Digital Twin Placement for Minimum Application Request Delay with Data Age Targets

Mehrad Vaezi*, Kiana Noroozi*, Terence D. Todd*, Dongmei Zhao* and George Karakostas†
*Department of Electrical and Computer Engineering
†Department of Computing and Software
McMaster University
Hamilton, Ontario, CANADA
Email: {vaezim,noroozik,todd,dzhao,karakos}@mcmaster.ca

***Abstract*—Digital twins (DTs) are virtual implementations of physical systems (PSs) and can represent the states of the PSs in realtime. In order to update the DTs with changes in their corresponding PSs, the PSs should regularly send their state information data to the DTs. Each DT must be assigned to an execution server (ES) that processes the forwarded data from its corresponding PS. The output is then made available to applications that are operating at an internet cloud server. In this paper we consider the problem of DT placement such that the maximum data request-response delay experienced by the application over all PSs is minimized, subject to maximum data age target constraints at the DTs and the application server. The problem is first formulated as an integer quadratic program (IQP) and then transformed into a semidefinite program (SDP). The problem is NP-complete. Since exact polynomial solutions are unavailable, several practical polynomial-time approximation algorithms are introduced. The algorithms are designed to give solutions with different trade-offs between the accommodation of the application input timing latency and the achievement of data age targets.**

***Index Terms*—Digital twin, placement, data age target, minimum delay.**

I. Introduction

A digital twin (DT) is a software-based implementation of a real physical system (PS) and can be used to represent the PS and evolve with the later throughout its lifetime [1], [2]. The usefulness of DTs has been demonstrated in many scenarios, especially in IoT [3], [4] and industrial application areas [5]. A common aspect of DTs is their need to regularly update information according to the changes in their associated PSs [6]. This is referred to as DT *synchronization*.

DTs are especially suitable for IoT applications since they are tailored to accommodate the real-time flow and processing of data. For example, [7] introduces a dynamic digital twin of vehicles and road-side units (RSUs) to capture time-varying resource demand information. In [8], a DT takes the dynamic network topology of a vehicular network into account to capture social features and map them into a virtual space. The work in [9] uses the digital twin of a warehouse to obtain real-time data and visual feedback on cargo information.

In the field of cloud computing, [10] proposes a DT-based network architecture that replaces the traditional end-to-end communication model with a cloud-to-end communication version, where a communication end (e.g., a person or a device) can be replaced by its digital representation. In [11], [12], digital twins are integrated with a conventional edge network to build a digital twin edge network, where the DTs can exchange information collected from their PSs, process the information by taking advantage of the computation resource of the edge servers, and generate results needed to improve performance of the PSs. In conventional edge computing applications, synchronizing real-time data between users and edge servers requires significant wireless resources. By mapping mobile devices to DTs in edge servers, the DTs can assist in both latency reduction and reliability enhancement. Reference [13] proposed a DT-based approach that optimizes the performance of a mobile network based on its current state and uses the DT to predict future states and behavior of the network.

As a digital representation of its PS, a DT can interact with applications on behalf of its PS. This helps decompose the communications between real systems and the applications into the synchronization process between the PSs and their DTs and the information delivery process between the DTs and the applications. Since the two processes can run in parallel, this brings more flexibility to network resource allocations, while maintaining low age of information at the applications. In this scenario, the placement of a DT at a given execution server (ES) affects both the synchronization delay between the PS and the DT and the communication delay between the DT and the application.

DTs typically offer functionality in addition to simple data relaying, e.g., they may compress, process and optimize data received from their PSs, before they make them available to the application. They may also provide information of the PS based on historical information associated with the PS, which would not typically be provided by the PS itself [14] [15]. Therefore, the problem of DT placement should consider both the computation resources needed for hosting the DTs and the continuous data updates between the PSs and the DTs. This makes the DT placement problem different from classic proxy server placement problems, e.g., [16].

In this paper we consider the problem of DT placement when there are multiple PSs communicating with applications through their DTs. There are data age targets for both the applications and the DTs, which capture the real-time nature of the system and the need for DT synchronization. Each

DT must be assigned to one of the available candidate ES locations. When a DT is placed, the PS periodically updates it so that its state accurately tracks that of its PS. The DT placement problem is defined so that the maximum data request response delay experienced by the application over all PSs is minimized, subject to a maximum *data age target* at the DT, i.e., the (given) data age target is an upper bound of the Age-of-Information or "freshness" of the data available at the DT [17]. The main contributions of this paper are summarized as follows:

- A DT to Execution Server (ES) placement problem is formulated to minimize the maximum application interaction delay when accessing multiple PSs, while ensuring the timely delivery of PSs' data to their DTs and then to the application.
- The placement problem is an integer quadratic program (IQP), whose fractional relaxation can be further strengthened into a semidefinite program (SDP). The problem is shown to be NP-complete. Since an exact polynomial-time solution is not available, polynomial-time approximation algorithms are introduced to obtain DT placements, which may not be feasible for the original problem. However, the algorithms are designed to offer practical solutions that give different performance tradeoffs between application server response times and the satisfaction of PS data age targets.
- The fractional solution of the relaxed SDP is rounded to obtain the final (integral) assignments of DTs to ESs. Different rounding alternatives, i.e., Random Selection, X-Congestion, Z-Congestion, Constraint Slack SDP, and Constraint Slack QP, are proposed.
- Simulation results are provided that show Z-congestion (and, to a lesser degree, Constraint Slack SDP) achieves the best application interaction delay, while coming closer to (given) data age targets than the other alternatives.

The rest of the paper is organized as follows. Section II summarizes the recent work related to DT placement. Section III defines the system model. The optimum DT placement problem is formulated in Section IV, where the original IQP is translated into an integer linear programming (ILP) and further relaxed into an SDP. Section V presents our proposed approximation algorithms. Simulation results are shown in Section VI to demonstrate the performance of the proposed algorithms. The paper is concluded in Section VII.

II. Related Work

In this section we summarize some work related to DT placement. A more complete list of related work can be found in [18].

An algorithm is proposed in [19] for service entity placement in Virtual Reality (VR) applications. The proposed method aims at minimizing the costs of placing a service entity on an edge server. Different from DT placement in this work, the service entity placement in [19] does not contain constraints on the data update rate by the corresponding PS.

Deep-learning-based algorithms are proposed in [20] for digital twin placement and migration. Two types of DTs are considered, device DTs and service DTs. The former is a full replica of a physical device, and the latter is a lightweight DT that extracts information directly related to a specific application from multiple devices. The placement of these DTs to the servers is optimized with the objective of minimizing the average system delay.

The problem of DT placement for IoT devices in edge IoT networks is studied in [21]. The objective is to minimize the total communication latencies between the IoT devices and their corresponding DTs. One assumption behind this formulation is sufficient server capacity so that the data processing time at the DTs can be neglected.

Different from the above work, our paper addresses the DT placement issue by considering the age-of-information at both the DTs and the application. We employ a time-sharing resource allocation model for computational resources at the servers, although this results in an integer quadratic problem that is proven to be NP-complete. Our objective is to minimize the maximum delay experienced by an application that is accessing information from multiple DTs. There is prior work such as [19], [22] where digital twins are optimized to serve particular applications. However, to the best of our knowledge, ours is the first paper that places a maximum data age target constraint on the data provided by the DTs to the application.

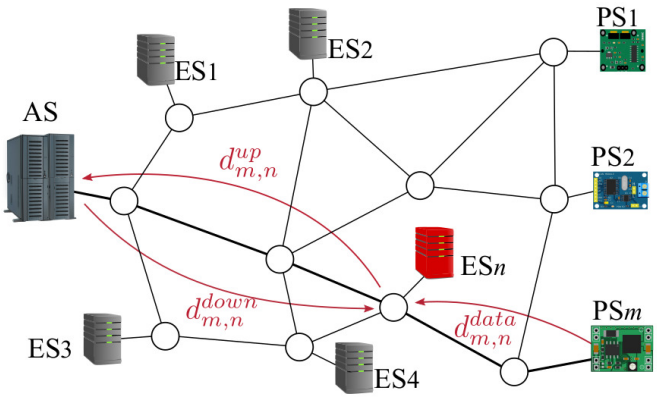


Fig. 1. Digital Twin Placement Model

III. System Model

Figure 1 shows an application server (AS) that requires input from multiple PSs that are connected to the same network. Each PS has an associated DT that it regularly communicates with so that the latter can interact with the application on its behalf. Each DT is hosted at one of a set of execution servers, ESs, which are also shown in the figure. A practical example of this is where the PSs include sensors located in a manufacturing plant and provide their inputs at regular intervals so that the application can monitor the performance of the system [14] [15].

