Online Optimization of Edge Empowered Human Digital Twin Deployment and Task Offloading

Yuye Yang*, You Shi*, Ruoyang Chen*, Changyan Yi*, Jiawen Kang†
*College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China
†School of Automation, Guangdong University of Technology, Guangzhou, China
E-mail: {mryyy, shyou, ruoyangchen, changyan.yi}@nuaa.edu.cn, kavinkang@gdut.edu.cn

2024 IEEE/CIC International Conference on Communications in China (ICCC) | 979-8-3503-7841-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCC62479.2024.10681718

***Abstract*—Human digital twin (HDT) is an emerging paradigm that constructs powerful virtual twins (VTs) of physical twins (PTs) for assisting human-centric and complex task executions. In this paper, a two-timescale online optimization for HDT deployment under an end-edge-cloud framework is studied. We consider that PTs' corresponding VTs are deployed on edge servers, including placing generic models by downloading experiential knowledge from the cloud and updating customized models by uploading personalized data from PTs. Taking into account HDT's unpredictable mobility and status variation, we dynamically optimize VTs' construction and PTs' task offloading, together with communication and computation resource allocations to maximize task execution accuracy under stringent delay constraint. Observing the asynchronization of different decision variables, we propose a novel two-timescale accuracy-aware online optimization approach (TACO). Specifically, TACO employs an improved Lyapunov method to decompose the problem into multiple instant ones, and then addresses the two-timescale issue alternately by leveraging piecewise McCormick envelopes and block coordinate descent based algorithms. Theoretical analyses and simulations evaluate the performance of the proposed solution, and show its superiority over counterparts.**

I. Introduction

Human digital twin (HDT) is defined as a paradigm that can vividly characterize the replication of individual human in the virtual space while real-time reflecting its actual physical and mental status in the physical space [1]. Because of the large potential in assisting complex task execution with human-centric concerns, HDT has been envisioned as a key enabler for Metaverse, Healthcare 5.0, Society 5.0, etc., attracting significant attentions recently [2].

Essentially, the HDT system consists of a number of physical twin (PT) and virtual twin (VT) pairs, where PT stands for the physical entity (i.e., human) and VT represents the corresponding virtual model. Obviously, the successful realization of HDT largely depends on the well construction and management of VT, so as to provide fast-responsive interactions, customized services and high-accurate task execution for its paired PT. These requirements prompt the adoption of end-edge-cloud collaborative framework, by which HDT can be built and operated at the network edge [3]. Although some preliminary efforts have been dedicated on studying similar problems, such as industrial digital twin construction at the edge and service application deployment across edges, establishing HDT at the edge for assisting task execution particularly involves some fundamentally different and unique issues that remain unexplored but are of great importance. On one hand, different from position-fixed industrial plants, PTs in the HDT system are highly mobile with unpredictable mobility patterns, leading to potential instability of PT-VT connectivities [1]. Therefore, it is necessary to dynamically place the associated VT of each PT on the edge server (ES) that this PT may switch its access to. On the other hand, unlike generalized applications requesting encapsulated services, PTs are extremely personalized and their status may vary frequently by uncertain external factors or physiological state changes, resulting in the potential inconsistency between each PT and its associated VT. Hence, it is necessary to keep the associated VT on the ES updated in a real-time manner. Nevertheless, meeting all aforementioned requirements are very challenging because of the following reasons.

1) To enhance the accuracy of complex task execution assisted by HDT, it is required to construct fine-grained VTs on ESs. However, the massive data brought by VT constructions [4] inherently increases the service delay, and thus the data size of generic model placement and customized model update should be carefully optimized for striking a balance between accuracy and service delay. Hence, how to efficiently offloading tasks from PTs to ESs should also be jointly considered with VT constructions, because both of them share the same communication and computation resources.
2) Since HDT is time-varying evolutionalized with uncertain PT-VT mobility and status variations, the optimization has to be conducted online while the future information may be hard to obtain [5]. Moreover, the dynamic placement of generic VT models is triggered by PTs' access handover, which usually happens over a long time period. In contrast, the dynamic update of customized VT models and task offloading are triggered by PTs' status variations, requiring to be adapted in a higher frequency. These indicate that such optimization should be performed asynchronously in different timescales.

