Figure 2 represents the total utility achieved by each scheduling method as the number of requests increases. `ILP` and `Proposed` methods achieve significantly higher total utility than `U-WoAVL` and `U-Random`. The `Proposed` method performs close to `ILP` while being more computationally efficient. `U-WoAVL` and `U-Random` yield lower utility, suggesting poor decision-making in resource distribution. Specifically, `U-WoAVL` provides the lowest utility when availability is not considered. Maximizing total utility indicates better service quality and effective scheduling.

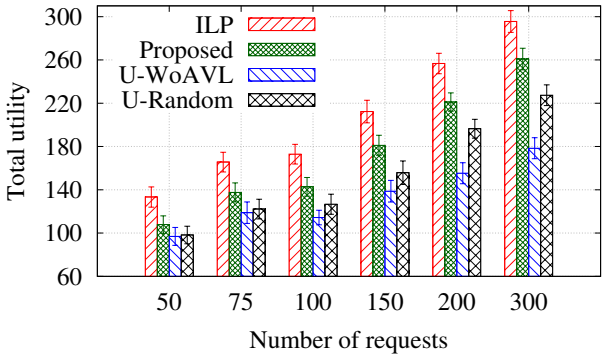


Fig. 2: Total utility with different number of requests

2) Percentage of Admitted Requests: Figure 3 shows the percentage of admitted requests compared to the total number of incoming requests. The `Proposed` method outperforms `U-WoAVL` and `U-Random` in admitting a higher percentage of requests. The `ILP` method has slightly better or similar admission rates compared to the `Proposed` method. `U-WoAVL` and `U-Random` methods admit fewer requests. In `U-WoAVL` requests are virtually allocated the resources, but may not be served physically due to the availability requirements. On the other hand, `U-Random` yields a low admission percentage due to inefficient allocation strategies. A higher admission rate implies better system efficiency and resource allocation.

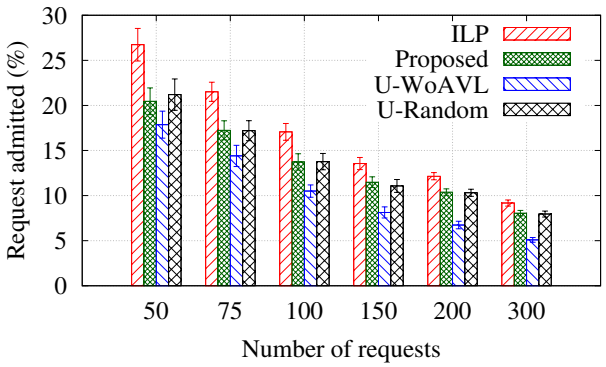
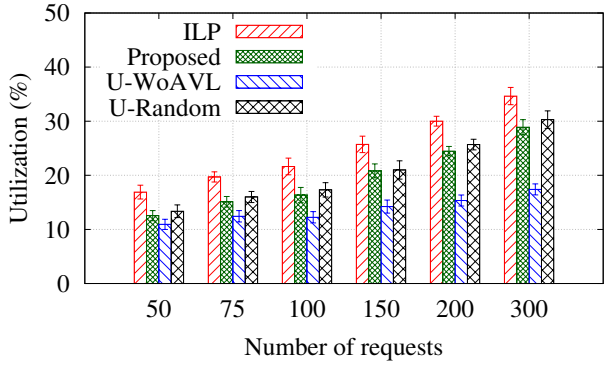


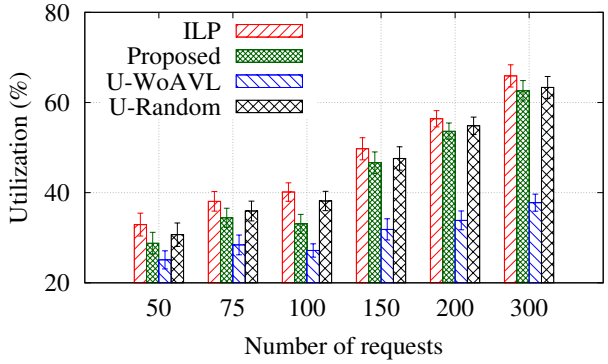
Fig. 3: Percentage of requests admitted into the network

3) Resource Utilization: Figure 4(a) illustrates how CPU usage (%) changes as the number of requests increases. `ILP` has higher CPU utilization compared to the `Proposed` method. The Proposed approach efficiently utilizes CPU resources without excessive load. `U-WoAVL` and `U-Random` have relatively lower CPU utilization, which indicate sub-optimal resource allocation. The Proposed method achieves a balance between performance and resource efficiency.

Figure 4(b) presents the percentage of RAM utilized for different request loads. `ILP` and `Proposed` methods tend to have higher RAM utilization than `U-WoAVL` and `U-Random`. `U-WoAVL` and `U-Random` approaches utilize less RAM, potentially indicating inefficient resource management. The `Proposed` method manages RAM better compared to `ILP` while still maintaining high efficiency. Proper memory utilization ensures that more requests can be processed efficiently.