The main objective of this paper is to determine the placement of the DTs at the ESs so that the maximum communication delay experienced by the application is minimized, and at the same time, the data delivered best adheres to a desired data age target. When placed at an ES, a DT incurs both communication and computational delays as its PS data

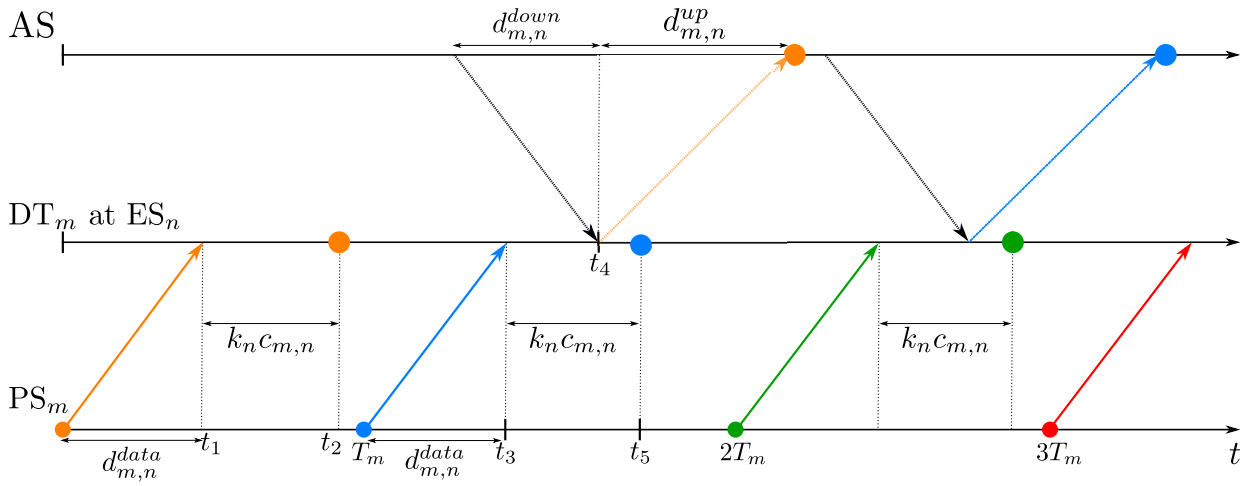


Fig. 2. DT Update and AS Request Timing

is periodically refreshed. Let $\mathcal{M} = \{1, 2, ..., M\}$ be a set of M PSs and $\mathcal{N} = \{1, 2, ...N\}$ be a set of N ESs. Each PS has an associated DT, i.e., DT_m for PS_m that must be placed at one of the ESs, e.g., ES_n for $n \in \mathcal{N}$.

Figure 2 shows a timeline example of PS_m, DT_m (at ES_n), and the AS. Each PS periodically updates its DT with new data as shown in the bottom two timelines. At $t = 0$, the first update (shown by the purple dot) is transferred from PS_m to DT_m (at ES_n) in time $d^{data}_{m,n}$ and is received by the ES server at time t_1. At DT_m, this update requires $c_{m,n}$ seconds of computation from ES_n before the update is completed. If k DTs have been assigned to ES_n then, as in the colocation constraints in [19], the processing will be completed after a delay no longer than $k \cdot c_{m,n}$ seconds, at time t_2 in the figure. This updating procedure recurs periodically for each PS, i.e., for PS_m it repeats every T_m seconds. At $t = T_m$ and $t = 2T_m$, the second and third PS_m updates (at the blue and green dots, respectively) are shown in Figure 2.

The top timeline (AS) in Figure 2 shows the application generating a request for input from DT_m. This request arrives at the DT $d^{down}_{m,n}$ seconds later, and the requested data are returned to the AS after a further delay of $d^{up}_{m,n}$ seconds. These time durations are also shown by the arrows in Figure 1. A DT always responds to application requests immediately with the most recently updated version of the requested data. In Figure 2, the first application request arrives at the DT at $t = t_4$, and therefore, the DT passes the results based on processing the first update to the application server since the second update has not completed.

One of the data age requirements is that the PS_m data available at DT_m must be synchronized every T_m seconds, which implies that the time needed for m's data to be transmitted to and processed by DT_m should not be more than T_m seconds. That is,

$$d^{data}_{m,n} + k_n \cdot c_{m,n} \leq T_m, \tag{1}$$

where k_n is the total number of DTs hosted by ES_n.

The application may query multiple DTs in order to obtain the information that it needs. It has its own data age target, denoted by A^*, which is the desired maximum age target over all the queried PSs, i.e., if the target is satisfied, then when data arrives from all the queried DTs, the time since the data was generated at all the PSs will be at most A^* seconds. We also assume that there are known upper bounds on data transfer latencies between ES_n and the application server and each PS [23]. These bounds can be guaranteed, e.g., by resource allocation via contractual terms with the network provider, and by power control when the PS has a wireless connection to the network [24] [25].

Figure 2 shows the worst case for the age of the PS_m data delivered to the application when the data request from the AS arrives just before the next PS_m-to-DT_m data update cycle has completed, i.e., $t_4 < t_5$ and $t_5 - t_4 \approx 0$. This coincides with the time $T_m + d^{data}_{m,n} + k_n \cdot c_{m,n}$ in the figure, when the processing of data generated at time T_m (the blue one) is almost (but not quite) ready at DT_m. As a result, the application receives the data generated at time $t = 0$ (the purple one in the figure) $T_m + d^{data}_{m,n} + k_n \cdot c_{m,n} + d^{up}_{m,n}$ seconds later. According to the discussion above, this worst-case time cannot be larger than A^* if the application data age target is to be achieved, i.e.,

$$T_m + d^{data}_{m,n} + k_n \cdot c_{m,n} + d^{up}_{m,n} \leq A^*. \tag{2}$$

As in [23], we assume that network latencies are known or can be estimated. Since the loading imposed on the network is assumed to be small, we also assume that they are independent of the DT placement, i.e., the aggregate loading of the PSs communicating with the ESs is assumed to be far less than the actual capacity of the underlying links connecting the ESs, as in [19].

Note that for a given set of upper bounds in (1) and (2), there may not exist a feasible placement of DTs to ESs, i.e., at least one of these constraints will be violated no matter where the DTs are placed. We show below that detecting the infeasibility of the given input (and, therefore, calculating an optimal placement, if such a placement exists) is NP-complete (see Theorem 4.1). Therefore, our proposed algorithms may violate a number of the constraints.The algorithms, however,

work towards incurring the smallest possible violation of (1) and (2), and, therefore, they work even if the original input is infeasible. Nevertheless, in order to study their performance, we compare them only on feasible instances in Section VI, since we do not want to combine their own potential inefficiencies with inefficiencies inherent in the input itself.

IV. Problem Formulation

We define binary decision variables $X_{m,n} \in \{0,1\}$ for $m = 1,2,\ldots,M$ and $n = 1,2,\ldots,N$ with $X_{m,n} = 1$ if DT_m is placed on ES n and $X_{m,n} = 0$ otherwise. Then conditions (1) and (2) become

$$d_{m,n}^{data} + c_{m,n} X_{m,n} \sum_{k=1}^{M} X_{k,n} \leq T_m, \tag{3}$$

and

$$X_{m,n}\left(T_m + d_{m,n}^{data} + c_{m,n}\sum_{k=1}^{M} X_{k,n} + d_{m,n}^{up}\right) \leq A^*, \tag{4}$$

respectively.

By addressing the problem of placing the DTs of M PSs on N ESs, our objective is that the maximum data request response delay experienced by the application over all PSs is minimized, while its data age requirement, and the refreshing rate requirement of all DTs are satisfied. This is formulated as the following Integer Quadratic Program (IQP):

$$\min_{X,\tau}\ \tau \text{ s.t.} \tag{IQP}$$
$$\sum_{n=1}^{N}(d_{m,n}^{down} + d_{m,n}^{up})X_{m,n} \leq \tau,\ \forall m \in \mathcal{M} \tag{5}$$
$$d_{m,n}^{data} + c_{m,n}X_{m,n}\sum_{k=1}^{M} X_{k,n} \leq T_m,\ \forall m \in \mathcal{M}, n \in \mathcal{N} \tag{6}$$
$$X_{m,n}(T_m + c_{m,n}\sum_{k=1}^{M} X_{k,n} + d_{m,n}^{data} + d_{m,n}^{up}) \leq A^*, \forall m,n \tag{7}$$
$$\sum_{n=1}^{N} X_{m,n} = 1,\ \forall m \in \mathcal{M} \tag{8}$$
$$X_{m,n} \in \{0,1\},\ \forall m \in \mathcal{M}, n \in \mathcal{N} \tag{9}$$
$$\tau \geq 0, \tag{10}$$

where the LHS of constraint (5) is the delay experienced by the application when requesting the PS m data, τ is the maximum of such delays over all m, and we seek to minimize τ. Constraint (8) ensures that a DT is assigned to one ES only, for all M DTs.