In this paper, we study a two-timescale online optimization problem for building HDT in assisting complex task execution under an end-edge-cloud collaborative framework. With the objective of maximizing the average accuracy of complex task execution assisted by HDT under stringent delay constraint, and by taking into account the system uncertainties, we propose a novel two-timescale accuracy-aware online optimization approach (TACO) based on the improved Lyapunov optimization. Specifically, the long-term problem is first decomposed into a series of short-term deterministic

subproblems with different timescales, and then an alternating algorithm is proposed, integrating piecewise McCormick envelopes (PME) and block coordinate descent (BCD) based methods, for iteratively solving these subproblems.

The main contributions of this paper are in the following.

- We study the HDT deployment at the network edge for assisting human-centric task execution by formulating a two-timescale accuracy-aware online optimization problem, which jointly optimizes VTs' construction and PTs' task offloading together with PT-ES access selection and communication and computation resource allocations.
- We propose a novel approach, called TACO, which first decomposes the long-term problem into multiple instant ones. Then, we leverage PME and BCD based algorithms for alternately solving the decoupled subproblems.
- Extensive theoretical analyses and simulations are conducted to justify the feasibility of the proposed solution and show its superiority over counterparts.

The rest of this paper is organized as follows. Section II describes the system model and the problem formulation. In Section III, the two-timescale accuracy-aware online optimization approach is proposed and analyzed theoretically. Simulation results are presented in Section IV, followed by the conclusion in Section V.

II. System Model and Problem Formulation

Consider an HDT system building upon an end-edge-cloud collaborative framework, as illustrated in Fig. 1, consisting of a set of end users (regarded as PTs) $\mathcal{I}$ with cardinality of $|\mathcal{I}| = I$, multiple geographically distributed ESs denoted as $\mathcal{M}$ with $|\mathcal{M}| = M$, and a cloud center (acting as the central controller). PTs (roaming around) generate streams of complex tasks which require the construction of exclusive VT models (forming one-to-one PT-VT pairs) at the edge to assist their task executions. Each VT should be deployed on the ES to which its associated PT may access and one ES can host multiple VT models for different PTs, where communication and computation resources are shared among VTs. Furthermore, the construction of a high-fidelity VT on the ES consists of two main procedures, i.e., generic model placement and customized model update. For each VT, the generic part of the model is obtained by downloading the experiential knowledge with a selected granularity from the cloud center, and the target ES for its placement is determined following the access selection of the associated PT. By contrast, the customized part of each VT model is updated by uploading the personalized data with an optimized data size obtained from sensors worn on the associated PT. After VT establishment, PTs' tasks can be either offloaded to VTs deployed on the ES or processed locally. It is worth noting that although VT models are not able to be built locally, PTs' tasks can be executed by running offline service applications pre-installed on PTs, which are much less powerful but do not require to be real-time updated.

In practice, we define that in the considered online optimization framework, the access selection of each PT and the granularity of experiential knowledge for its generic VT model placement are decided in the large-timescale, while the amount of personalized data for its customized VT model update, task offloading, communication and computation resource allocations are decided in the small-timescale. Specifically, the timeline is segmented into $T \in \mathbb{N}^+$ coarse-grained time frames, and each frame can be further divided into a combination of $K \in \mathbb{N}^+$ fine-grained time slots. Let $t \in \mathcal{T} = \{0, 1, \ldots, T-1\}$ be the index of the t-th time frame, and define $\tau \in \mathcal{T}_t = \{tK, tK+1, \ldots, tK+K-1\}$ as the index of the τ-th time slot in the t-th time frame.

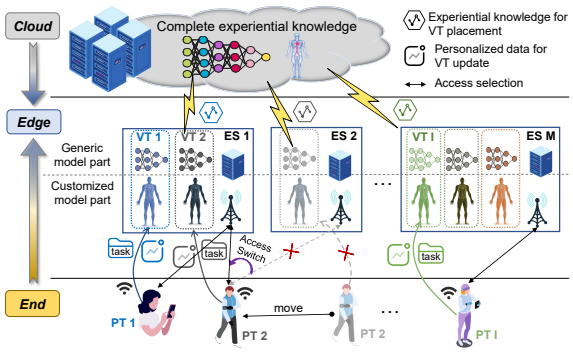

Fig. 1: The end-edge-cloud collaborative HDT system.