(a) CPU utilization



(b) RAM utilization

Fig. 4: Percentage of CPU and RAM utilization

4) Computation Time: Figure 5 shows the computation time required for different scheduling algorithms as the number of requests increases. The `ILP` method takes the most time, likely due to its optimization complexity. The `Proposed` method achieves lower computation time than `ILP`, indicating better efficiency. `U-WoAVL` and `U-Random` methods have significantly lower computation times but may lack optimal resource allocation. The Proposed method balances efficiency and computational cost.

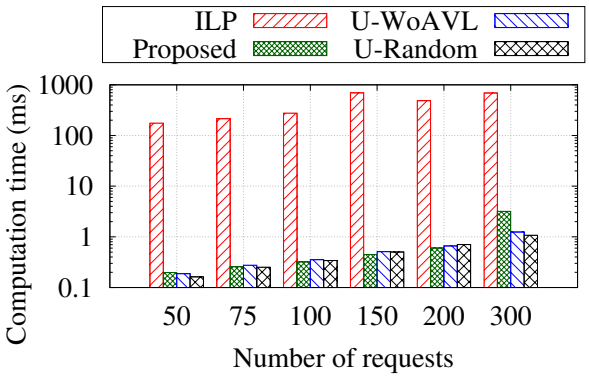


Fig. 5: Computation time

VII. Conclusion

This paper introduces a comprehensive framework for Age of Information (AoI) and availability-aware service provisioning and inference in edge-enabled Digital Twin (DT) systems. Unlike conventional approaches that focus solely on minimizing AoI, the proposed model incorporates additional critical factors such as quality-of-services, node availability, and resource constraints. This holistic design enables a more realistic and effective solution for deploying DT applications in heterogeneous and dynamic edge environments. The framework employs a two-phase optimization strategy: First, it selects a minimal set of master nodes using hop-based heuristics to ensure efficient coverage and centralized coordination; Second, it utilizes a utility-aware greedy algorithm to allocate service requests by maximizing the utility-to-cost ratio. This approach takes into account various delay components—computation, transmission, propagation, and queuing—offering an accurate representation of system latency. Experimental results confirm the superiority of the proposed method across several performance metrics. It consistently achieves higher total utility, near-optimal solutions, and demonstrates better request admission rates, reflecting enhanced system throughput. Moreover, it shows improved CPU and RAM utilization, highlighting effective resource management, while maintaining lower computation time, underscoring its practical viability and scalability in real-world applications. The future extension of this work includes the use of deep learning approaches for service placement in digital twin.

References

[1] Y. Zhang, W. Liang, Z. Xu, W. Xu, and M. Chen, "AoI-Aware Inference Services in Edge Computing via Digital Twin Network Slicing," *IEEE Transactions on Services Computing*, vol. 17, no. 6, pp. 3154–3170, Nov. 2024.

[2] L. Wei, H. Zhang, Y. Zhang, W. Sun, and Y. Zhang, "Energy-Efficient Digital Twin Placement in Mobile Edge Computing," in *ICC 2023 - IEEE International Conference on Communications*, May 2023, pp. 2480–2485.

[3] V. Farhadi, F. Mehmeti, T. He, T. L. Porta, H. Khamfroush, S. Wang, and K. S. Chan, "Service Placement and Request Scheduling for Data-intensive Applications in Edge Clouds," in *Proc. of IEEE INFOCOM*, Apr. 2019, pp. 1279–1287.

[4] Y. Lu, S. Maharjan, and Y. Zhang, "Adaptive Edge Association for Wireless Digital Twin Networks in 6G," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16 219–16 230, Nov. 2021.

[5] Y. Dai, J. Zhao, J. Zhang, Y. Zhang, and T. Jiang, "Federated Deep Reinforcement Learning for Task Offloading in Digital Twin Edge Networks," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 3, pp. 2849–2863, May 2024.

[6] J. Li, S. Guo, W. Liang, J. Wang, Q. Chen, Z. Hong, Z. Xu, W. Xu, and B. Xiao, "AoI-Aware Service Provisioning in Edge Computing for Digital Twin Network Slicing Requests," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 14 607–14 621, Dec. 2024.

[7] Y. Wang, J. Fang, Y. Cheng, H. She, Y. Guo, and G. Zheng, "Cooperative End-Edge-Cloud Computing and Resource Allocation for Digital Twin Enabled 6G Industrial IoT," *IEEE Journal of Selected Topics in Signal Processing*, vol. 18, no. 1, pp. 124–137, Jan. 2024.

[8] Y. Lu, X. Huang, K. Zhang, S. Maharjan, and Y. Zhang, "Communication-Efficient Federated Learning and Permissioned Blockchain for Digital Twin Edge Networks," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2276–2288, Feb. 2021.

[9] I. Turcanu, G. Castignani, and S. Faye, "On the Integration of Digital Twin Networks into City Digital Twins: Benefits and Challenges," in *Prof. of the IEEE CCNC*, Jan. 2024, pp. 752–758.

[10] S. Bera, "Availability-Aware VNF Placement for uRLLC Applications in MEC-Enabled 5G Networks," in *2023 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, Dec. 2023, pp. 300–305.

[11] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. ACM NIPS*, Dec. 2012, pp. 2951–2959.

[12] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv:1012.2599 [cs]*, Dec. 2010.

[13] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv:1807.05118 [cs, stat]*, Jul. 2018.

[14] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.