Constraint (6) can be rewritten as

$$X_{m,n}\sum_{k=1}^{M} X_{k,n} \leq \frac{T_m - d_{m,n}^{data}}{c_{m,n}}, \tag{11}$$

and constraint (7) as

$$X_{m,n}\sum_{k=1}^{M} X_{k,n} \leq \frac{A^* - X_{m,n}(T_m + d_{m,n}^{data} + d_{m,n}^{up})}{c_{m,n}}, \tag{12}$$

for all m,n. Now, both constraint (11) and (12) share the same LHS and can be merged into one constraint

$$X_{m,n}\sum_{k=1}^{M} X_{k,n} \leq u_{m,n} \tag{13}$$

where

$$u_{m,n} = \left\lfloor \min\left\{ \frac{T_m - d_{m,n}^{data}}{c_{m,n}}, \frac{A^* - (T_m + d_{m,n}^{data} + d_{m,n}^{up})}{c_{m,n}} \right\} \right\rfloor \tag{14}$$

Hence, values for $\mathbf{X}$ satisfy (6), (7) iff these values satisfy (13). The new formulation can be written as:

$$\min_{X,\tau}\ \tau \text{ s.t.} \tag{IQP'}$$
$$\sum_{n=1}^{N}(d_{m,n}^{down} + d_{m,n}^{up})X_{m,n} \leq \tau,\ \forall m \in \mathcal{M} \tag{15}$$
$$X_{m,n}\sum_{k=1}^{M} X_{k,n} \leq u_{m,n},\ \forall m \in \mathcal{M}, n \in \mathcal{N} \tag{16}$$
$$\sum_{n=1}^{N} X_{m,n} = 1,\ \forall m \in \mathcal{M} \tag{17}$$
$$X_{m,n} \in \{0,1\},\ \forall m \in \mathcal{M}, n \in \mathcal{N} \tag{18}$$
$$\tau \geq 0 \tag{19}$$

For a specific τ, all variables $X_{m,n}$ in (15) with coefficients $d_{m,n}^{down} + d_{m,n}^{up} > \tau$ are forced to be 0 because of (17). Therefore, we can perform a binary search in range $[0, \max_{m,n}\{d_{m,n}^{down} + d_{m,n}^{up}\}]$ for the minimum feasible τ. Every value of τ, in effect, defines a bipartite graph $G = (A, B, E)$ with the nodes in A corresponding to DTs, the nodes in B corresponding to ESs, and edges $(m,n) \in E$ only when $d_{m,n}^{down} + d_{m,n}^{up} \leq \tau$. Therefore, for any value of τ, we can simplify problem (IQP') to the question of feasibility of constraints (16)-(18) on a bipartite graph (where $X_{m,n} := 0$ whenever $(m,n) \notin E$). We adopt this approach when we study the τ value and the violation of feasibility of (16) in Section VI. Problem (16)-(18) on a bipartite graph is an NP-complete problem, as shown by the following theorem.

Theorem 4.1: Deciding the feasibility of (16)-(18) on a bipartite graph is an NP-complete problem.

Proof: The problem is clearly in NP, since, given a $\{0,1\}$-assignment for variables $X_{m,n}$, checking the feasibility of constraints (16), (17) can be done in polynomial time.

We reduce SATISFIABILITY (i.e., given a CNF formula, asking whether there is a satisfying truth assignment for its variables) to our problem. Given a CNF formula with n variables and m clauses, we construct a bipartite graph as follows: The left-side set of nodes A consists of $n+m$ nodes, i.e., one node for each variable or clause. On the right-side B there are $2n$ nodes, i.e., a pair of nodes for every pair of literals $x_i, \bar{x_i}$ corresponding to the i-th variable. For every clause l, there is an edge between $l \in A$ and the node of every literal used by l in B. For example, for clause $l = (x_2 \vee \bar{x_5} \vee x_8)$ there are edges (l, x_2), $(l, \bar{x_5})$, (l, x_8). For each such edge (m,n) we set $u_{m,n} := \infty$. Also, for the i-th variable node in A, we add edges $(i, x_i), (i, \bar{x_i})$, with $u_{i,x_i} = u_{i,\bar{x_i}} := 1$. By construction, if $X_{i,x_i} = 1$, then $X_{l,x_i} = 0$ for any clause l that uses literal x_i, due to (13) for $m = i, n = x_i$, i.e., only variables $X_{k,\bar{x_i}}$ will be allowed to take value 1, for clauses k that use literal $\bar{x_i}$; the case $X_{i,\bar{x_i}} = 1$ is symmetric.

Now it is easy to see that the given CNF formula is satisfiable iff there is a $\{0,1\}$-assignment to variables $\mathbf{X}$ that also satisfies (16)-(18). If the formula is satisfiable, then set $X_{i,x_i}=1, X_{i,\bar{x}_i}=0$ if $x_i=0$, or $X_{i,x_i}=0, X_{i,\bar{x}_i}=1$ if $x_i=1$. Also, each clause l must contain a literal x_i or $\bar{x}_i$ that is set to 1, and so we can set $X_{l,x_i}=1$ or $X_{l,\bar{x}_i}=1$, without violating any of the constraints. We set all other variables $X_{m,n}:=0$. It is easy to verify that this assignment satisfies all constraints (16)-(18). Conversely, if there is a value assignment to variables $X_{m,n}$ that satisfies (16)-(18), then this assignment forces $X_{i,x_i}=1, X_{i,\bar{x}_i}=0$ or $X_{i,x_i}=0, X_{i,\bar{x}_i}=1$ for each i-th variable; we translate this assignment to $x_i=0$ or $x_i=1$, respectively. The assignment of *each* clause node l to exactly one node of its literal(s) in the bipartite graph, is consistent with our truth assignment, and satisfies each clause. ■

As a result of Theorem 4.1, we do not expect that there is a polynomial-time algorithm that exactly solves (IQP').

We can linearize problem (IQP') as follows. We use a binary variable $Z^n_{k,m}\in\{0,1\}$ to replace the product $X_{k,n}X_{m,n}$ and add the following valid constraints:

$$\begin{aligned} Z^n_{k,m} &= Z^n_{m,k}, && \forall k,m\in\mathcal{M}, n\in\mathcal{N}\\ Z^n_{m,m} &= X_{m,n}, && \forall m\in\mathcal{M}, n\in\mathcal{N}\\ Z^n_{k,m} &\le X_{m,n}, && \forall k,m\in\mathcal{M}, n\in\mathcal{N}\\ X_{m,n}+X_{k,n}-Z^n_{k,m} &\le 1, && \forall k,m\in\mathcal{M}, n\in\mathcal{N} \end{aligned}$$

The new problem formulation is an integer linear programming (ILP) given as follows:

$$\min_{X,Z,\tau}\ \tau \text{ s.t.} \tag{ILP}$$
$$\textstyle\sum_n (d^{down}_{m,n}+d^{up}_{m,n})X_{m,n}\le\tau,\ \forall m\in\mathcal{M} \tag{20}$$
$$\textstyle\sum_k Z^n_{k,m}\le u_{m,n},\quad \forall m\in\mathcal{M}, n\in\mathcal{N} \tag{21}$$
$$\textstyle\sum_n X_{m,n}=1,\ \forall m\in\mathcal{M} \tag{22}$$
$$Z^n_{k,m}=Z^n_{m,k},\ \forall k,m\in\mathcal{M}, n\in\mathcal{N} \tag{23}$$
$$Z^n_{m,m}=X_{m,n},\ \forall m\in\mathcal{M}, n\in\mathcal{N} \tag{24}$$
$$Z^n_{k,m}\le X_{m,n},\ \forall k,m\in\mathcal{M}, n\in\mathcal{N} \tag{25}$$
$$X_{m,n}+X_{k,n}-Z^n_{k,m}\le 1,\ \forall k,m\in\mathcal{M}, n\in\mathcal{N} \tag{26}$$
$$X_{m,n}, Z^n_{k,m}\in\{0,1\},\ \forall k,m\in\mathcal{M}, n\in\mathcal{N} \tag{27}$$
$$\tau\ge 0 \tag{28}$$

With these added constraints, any integral feasible solution sets $Z^n_{k,m}=X_{k,n}X_{m,n},\ \forall k,m,n$. Note that for (24), $Z^n_{m,m}=X^2_{m,n}=X_{m,n}$.