A. VT Model Deployment

Since PTs are mobile, to construct VT for each of them at the network edge so as to enable seamless PT-VT interactions, the generic VT model should be re-displaced on the ES that its associated PT switches its access to in each time frame $t \in \mathcal{T}$. Denote $a_{i,m}(t) \in \{0,1\}$ as the access selection decision in the large-timescale indicating whether PT $i \in \mathcal{I}$ selects to access ES $m \in \mathcal{M}$ or not in time frame $t \in \mathcal{T}$, i.e., $a_{i,m}(t) = 1$ if PT $i \in \mathcal{I}$ connects to ES $m \in \mathcal{M}$, and $a_{i,m}(t) = 0$ otherwise. For each PT $i \in \mathcal{I}$, we define that the full experiential knowledge of each PT $i \in \mathcal{I}$ for its generic VT model placement has a total size of $D_i(t)$, and denote $x_i(t) \in [0,1]$ as its decision of granularity in time frame $t \in \mathcal{T}$. Then, the corresponding data size can be represented as $x_i(t)D_i(t)$. Based on these, the corresponding delay of downloading such experiential knowledge can be expressed as $T_i^{dl}(t) = \sum_{m\in\mathcal{M}} a_{i,m}(t)(x_i(t)D_i(t))/r^c, \forall i \in \mathcal{I}, \forall t \in \mathcal{T}$, where r^c stands for the downlink transmission rate from the cloud center to each ES, which is considered as a constant.

Then, to exploit this experiential knowledge, each ES has to allocate a proportion of its computation resource for completing the generic VT model placement. The delay of doing this for PT $i \in \mathcal{I}$ on ES $m \in \mathcal{M}$ in time frame $t \in \mathcal{T}$ can be calculated as $T_{i,m}^{pl}(t) = (a_{i,m}(t)x_i(t)D_i(t)C_m)/(f_i(tK)F_m)$, where C_m is the number of CPU cycles required for ES m to process a unit of data, F_m is the CPU speed (measured by cycles/s) of ES m, and $f_i(tK) \in (0,1]$ represents the ratio of computation resource allocated to PT i for its VT construction at the beginning of time frame t.

Since PTs are personalized and their status may vary frequently due to uncertain external or internal factors, the customized VT model of each PT should be updated in each

time slot $\tau \in \mathcal{T}_t$. Let $S_i(\tau)$ be the total amount of personalized data generated by PT $i \in \mathcal{I}$, and define $y_i(\tau) \in [0,1]$ as the percentage of personalized data chosen to be uploaded in time slot $\tau \in \mathcal{T}_t$. Then, the size of personalized data uploaded for updating PT i's customized VT model in time slot $\tau \in \mathcal{T}_t$ can be expressed as $y_i(\tau)S_i(\tau)$. Within each time slot $\tau \in \mathcal{T}_t$, we denote the location of PT $i \in \mathcal{I}$ as $(x_i(\tau), y_i(\tau))$, which is a state information following its random mobility pattern, and let (x_m, y_m) be the fixed location of each ES $m \in \mathcal{M}$. The distance between any PT $i \in \mathcal{I}$ and ES $m \in \mathcal{M}$ can then be calculated as $S_{i,m}(\tau) = \sqrt{(x_i(\tau)-x_m)^2 + (y_i(\tau)-y_m)^2}$, and according to Shannon-Hartley formula, the transmission rate from PT $i \in \mathcal{I}$ to its accessed ES $m \in \mathcal{M}$ is written as $r_{i,m}(\tau) = a_{i,m}(t)b_i(\tau)B_m \log(1+\frac{p_i|h_{i,m}(\tau)|^2}{(S_{i,m}(\tau))^\theta N_0 b_i(\tau)B_m})$, where $b_i(\tau) \in (0,1]$ is the proportion of communication resource allocated to PT $i \in \mathcal{I}$ in time slot $\tau \in \mathcal{T}_t$, $h_{i,m}(\tau)$ is the fading amplitude between PT $i \in \mathcal{I}$ and ES $m \in \mathcal{M}$ in time slot $\tau \in \mathcal{T}_t$, B_m is the communication bandwidth of ES $m \in \mathcal{M}$, N_0 is the spectral density of the channel noise power, p_i is the pre-determined transmission power of PT $i \in \mathcal{I}$, and $\theta \geq 2$ is the path loss exponent. Correspondingly, the delay of PT $i \in \mathcal{I}$ in uploading the personalized data can be expressed as $T^{ul}_{i,m}(\tau) = (y_i(\tau)S_i(\tau))/(r_{i,m}(\tau))$.