We can strengthen problem (ILP) by observing that when $X_{m,n}$'s have integral values, the matrices $\mathbf{Z}^n$ are positive semi-definite (PSD) for all n, since $\mathbf{Z}^n=\mathbf{X}^n\mathbf{X}^{nT}$, where $\mathbf{X}^n$ is the n-th column of matrix $\mathbf{X}=[X_{m,n}]$. Therefore we can add the following constraint into problem (ILP)

$$\mathbf{Z}^n\in\mathcal{PSD},\ \forall n\in\mathcal{N}. \tag{29}$$

If we relax (27) to $X_{m,n}, Z^n_{m,k}\ge 0,\ \forall k,m\in\mathcal{M}, n\in\mathcal{N}$, (ILP) becomes the following problem

$$\min_{X,Z,\tau}\ \tau \text{ s.t.} \tag{SDP-relaxed}$$
$$(20)-(26),(28),(29)$$
$$X_{mn}, Z^n_{k,m}\ge 0,\quad \forall k,m\in\mathcal{M}, n\in\mathcal{N}. \tag{30}$$

This relaxed problem will be referred to as the *SDP-relaxed problem*. It is a convex SDP program and can be solved in polynomial time by any SDP solver. Note that constraints $X_{m,n}\le 1$ and $Z^n_{m,k}\le 1$ are implied by (22), (25).

As noted in Section III, it may be the case that for a given input, (SDP-relaxed) is infeasible. This implies that the integer problem (IQP'), as well as the original problem (IQP) are also infeasible. In the case when (IQP') is infeasible, but (SDP-relaxed) is feasible, our algorithms are not affected, since they round the fractional solution of (SDP-relaxed). If (SDP-relaxed) is infeasible, (21) is modified by "inflating" the upper bounds with $\lambda u_{m,n}$, where $\lambda>1$. Let $\lambda_{\max}=M/(\min_{\forall\ m,n}u_{m,n})$. Setting $\lambda=\lambda_{\max}$ allows the modified (21) to be always satisfied. By running a binary search on the interval $[1,\lambda_{\max}]$, the smallest λ can be found such that (SDP-relaxed) becomes feasible. Our algorithms will work with the fractional solution corresponding to these new (inflated) upper bounds. Therefore, for the rest of the paper, we will assume that (SDP-relaxed) is feasible.

V. Approximation Algorithms

The problem formulation of Section IV treats constraint (13) as a *hard* constraint that would be satisfied if an optimal solution to (IQP') was available. For practical systems this is not possible since even finding a feasible (not necessarily optimal) integral solution to problem (IQP') is NP-complete (Theorem 4.1). Therefore, in this section we introduce polynomial-time approximation algorithms for solving the DT placement problem. The algorithms we propose will return solutions that are approximate in terms of the objective τ, and also will violate the merged DT refresh and application data age constraints (21).

In Section VI we include comparisons of the proposed algorithms in cases where there is an integral optimum, i.e., the original problem instance is feasible for (5)-(10).

Rounding algorithm: We start by describing how to round a fractional solution of the (SDP-relaxed) problem to an integral assignment of DTs to ESs. This rounding subroutine will be the main component of the algorithms we propose for solving the problem or detect its infeasibility (Algorithms 2 & 3).

After obtaining the fractional solution $\mathbf{X},\mathbf{Z}$ of the (SDP-relaxed) problem, Algorithm 1 is used to round it to an integral one. In lines 2-6, all the integral $X_{m,n}$'s from the solution to the (SDP-relaxed) problem are fixed, i.e., DT m is assigned to ES n if $X_{m,n}=1$, and it will never be assigned so if $X_{m,n}=0$. After the for-loop, set $\mathcal{SM}$ contains the DTs that are still fractionally assigned to ESs.

While $\mathcal{SM}\neq\emptyset$, the algorithm picks a DT m from $\mathcal{SM}$ and a pair of ESs n_1,n_2 with $0<X_{m,n_1},X_{m,n_2}<1$ (line 8). Different selection criteria give different rounding algorithms, and will be described in detail below.

We set $X_{m,n_1} := X_{m,n_1} + X_{m,n_2}, X_{m,n_2} := 0$, resulting in at least one more rounded variable of $\mathbf{X}$ (lines 9-13). Note that this update may violate the constraints. Therefore, the algorithm goes through a series of updates to make the problem to be feasible again. This includes increasing T_m's and A^* to satisfy constraint (21) (lines 16-27) and adjusting the $Z^n_{k,m}$ values based on the updated $X_{m,n}$'s to satisfy constraints (25) and (26) (lines 29-36). Note that fixing constraints (26) (line 34) after constraints (25) (line 31) ensures that the latter will still be satisfied.

Line 15 records the original $u_{m,n}$ as $u^*_{m,n}$ and line 26 records the updated $u_{m,n}$. Let

$$\Delta u = \max_{m,n} \frac{u_{m,n} - u^*_{m,n}}{u^*_{m,n}} \tag{31}$$

be the maximum violation of $u_{m,n}$ due to the rounding. The algorithm outputs the rounded X, Z, and the Δu.

As already mentioned, the choice of m, n_1, n_2 in Algorithm 1 (line 8) gives rise to different rounding algorithms described below.

- **Random Selection:** Pick uniformly at random a DT $m \in \mathcal{SM}$ and two ESs $n_1, n_2 \in \mathcal{N}$ such that $0 < X_{m,n_1}, X_{m,n_2} < 1$.
- **X-Congestion:** $\sum_m X_{m,n}$ is used as a measure of the *congestion* of ES n. Let $n_2 = \arg\max_n \sum_{m=1}^{M} X_{m,n}$, i.e., n_2 is the most congested ES, and $m = \arg\max_k X_{k,n_2}$. Set $n_1 = \arg\min_n X_{m,n}$ (we break ties arbitrarily).
- **Z-Congestion:** $\sum_{k,m} Z^n_{k,m}$ is used as a measure of the congestion of ES n. Let $n_2 = \arg\max_n \sum_{k,m} Z^n_{k,m}$. Then m, n_1 are picked as in **X-Congestion**.
- **Constraint Slack SDP:** Let $m, n_2 = \arg\min_{m,n}\{u_{m,n}\}$, i.e., (m, n_2) is the DT-ES pair for which the slack of constraint (21) is minimum. Note that if any of these constraints is tight, the minimum slack is 0. Set $n_1 = \arg\max_n u_{m,n}$, i.e., n_1 is the ES that has the maximum slack among all the constraints (21) for m. The intuition behind this selection of m, n_1, and n_2 is that the algorithm is trying to take away assignment "weight" from tight (or near tight) constraints, and assign it to constraints with lots of slack.
- **Constraint Slack QP:** Same as **Constraint Slack SDP**, only now we consider the slack of constraints (13).

Note that every iteration of the main loop of Algorithm 1 (lines 7-37) rounds at least one of the $O(MN)$ variables, and each iteration takes $O(MN)$ time for an overall running time of $O(M^2N^2)$.

Approximation algorithms: We use Algorithm 1 as a subroutine to develop algorithms that work towards restricting deviation from fractional u or τ (Algorithms 2 and 3, respectively).