Then, to utilize this personalized data, each ES has to allocate its computation resource for completing the customized VT model update. The delay of doing this for PT $i \in \mathcal{I}$ on ES $m \in \mathcal{M}$ in time slot $\tau \in \mathcal{T}_t$ can be calculated as $T^{ud}_{i,m}(\tau) = \frac{a_{i,m}(t)y_i(\tau)S_i(\tau)C_m}{f_i(\tau)F_m}$, where $f_i(\tau) \in (0,1]$ indicates the proportion of computation resource allocated to PT i for its VT update in each time slot $\tau \in [tK, tK+K-1]$.

B. HDT-Assisted Task Execution

Let $\lambda_i(\tau)$ be the data size of the complex task produced by PT $i \in \mathcal{I}$ in each time slot $\tau \in \mathcal{T}_t$, which is allowed to follow a general random distribution. Denote the task offloading decision of PT $i \in \mathcal{I}$ in time slot $\tau \in \mathcal{T}_t$ as $z_i(\tau) \in \{0,1\}$, i.e., $z_i(\tau) = 1$ if PT i offloads the task to its VT on the ES for assistance, and $z_i(\tau) = 0$ if PT i processes it locally. The delay of offloading such task from PT $i \in \mathcal{I}$ to its associated VT deployed on the ES $m \in \mathcal{M}$ in time slot $\tau \in \mathcal{T}_t$ can be expressed as $T^{ofld}_{i,m}(\tau) = \lambda_i(\tau)/r_{i,m}(\tau)$. Considering the possibility of both edge and local processing, the delay of HDT-assisted task execution of PT $i \in \mathcal{I}$ in time slot $\tau \in \mathcal{T}_t$ can be calculated as $T^{exec}_i(\tau) = \sum_{m\in\mathcal{M}} a_{i,m}(t)z_i(\tau)\frac{\lambda_i(\tau)C_m}{f_i(\tau)F_m} + (1-z_i(\tau))\frac{\lambda_i(\tau)C_i}{F_i}$, where C_i is the number of CPU cycles required for PT i to locally process a unit of data, and F_i denotes its CPU speed.

With the help of HDT at the edge, the accuracy of executing each task from PT $i \in \mathcal{I}$ in each time slot $\tau \in \mathcal{T}_t$ can be defined as $A_i(\tau) = z_i(\tau)g^{edge}_i(d_i(\tau)) + (1-z_i(\tau))g^{local}_i$, where g^{local}_i represents the task execution accuracy of local processing on PT i itself, and $g^{edge}_i(d_i(\tau))$ stands for the task execution accuracy of edge computing for PT i depending on the total size of data used for its VT construction, i.e., $d_i(\tau) = x_i(t)D_i(t) + y_i(\tau)S_i(\tau), \forall\tau \in \mathcal{T}_t, \forall t \in \mathcal{T}$. Note that $g^{edge}_i(d_i(\tau))$ is a mapping function that can be obtained via empirical studies or experiments [6].

C. Problem Formulation

In summary, the total response delay for all tasks of PT $i \in \mathcal{I}$ in each time frame $t \in \mathcal{T}$ can be derived as $T^{tol}_i(t) = T^{dl}_i(t) + \sum_{m\in\mathcal{M}} T^{pl}_{i,m}(t) + \sum_{\tau\in\mathcal{T}_t}[z_i(\tau)\sum_{m\in\mathcal{M}}(T^{ul}_{i,m}(\tau) + T^{ud}_{i,m}(\tau) + T^{ofld}_{i,m}(\tau)) + T^{exec}_i(\tau)]$. Then, we take the long-term average accuracy of task execution assisted by the considered end-edge-cloud collaborative HDT system over all time frames as the performance measurement, which can be expressed as

$$\mathcal{A} = [\sum\nolimits_{t\in\mathcal{T}} \sum\nolimits_{\tau\in\mathcal{T}_t} \sum\nolimits_{i\in\mathcal{I}} A_i(\tau)]/TK. \tag{1}$$

With the objective of maximizing $\mathcal{A}$ while ensuring that the response delay for all tasks of each PT $i \in \mathcal{I}$ does not exceed certain threshold T^{max}_i, we formulate a two-timescale online optimization problem by jointly optimizing $\mathcal{J}^A_i(t) = \{a_{i,m}(t), x_i(t)\}$ in any time frame $t \in \mathcal{T}$, and $J^B_i(\tau) = \{b_i(\tau), y_i(\tau), f_i(\tau), z_i(\tau)\}$ in any time slot $\tau \in \mathcal{T}_t$.

Mathematically, such a two-timescale online optimization problem can be formulated as

$$\mathcal{P}_1: \max_{\mathcal{J}^A_i(t), \mathcal{J}^B_i(\tau)} \lim_{t\to\infty} \mathcal{A}$$

$$\text{s.t.} \sum\nolimits_{m\in\mathcal{M}} a_{i,m}(t) \leq 1, \tag{2a}$$

$$\sum\nolimits_{i\in\mathcal{I}} a_{i,m}(t)b_i(\tau) \leq 1, \tag{2b}$$

$$\sum\nolimits_{i\in\mathcal{I}} a_{i,m}(t)f_i(\tau) \leq 1, \tag{2c}$$

$$\lim_{T\to\infty} \sum\nolimits_{t\in\mathcal{T}} T^{tol}_i(t)/T \leq T^{max}_i, \tag{2d}$$

where constraint (2a) guarantees that one PT can connect to at most one ES; constraints (2b) and (2c) restrict that the communication and computation resource allocation should be less than the total capacities of each ES; constraints (2d) is the long-term average task response delay constraint.

Obviously, solving $\mathcal{P}_1$ directly is very challenging because i) PTs' mobilities and status variations are extremely hard to obtain system statistics in advance, which necessitates the design of an online optimization algorithm; ii) although Lyapunov method [7] is an effective method to solve such problem, decision variables in different timescales are tightly coupled in the objective function and constraint (2d); and iii) all constraints include discrete decision variables, and constraint (2d) is non-convex. These indicate that $\mathcal{P}_1$ is a two-timescale online non-convex mixed integer programming problem, which must be NP-hard.

III. Two-Timescale Online Optimization Approach

A. Problem Reformulation and Decomposition

Observed from problem $\mathcal{P}_1$ that the delay caused by generic VT placement and customized VT update are on the different timescales. To facilitate the solution, we evenly distribute the task response delay in each time frame $t \in \mathcal{T}$ into all $|\ \mathcal{T}_t\ | = K$ time slots within this frame, which yields $T^{tol}_i(\tau) = (T^{dl}_i(t) + \sum_{m\in\mathcal{M}} T^{pl}_{i,m}(t))/K + \sum_{\tau\in\mathcal{T}_t}[z_i(\tau)\sum_{m\in\mathcal{M}}(T^{ul}_{i,m}(\tau) + T^{ud}_{i,m}(\tau) + T^{ofld}_{i,m}(\tau)) + T^{exec}_i(\tau)]$. Substituting it into (2d) of problem $\mathcal{P}_1$, we have

$$\mathcal{P}_2: \max_{\mathcal{J}^A_i(t), \mathcal{J}^B_i(\tau)} \lim_{t\to\infty} \mathcal{A}$$

$$\text{s.t. } (2a),(2b),(2c),$$
$$\lim_{T\to\infty}[\sum\nolimits_{t\in\mathcal{T}}\sum\nolimits_{\tau\in\mathcal{T}_t} T_i^{tol}(\tau)]/TK \le T_i^{max}/K, \quad (3a)$$

Note that the reformulated problem $\mathcal{P}_2$ is equivalent to $\mathcal{P}_1$ with exactly the same decision variables remaining in two different timescales, while all long-term constraints have been unified into a single timescale but will not affect the optimization performance. Then, we employ the idea of Lyapunov method and modify it to accommodate the characteristics of $\mathcal{P}_2$.

We first define a delay overflow queue to describe how task response delay $T_i^{tol}(\tau)$ of each PT $i \in \mathcal{I}$ in time slot $\tau \in \mathcal{T}_t$ may deviate from the long-term budget T_i^{max}/K. The dynamic evolution of this queue can be expressed as $H_i(\tau+1) = \max[H_i(\tau) + T_i^{tol}(\tau) - T_i^{max}/K, 0]$.