Algorithm 1 Rounding Algorithm

Input: Fractional solution $\mathbf{X}, \mathbf{Z} \geq \mathbf{0}$ of (SDP-relaxed)

1: $\mathcal{SM} := \mathcal{M}$
2: **for all** $\mathtt{m} \in \mathcal{SM}$ **do**
3: **if** $X_{m,n} \in \{0, 1\}$ for some n **then**
4: $\mathcal{SM} := \mathcal{SM} \setminus \{m\}$
5: **end if**
6: **end for**
7: **while** $\mathcal{SM} \neq \emptyset$ **do**
8: $\mathbf{Select}(m, n_1, n_2)$ {edge-pair selection algorithm}
9: $X_{m,n_1} := X_{m,n_1} + X_{m,n_2}$
10: $X_{m,n_2} := 0$
11: **if** $X_{m,n_1} = 1$ **then**
12: $\mathcal{SM} := \mathcal{SM} \setminus \{m\}$
13: **end if**
14: **for all** $m \in \mathcal{M}, n \in \mathcal{N}$ **do**
15: $u^*_{m,n} := u_{m,n}$
16: **if** constraint (21) is violated **then**
17: $u_A := \frac{A^* - (T_m + d^{data}_{m,n} + d^{up}_{m,n})}{c_{m,n}}$
18: $u_T := \frac{T_m - d^{data}_{m,n}}{c_{m,n}}$
19: **if** $u_T < LHS < u_A$ **then**
20: Increase T until $u_T = LHS$
21: **else if** $u_A < LHS < u_T$ **then**
22: Increase A^* until $u_A = LHS$
23: **else if** $u_T, u_A < LHS$ **then**
24: Increase T and A^* until $u_A = u_T = LHS$
25: **end if**
26: $u_{m,n} = \min\{u_A, u_T\}$
27: **end if**.
28: **end for**
29: **for all** $k, m \in \mathcal{M}, n \in \mathcal{N}$ **do**
30: **if** constraint (25) is violated **then**
31: $Z^n_{k,m}, Z^n_{m,k} := X_{m,n}$
32: **end if**
33: **if** constraint (26) is violated **then**
34: $Z^n_{k,m}, Z^n_{m,k} := X_{m,n} + X_{k,n} - 1$
35: **end if**
36: **end for**
37: **end while**
38: **Output:** $\mathbf{X}$, $\mathbf{Z}$, and Δu

Algorithm 2 Finding solution with sub-ϵ_u violation

Input: $D_{sorted} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_{M,N}\}, \epsilon_u$

1: $\hat{\tau}_f$ = min in D_{sorted} s.t. (SDP-relaxed) is feasible (binary search in D_{sorted})
2: $\hat{\tau}_s$ = min in $\{\hat{\tau}_f, \hat{\tau}_{f+1}, \ldots, \hat{\tau}_{M,N}\}$ s.t. {Binary search}
- X_f, Z_f = solution of (SDP-relaxed) problem with $\tau = \hat{\tau}_s$
- $X, Z, \Delta u$ = Algorithm 1(X_f, Z_f)
- $\Delta u \leq \epsilon_u$

3: **if** no $\hat{\tau}_s$ **then**
4: return INFEASIBLE
5: **else**
6: return $\mathbf{X}$
7: **end if**

Algorithm 3 Finding solution with sub-ϵ_τ violation

Input: $D_{sorted} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_{M,N}\}, \epsilon_\tau$

τ_{opt}^{SP} = fractional optimum of (SDP-relaxed)
Find max s such that $\hat{\tau}_s \leq \tau_{opt}^{SP}(1+\epsilon_\tau)$
X_f, Z_f = solution of (SDP-relaxed) problem with $\tau = \hat{\tau}_s$
$X, Z, \Delta u$ = Algorithm 1(X_f, Z_f)
if no $\hat{\tau}_s$ **then**
    return INFEASIBLE
else
    return **X**
end if

Let τ_{opt}^{QP} be the optimum τ of (IQP'). We observe that $\tau_{opt}^{QP} \in \{d_{m,n}^{up} + d_{m,n}^{down} : m \in \mathcal{M}, n \in \mathcal{N}\}$. Each of these MN possible values for τ_{opt}^{QP} corresponds to a restriction of the set of possible assignments of DTs to ESs. More specifically, define $D_{sorted} = \{\hat{\tau}_1, \hat{\tau}_2, \ldots, \hat{\tau}_{M,N}\}$ to be the sorted list of values $\{d_{m,n}^{up} + d_{m,n}^{down} : m \in \mathcal{M}, n \in \mathcal{N}\}$ in ascending order (note that if there are repetitions of values for different m, n combinations, then $|D_{sorted}| < MN$, but for clarity of the presentation we will assume that all these values are distinct). By fixing $\tau := \hat{\tau}_s \in D_{sorted}$, (SDP-relaxed) becomes a *feasibility* problem as follows: In order to satisfy (20), $X_{m,n} = 0$ for all m and n with $d_{m,n}^{up} + d_{m,n}^{down} > \hat{\tau}_s$ must be true. Therefore, the following constraints are added:

$$X_{m,n} = 0, \forall m, n : d_{m,n}^{up} + d_{m,n}^{down} \in \{\hat{\tau}_{s+1}, \ldots, \hat{\tau}_{m,n},\} \quad (32)$$

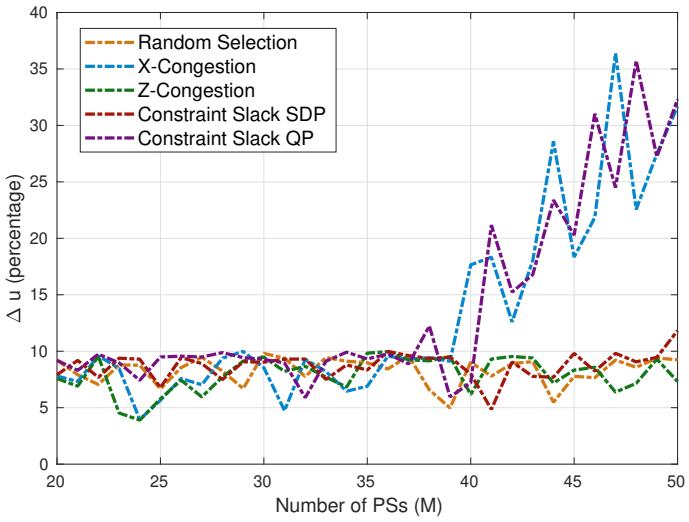
and are considered together with constraints (21)-(27) and (29). The relaxation of this feasibility problem is the new (SDP-relaxed) problem, and a (fractional) feasible solution can be obtained in polynomial time. If the problem is infeasible, then we proceed with a different (smaller) value of τ. Recall that we have assumed that the original (SDP-relaxed) is feasible, so there is at least one value of τ for which we will obtain a fractional feasible solution.

In Algorithm 2, first binary search is used in order to discover the smaller $\hat{\tau}_f \in D_{sorted}$ that maintains the feasibility of the (SDP-relaxed) problem. This is done after solving at most $O(\log(MN))$ SDPs. Note that $\tau_{opt}^{QP} \geq \hat{\tau}_f$, since the first is the integral optimum and the second the fractional one. Then binary search is used in set $\{\hat{\tau}_f, \hat{\tau}_{f+1}, \ldots, \hat{\tau}_{M,N}\}$, in order to find the smallest $\hat{\tau}_s$ for which $\Delta u \leq \epsilon_u$ when the rounding of Algorithm 1 is applied, where ϵ_u is the desired constraint violation tolerance. If no such solution is found, the algorithm reports failure (for the given tolerance).

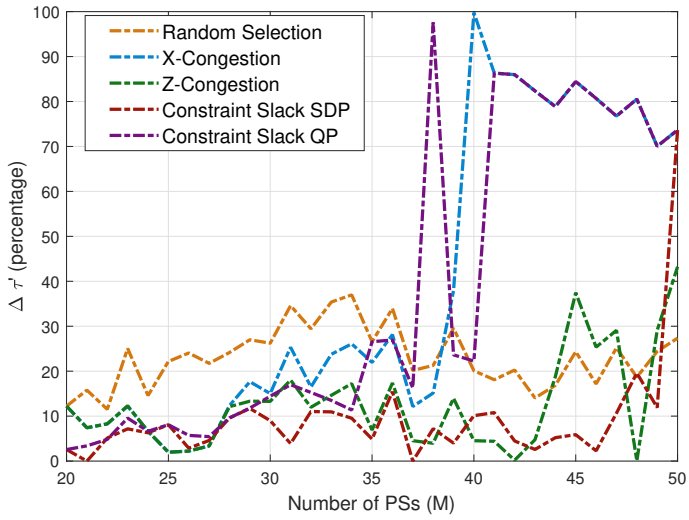
In Algorithm 2 we were aiming to find a solution with sub-ϵ_u constraint violation. Similarly, an ϵ_τ bound can be applied to the objective τ as follows: Let

$$\Delta\tau(x) = \frac{x - \tau_{opt}^{SP}}{\tau_{opt}^{SP}}, \quad (33)$$

where τ_{opt}^{SP} is the (fractional) optimum of the (SDP-relaxed). After calculating τ_{opt}^{SP}, the algorithm chooses the largest $\hat{\tau}_s$ from the set D_{sorted} with $\Delta\tau(\hat{\tau}_s) \leq \epsilon_\tau$. The (SDP-relaxed) problem for $\tau = \hat{\tau}_s$ is solved, and the fractional solution is rounded using Algorithm 1. If no such $\hat{\tau}_s$ exists, the algorithm terminates with infeasibility.