After that, we combine the delay overflow queue $H_i(\tau)$ for all tasks of PTs by a vector as $\mathbf{\Theta}(\tau) = [\boldsymbol{H}(\tau)]$, and introduce a quadratic Lyapunov function [7] as $L(\mathbf{\Theta}(\tau)) \triangleq \frac{1}{2}[\sum_{i\in\mathcal{I}} H_i(\tau)^2]$, which reflects the congestion of all queues, and should be pushed towards a minimum value to keep queue stabilities. The conditional Lyapunov drift is given by

$$\Delta(\mathbf{\Theta}(\tau)) = \mathbb{E}[L(\mathbf{\Theta}(\tau+K)) - L(\mathbf{\Theta}(\tau))|\mathbf{\Theta}(\tau)], \quad (4)$$

where $\mathbb{E}[\cdot]$ denotes the expectation, and $\Delta(\mathbf{\Theta}(\tau))$ measures the difference of the Lyapunov function between K consecutive time slots. Intuitively, by minimizing the Lyapunov drift in (4), we can prevent queue backlog from unbounded growth, and thus guarantee that the desired delay constraint can be met.

Hence, the Lyapunov drift-plus-penalty function becomes $\Delta(\mathbf{\Theta}(\tau)) - V\mathbb{E}[\sum_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)]$, where a control parameter $V > 0$ is introduced, representing the weight of significance on maximizing the HDT-assisted task execution accuracy versus that of strictly satisfying the delay constraint.

Theorem 1: Let $V>0$, and the drift-plus-penalty is bounded by any possible decisions in any time slot $\tau \in \mathcal{T}_t$, i.e.,

$$\begin{aligned}&\Delta(\mathbf{\Theta}(\tau)) - V\mathbb{E}[\sum\nolimits_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)]\\ &\le G + \sum\nolimits_{i\in\mathcal{I}} \mathbb{E}[H_i(\tau)(T_i^{tol}(\tau) - T_i^{max}/K) \mid \mathbf{\Theta}(\tau)]\\ &- V\mathbb{E}[\sum\nolimits_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)],\end{aligned} \quad (5)$$

where $G = \frac{1}{2}\sum_{i\in\mathcal{I}}[T_i^{tol}(max) - T_i^{max}]^2$ is a positive constant that adjusts the tradeoff between the task execution accuracy and the satisfaction degree of the delay constraint.

Proof: This proof is omitted due to the page limit. ■

Theorem 1 shows that the drift-plus-penalty is upper bounded in each time slot τ. Then, taking the sum over all time slots within time frame t for both sides of (5), we have

$$\begin{aligned}&\sum\nolimits_{\tau\in\mathcal{T}_t} \Delta(\mathbf{\Theta}(\tau)) - V\sum\nolimits_{\tau\in\mathcal{T}_t} \mathbb{E}[\sum\nolimits_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)] \le GK\\ &+ \sum\nolimits_{\tau\in\mathcal{T}_t}\sum\nolimits_{i\in\mathcal{I}} \mathbb{E}[H_i(\tau)(T_i^{tol}(\tau) - T_i^{max}/K) \mid \mathbf{\Theta}(\tau)]\\ &- V\sum\nolimits_{\tau\in\mathcal{T}_t} \mathbb{E}[\sum\nolimits_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)].\end{aligned} \quad (6)$$

Then, $\mathcal{P}_2$ can be decomposed into multiple subproblems, each of which opportunistically minimize the right-hand-side of (6) in one time frame $t \in \mathcal{T}$, i.e.,

$$\begin{aligned}\mathcal{P}_3 : \min_{\mathcal{J}_i^A(t),\mathcal{J}_i^B(\tau)} &\sum\nolimits_{\tau\in\mathcal{T}_t}\sum\nolimits_{i\in\mathcal{I}} \mathbb{E}[H_i(\tau)(T_i^{tol}(\tau) - T_i^{max}/K)\\ &\mid \mathbf{\Theta}(\tau)] + \sum\nolimits_{\tau\in\mathcal{T}_t} -V\sum\nolimits_{\tau\in\mathcal{T}_t} \mathbb{E}[\sum\nolimits_{i\in\mathcal{I}} A_i(\tau) \mid \mathbf{\Theta}(\tau)]\\ &\text{s.t. } (2a),(2b),(2c).\end{aligned}$$

Note that, in problem $\mathcal{P}_3$, decisions $\mathcal{J}_i^A(t)$ and $\mathcal{J}_i^B(\tau)$ remain unchanged as those in $\mathcal{P}_3$. This means that, although $\mathcal{P}_3$ focuses on the optimization in a single time frame, it still includes two-timescale variables.

B. Alternating Algorithm between Two Timescales

1) Two-Timescale Decoupling and Alternation: To solve problem $\mathcal{P}_3$, we can decouple it into two subproblems, and then solve them alternately till the convergence.

Large-timescale Problem: Given the small-timescale decision $\mathcal{J}_i^B(\tau)$ and the current backlogs of delay overflow queues $H_i(\tau)$, the large-timescale subproblem can be formulated as

$$\begin{aligned}\mathcal{P}_4 : \min_{\mathcal{J}_i^A(t)} &\sum\nolimits_{i\in\mathcal{I}} H_i(\tau)[(T_i^{dl}(t) + \sum\nolimits_{m\in\mathcal{M}} T_{i,m}^{pl}(t))/K\\ &+ z_i(\tau)\sum\nolimits_{m\in\mathcal{M}}(T_{i,m}^{ul}(\tau) + T_{i,m}^{ud}(\tau) + T_{i,m}^{ofld}(\tau))\\ &+ T_i^{exec}(\tau)] - V\sum\nolimits_{i\in\mathcal{I}} A_i(\tau)\\ &\text{s.t. } (2a),(2b),(2c).\end{aligned}$$

Small-timescale Problem: Given the large-timescale decision $\mathcal{J}_i^A(t)$, the small-timescale subproblem can be given as

$$\begin{aligned}\mathcal{P}_5 : \min_{\mathcal{J}_i^B(\tau)} &\sum\nolimits_{i\in\mathcal{I}} H_i(\tau)[\sum\nolimits_{m\in\mathcal{M}} T_{i,m}^{pl}(t)/K + z_i(\tau)\sum\nolimits_{m\in\mathcal{M}}\\ &(T_{i,m}^{ul}(\tau) + T_{i,m}^{ud}(\tau) + T_{i,m}^{ofld}(\tau)) + T_i^{exec}(\tau)] - V\sum\nolimits_{i\in\mathcal{I}} A_i(\tau)\\ &\text{s.t. } (2b),(2c).\end{aligned}$$

Alternating Process: For $\tau = tK$, we iteratively optimize $\mathcal{P}_4$ and $\mathcal{P}_5$ until the objective of $\mathcal{P}_3$ converges. Specifically, by fixing $\mathcal{J}_i^B(tK) = J_i^B(tK-1)$ (inherited from the last time slot $\tau = tK-1$ of the previous time frame $\mathcal{T}_{t-1}$), large-timescale subproblem $\mathcal{P}_4$ is first solved to optimize $\mathcal{J}_i^A(tK)$. Then, given $\mathcal{J}_i^A(tK)$, small-timescale subproblem $\mathcal{P}_5$ is solved to update $\mathcal{J}_i^B(tK)$, which is returned back to $\mathcal{P}_4$. For each $\tau \in [tK+1, tK+K-1]$, with the optimized $\mathcal{J}_i^A(tK)$ after the convergence, we repeatedly solve $\mathcal{P}_5$ to obtain $\mathcal{J}_i^B(\tau)$ in each time slot.

2) Solution for Large-Timescale Decisions: Non-convex subproblem $\mathcal{P}_4$ in large-timescale can be regarded as a bilinear optimization problem, which motivates us to design a PME-based algorithm [8]. Specifically, we first construct convex envelops for bilinear terms, transform $\mathcal{P}_4$ to a piecewise linear form, and then solve it by *partitioning* and *pruning*.