(a) Δu versus M



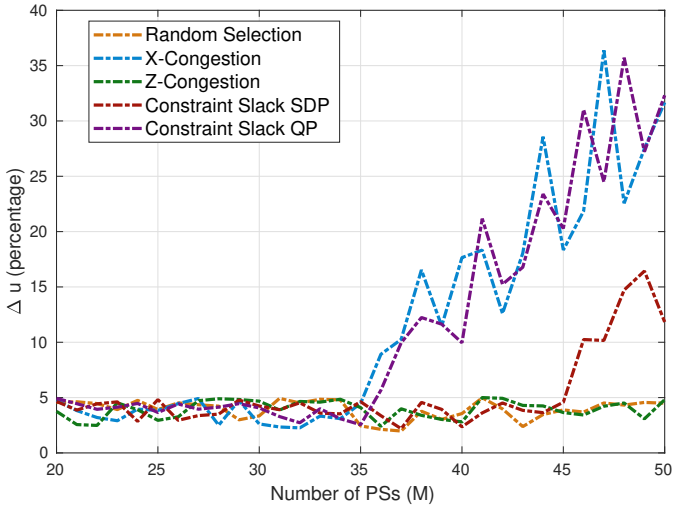
(b) $\Delta\tau'$ versus M

Fig. 3. Performance of Algorithm 2, $\epsilon_u = 10\%$.

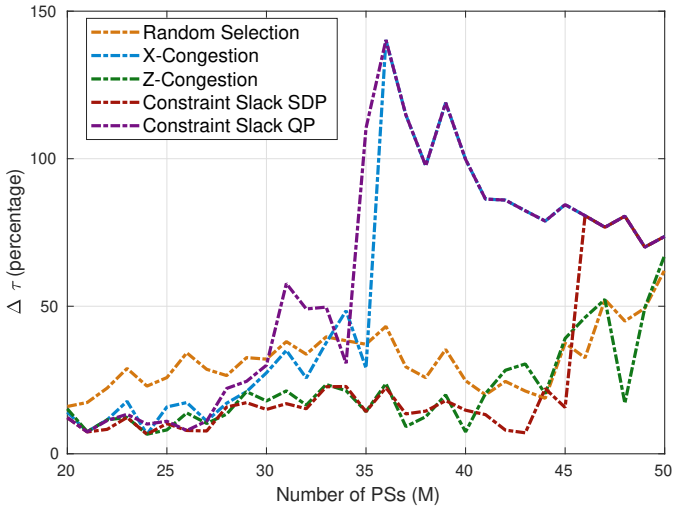
VI. Simulation Results

In this section we evaluate the performance of our proposed algorithms with via computer simulation. The algorithms are implemented using the five selection methods described in Section V. Since the solutions involve different relaxations of the constraints, the performance comparisons include the resulting constraint violation. Three sets of computer simulations were done to examine the performance of the algorithms from different perspectives. Mosek optimization toolbox [26] was used in Matlab for solving the optimization problems. In the presented figures, each point represents an average of 50 simulation runs. Since constraint (16) consists of $M \times N$ separate constraints, at each simulation run, the maximum violation of that set is used in the averaging.

In the experiments we will assume that the ESs are grouped in three groups: one that is nearest to the PSs and furthest from the AS (group 1), one that is in medium range from the PSs and AS (group 2). and one that is nearest to the AS and furthest from the PSs (group 3). The delays are defined accordingly:



(a) Δu versus M



(b) $\Delta\tau'$ versus M

Fig. 4. Performance of Algorithm 2, $\epsilon_u = 5\%$.

- The PS-ES delays d^{data} will be uniformly distributed in ranges $[5ms, 10ms]$, $[12.5ms, 17.5ms]$, $[20ms, 25ms]$ for the three groups, respectively.
- The AS-ES delays d^{down} will be uniformly distributed in ranges $[0.7ms, 1.1ms]$, $[0.4ms, 0.8ms]$, $[0.1ms, 0.5ms]$ for the three groups, respectively.
- The ES-AS delays d^{up} will be uniformly distributed in ranges $[16ms, 20ms]$, $[10ms, 14ms]$, $[4ms, 8ms]$ for the three groups, respectively.

These delay ranges are chosen by considering the fact that the 5G network backbone is capable of supporting bit rates ranging from 5G bits/s up to 10G bits/s at stable and uncongested network conditions [27] and some link processing overhead.

A. Simulation set 1

For the first set of simulations, we assigned 2 ESs to group 1, 4 ESs to group 2, and 2 ESs to group 3. The size of the data sent by a PS to its DT in every data update cycle is 25MB [22], the data size sent from a DT to the application server is 20MB [22], and the size of the request data from the application server to the DTs is 1MB [22]. In addition, T_m is uniformly distributed between 60ms and 80ms for each PS, and A^* is 200ms.

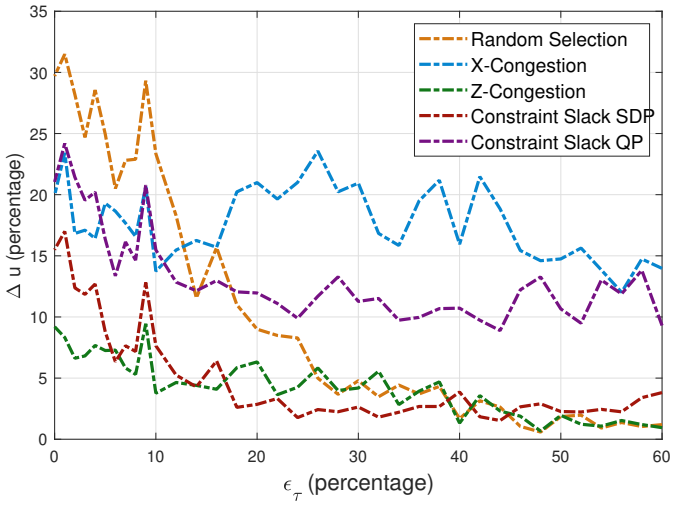


Fig. 5. Performance of Algorithm 3, Δu versus ϵ_τ, $M = 40$.

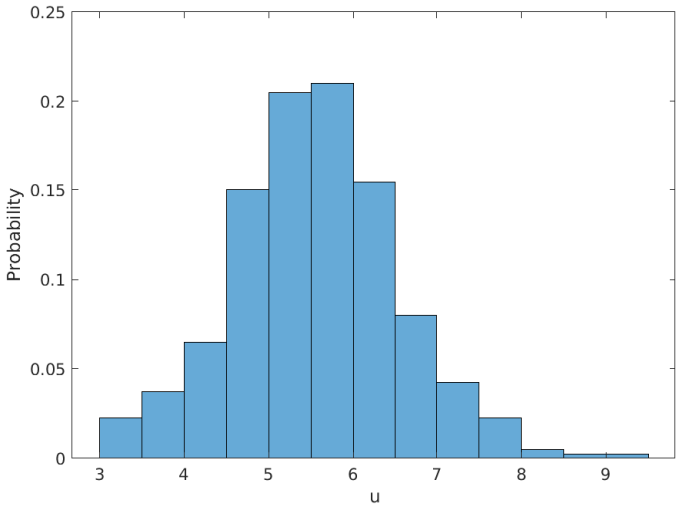
First, Algorithm 2 was run by varying the number of PSs. The results are plotted in figures 3a and 3b for $\epsilon_u = 10\%$ and figures 4a and 4b for $\epsilon_u = 5\%$. Figure 3a shows that all the methods can keep the Δu value below ϵ_u for a certain range of M values. However, compared to Constraint Slack QP and X-Congestion, Z-Congestion, Constraint Slack SDP, and Random Selection maintain $\Delta u < \epsilon_u$ for larger M values. The same trend is also seen in Figure 4a, although as ϵ_u becomes smaller, the range of M values for which $\Delta u < \epsilon_u$ is smaller for all methods, except for Z-congestion and Random Selection, which keep $\Delta < \epsilon_u$ for the entire range of M in the simulations.

Figures 3b and 4b show $\Delta\tau'$ values as the number of PSs change, where $\Delta\tau'$ is the relative difference between the resulted τ value after running Algorithm 2 and the optimum integer solution. More specifically, $\Delta\tau'$ is defined as

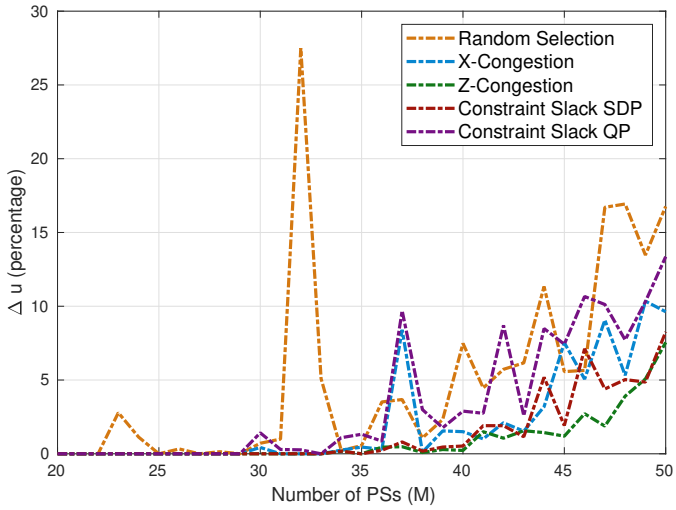
$$\Delta\tau' = \frac{\hat{\tau} - \tau_{opt}}{\tau_{opt}}, \tag{34}$$

where $\hat{\tau}$ is the objective value achieved by Algorithm 2, and τ_{opt} is obtained by solving (SDP-relaxed) in Section IV. Both figures 3b and 4b show that when M is too large for the algorithm to keep $\Delta u < \epsilon_u$, the corresponding $\Delta\tau'$ is also large. However, the Z-Congestion and Constraint Slack SDP achieve much smaller $\Delta\tau'$ than the other methods. A comparison between Figures 3b and 4b shows that a larger ϵ_u value helps reduce $\Delta\tau'$, as expected.

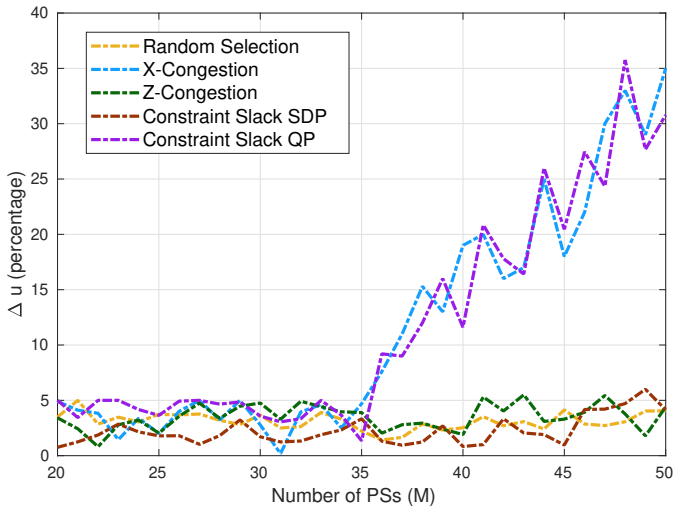
Next, Algorithm 3 was run with different ϵ_τ values and for each selection method. The corresponding maximum constraint violation Δu is given in Figure 5 as a percentage over its original u bound. In general, when ϵ_τ increases, the algorithm allows a larger $\Delta\tau$ value, which helps reduce the Δu value. This tradeoff between Δu and $\Delta\tau$ is well reflected in the Z-Congestion, the Constraint Slack SDP, and the Random Selection, but not obviously reflected in the X-Congestion



(a) Probability distribution of u



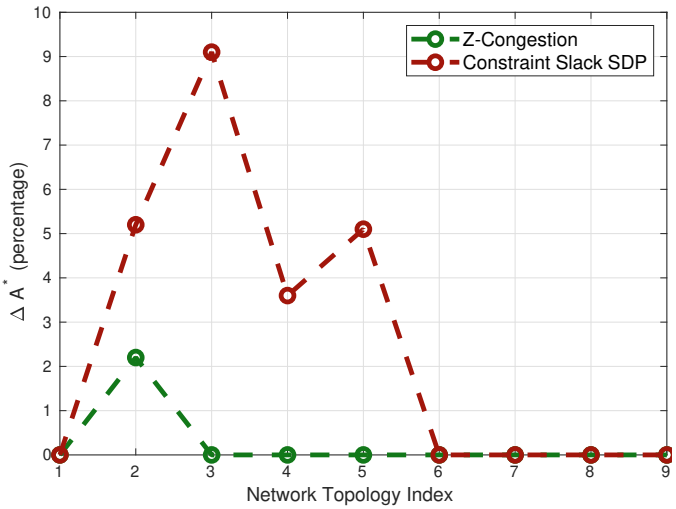
(b) Δu versus M, $\epsilon_\tau = 5\%$



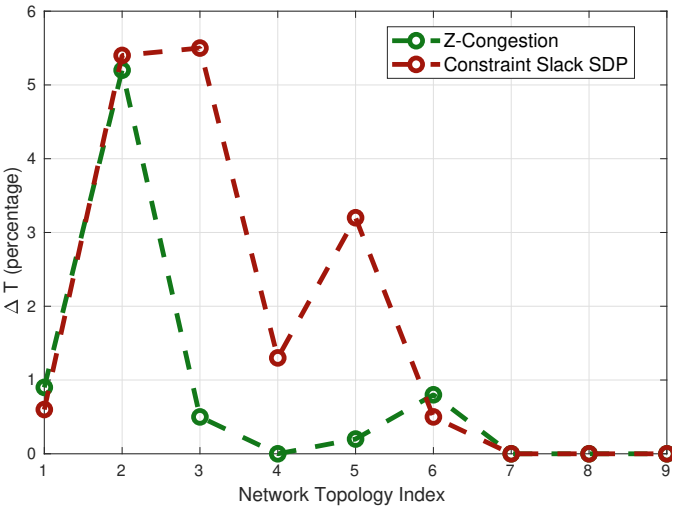
(c) Δu versus M, $\epsilon_u = 5\%$

Fig. 6. Constraint violation of algorithms over a non-uniform distribution of u.

and Constraint Slack QP. As can be observed from Figure 5,



(a)



(b)

Fig. 7. Performance of Constraint Slack SDP and Z-Congestion, $\epsilon_\tau = 5\%$ and $M = 30$.

Z-Congestion and Constraint Slack SDP achieve consistently smaller Δu values than Random Selection, Constraint Slack QP and X-Congestion for the entire range of ϵ_τ values.

Based on the above results, Constraint Slack SDP and Z-Congestion offer better overall performance than the other selection methods. This is not surprising, since these two methods use the constraints of (SDP-relaxed) as a guide for rounding its own fractional solution, while Constraint Slack QP and X-Congestion round the (SDP-relaxed) fractional solution using the constraints of (IQP') as their rounding criterion.

B. Simulation set 2

To further investigate the performance of the rounding methods, we ran another set of simulations. The network topology remains the same as in Section VI-A. Instead of specifying the values of T_m's, A^*, the $u_{m,n}$ values are randomly generated from the distribution of Figure 6a, which emulates a normal-like distribution. The results of running Algorithm 2 and 3

on the generated instances are plotted in Figures 6b and 6c. Again, Z-Congestion gives consistently lower Δu values, when compared to the other rounding methods.

C. Simulation set 3

The previous simulation results indicate that Z-Congestion and Constraint Slack SDP are superior to the others. Therefore, in the last set of simulations, we compare the performance of Z-Congestion and Constraint Slack SDP on the same network model as before but with different number of ESs at each set. More specifically, let s_1, s_2, and s_3 be the number of ESs in groups 1, 2, and 3, respectively. We consider different *network topologies* represented by the following (s_1, s_2, s_3) value combinations: $(2, 2, 4)$, $(4, 2, 2)$, $(2, 4, 2)$, $(2, 0, 6)$, $(6, 0, 2)$, $(4, 0, 4)$, $(0, 0, 8)$, $(0, 8, 0)$, and $(8, 0, 0)$. These combinations are indexed from 1 to 9, respectively. Based on these network topologies, the performance results after running Algorithm 3 are given in figures 7 and 8 for $M = 30$ and 40, respectively.

Instead of plotting Δu, we plotted ΔA^* and ΔT, which are defined similarly to Δu. More specifically, ΔA^* is calculated by comparing the A^* value after running the algorithm with the original A^* value, and ΔT is calculated by first comparing the T_m value after running the algorithm with the original T_m value for each m and taking the worst violation.

As shown in all the figures, Z-Congestion consistently exhibits lower constraint violation than Constraint Slack SDP because Z-congestion achieves a better load-balancing among the ESs. This happens because the constraint slacks used by Constraint Slack SDP depend on the server characteristics and parameters, while Z-Congestion ignores them and takes into account solely the load of DTs on the ESs. We observe that if the DT assignment decisions rely on both server characteristics and server load, as done by Constraint Slack SDP, the heuristic can be misled into decisions that leave the ESs unbalanced. Clearly, if all ESs share the same amount of resources and characteristics, the two heuristics would have the same performance; however, this is not the case in general, and is not the case in our simulations.

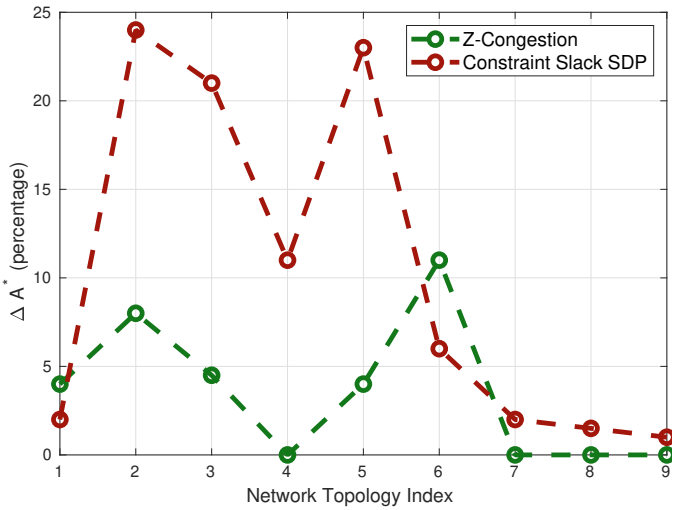


Fig. 8. Performance of Constraint Slack SDP and Z-Congestion, $\epsilon_\tau = 5\%$ and $M = 40$.

VII. Conclusions and Future Work

In this paper we have considered the problem of DT placement so that application data request timing latency targets are best accommodated. The objective is to minimize the data request response time at the application server subject to both the data age from the physical system to the application server, and the data update period between the physical systems and the DT execution servers. The problem was first formulated as an integer quadratic program (IQP), which was then transformed into a semidefinite program (SDP). Given the NP-completeness of the optimization problem, exact solutions are unavailable for practical systems. Practical polynomial-time approximation algorithms were introduced for solving the placement problem that provide different trade-offs between the accommodation of the application input timing latency and the achievement of data age targets. Through several simulations, it is shown that the Z-Congestion algorithm outperforms the rest in obtaining minimum constraint violation and timing latency.

In this work, we have implicitly assumed that the physical systems are immobile and their communication delay with their digital twins remains constant. As a potential future work, the optimal digital twin placement problem can be investigated for mobile physical systems and time-varying wireless transmission channels. In this case, the digital twins may have to be migrated between execution servers in order to accommodate the age of information targets of the applications.

References

[1] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," *Digital Twin White Paper*, 03 2015.

[2] ——, *Product Lifecycle Management: Driving the Next Generation of Lean Thinking*. New York, NY, USA: McGraw-Hill, 2006.

[3] Z. Liu, N. Meyendorf, and N. Mrad, "The role of data fusion in predictive maintenance using digital twin," in *AIP Conference Proceedings*, vol. 1949, no. 1. AIP Publishing LLC, 2018, p. 020023.

[4] R. Minerva, G. M. Lee, and N. Crespi, "Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models," *Proceedings of the IEEE*, vol. 108, pp. 1785–1824, 2020.

[5] F. Tao, H. Zhang, A. Liu, and A. Nee, "Digital Twin in Industry: State-of-the-Art," *IEEE Transactions on Industrial Informatics*, vol. 15, pp. 2405–2415, 2019.

[6] B. R. Barricelli, E. Casiraghi, and D. Fogli, "A survey on digital twin: Definitions, characteristics, applications, and design implications," *IEEE Access*, vol. 7, pp. 167 653–167 671, 2019.

[7] K. Zhang, J. Cao, S. Maharjan, and Y. Zhang, "Digital Twin Empowered Content Caching in Social-Aware Vehicular Edge Networks," *IEEE Trans. On Computational Social Systems*, 2020.

[8] Q. Song, S. Lei, W. Sun, and Y. Zhang, "Adaptive Federated Learning for Digital Twin Driven Industrial Internet of Things," *IEEE Wireless Communications and Networking Conference Workshops*, 2021.

[9] S. Chen, W. Meng, W. Xu, Z. Liu, J. Liu, and F. Wu, "A Warehouse Management System with UAV Based on Digital Twin and 5G Technologies," *International Conference on Information, Cybernetics, and Computational Social Systems*, 2020.

[10] Q. Yu, J. Ren, Y. Fu, Y. Li, and W. Zhang, "Cybertwin: An Origin of Next Generation Network Architecture," *IEEE Wireless Communications*, vol. 26, pp. 111–117, 2019.

[11] Z. Yin, N. Cheng, T. H. Luan, and P. Wang, "Physical layer security in cybertwin-enabled integrated satellite-terrestrial vehicle networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 5, pp. 4561–4572, 2022.

[12] X. Xu, B. Shen, S. Ding, G. Srivstava, M. Bilal, M. R. Khosravi, V. G. Menon, M. A. Jan, and M. Wang, "Service Offloading with Deep Q-Network for Digital Twinning Empowered Internet of Vehicles in Edge Computing," *IEEE Trans. On Industrial Informatics*, 2020.

[13] J. Deng, Q. Zheng, G. Liu, J. Bai, K. Tian, C. Sun, Y. Yan, and Y. Liu, "A Digital Twin Approach for Self-optimization of Mobile Networks," *IEEE WCNC'21*, 2021.

[14] T. Stockschläger. Dassault systèmes builds a second singapore. [Online]. Available: https://www.hannovermesse.de/en/news/news-articles/dassault-systemes-builds-a-second-singapore

[15] B. Van Hoof. Announcing azure digital twins: Create digital replicas of spaces and infrastructure using cloud, AI and IoT. [Online]. Available: https://azure.microsoft.com/en-us/blog/announcing-azure-digital-twins-create-digital-replicas-of-spaces\-and-infrastructure-using-cloud-ai-and-iot

[16] B. Li, M. J. Golin, G. F. Italiano, X. Deng, and K. Sohraby, "On the optimal placement of web proxies in the internet," in *Proceedings IEEE INFOCOM '99*, 1999, pp. 1282–1290.

[17] R. D. Yates, Y. Sun, D. R. Brown, S. K. Kaul, E. Modiano, and S. Ulukus, "Age of information: An introduction and survey," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 5, pp. 1183–1210, 2021.

[18] M. Vaezi, K. Noroozi, T. D. Todd, D. Zhao, G. Karakostas, H. Wu, and X. Shen, "Digital twins from a networking perspective," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 23 525–23 544, 2022.

[19] L. Wang, L. Jiao, T. He, J. Li, and M. Mühlhäuser, "Service entity placement for social virtual reality applications in edge computing," in *IEEE INFOCOM 2018*. IEEE, 2018, pp. 468–476.

[20] 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, 2021.

[21] O. Chukhno, N. Chukhno, G. Araniti, C. Campolo, A. Iera, and A. Molinaro, "Optimal placement of social digital twins in edge iot networks," *Sensors*, vol. 20, no. 21, p. 6181, 2020.

[22] P. Tiwari, "Mobility digital twin with connected vehicles and cloud computing," *IEEE Internet of Things*, 2021.

[23] J. Navarro-Ortiz, P. Romero-Diaz, S. Sendra, P. Ameigeiras, J. J. Ramos-Munoz, and J. M. Lopez-Soler, "A Survey on 5G Usage Scenarios and Traffic Models," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2, 2020.

[24] Q. Xu, J. Wang, and K. Wu, "Learning-based dynamic resource provisioning for network slicing with ensured end-to-end performance bound," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 28–41, 2020.

[25] K. Elsayed, "A framework for end-to-end deterministic-delay service provisioning in multiservice packet networks," *IEEE Transactions on Multimedia*, vol. 7, no. 3, pp. 563–571, 2005.

[26] A. Mosek, "The mosek optimization toolbox for matlab manual," 2015.

[27] "5g backbone," https://www.etsi.org/technologies/5G.