Let $u_{i,m}(t) = a_{i,m}(t)x_i(t)$ be an auxiliary variable of bivariate $a_{i,m}(t)x_i(t)$, and divide $\boldsymbol{x} = \{x_i(t), \forall i \in \mathcal{I}, \forall t \in \mathcal{T}\}$ and $\boldsymbol{a} = \{a_{i,m}(t), \forall i \in \mathcal{I}, \forall i \in \mathcal{I}, \forall t \in \mathcal{T}\}$ into $\mid \mathcal{N} \mid = N$ partitions. Specifically, we denote $x_{i,n}(t) \in [x_{i,n}^L(t), x_{i,n}^U(t)]$ as the range of $x_{i,n}(t)$ in partition $n \in \mathcal{N}$, where $x_{i,n}^L(t)$ and $x_{i,n}^U(t)$ are its lower and upper bounds, respectively. Besides, a new auxiliary variable $y_{i,n}(t) \in \{0,1\}$ is introduced, where $y_{i,n}(t) = 1$ if the value of $x_i(t)$ belongs to partition n, and $y_{i,n}(t) = 0$ otherwise. Similarly, the binary variable $a_{i,m}(t)$ is first relaxed into a continuous variable $\tilde{a}_{i,m}(t) \in [0,1]$ and then divided into N piecewise areas, where the range of partition $n \in \mathcal{N}$ is $\tilde{a}_{i,m,n}(t) \in [\tilde{a}_{i,m,n}^L(t), \tilde{a}_{i,m,n}^U(t)]$. Then, by

applying convex hull relaxation [9], $\mathcal{P}_4$ can be relaxed into a piecewise linear programming problem, which is expressed as

$$\mathcal{P}_4^1: \min_{\mathcal{J}_i^A(t),u_{i,m}(t)} \sum\nolimits_{i\in\mathcal{I}} H_i(\tau)[(T_i^{dl}(t)+\sum\nolimits_{m\in M} u_{i,m}(t)$$
$$(D_i(t)C_m)/(f_i(tK)F_m))/K + z_i(\tau)\sum\nolimits_{m\in\mathcal{M}}(T_{i,m}^{ul}(\tau)$$
$$+T_{i,m}^{ud}(\tau)+T_{i,m}^{ofld}(\tau))+T_{i,m}^{exec}(\tau)] - V\sum\nolimits_{i\in\mathcal{I}} A_i(\tau)$$
$$\text{s.t. } (2a),(2b),(2c),$$
$$u_{i,m}(t) \geqslant \hat{a}_{i,m,n}(t)\cdot x_{i,n}^L(t) + \tilde{a}_{i,m,n}^L(t)\cdot \hat{x}_{i,n}(t)$$
$$-\tilde{a}_{i,m,n}^L(t)\cdot x_{i,n}^L(t)\cdot y_{i,n}(t), \forall i,\forall m,\forall n,\forall t,$$
$$u_{i,m}(t) \geqslant \hat{a}_{i,m,n}(t)\cdot x_{i,n}^u(t) + \tilde{a}_{i,m,n}^U(t)\cdot \hat{x}_{i,n}(t)$$
$$-\tilde{a}_{i,m,n}^U(t)\cdot x_{i,n}^u(t)\cdot y_{i,n}(t), \forall i,\forall m,\forall n,\forall t,$$
$$u_{i,m}(t) \leqslant \hat{a}_{i,m,n}(t)\cdot x_{i,n}^L(t) + \tilde{a}_{i,m,n}^U(t)\cdot \hat{x}_{i,n}(t)$$
$$-\tilde{a}_{i,m,n}^U(t)\cdot x_{i,n}^L(t)\cdot y_{i,n}(t), \forall i,\forall m,\forall n,\forall t,$$
$$u_{i,m}(t) \leqslant \hat{a}_{i,m,n}(t)\cdot x_{i,n}^u(t) + \tilde{a}_{i,m,n}^L(t)\cdot \hat{x}_{i,n}(t)$$
$$-\tilde{a}_{i,m,n}^L(t)\cdot x_{i,n}^u(t)\cdot y_{i,n}(t), \forall i,\forall m,\forall n,\forall t,$$
$$\tilde{a}_{i,m}(t) = \sum\nolimits_{n\in\mathcal{N}} \hat{a}_{i,m,n}(t), \forall i,\forall m,\forall t,$$
$$x_i(t) = \sum\nolimits_{n\in\mathcal{N}} \hat{x}_{i,n}(t), \forall i,\forall t,$$
$$\sum\nolimits_{n\in\mathcal{N}} y_{i,n}(t) = 1, \forall t,$$
$$x_{i,n}^L(t) = x_i^L(t) + (x_i^U(t) - x_i^L(t))\cdot n(n-1)/N, \forall i,\forall n,\forall t,$$
$$x_{i,n}^U(t) = x_i^L(t) + (x_i^U(t) - x_i^L(t))\cdot nn/N, \forall i,\forall n,\forall t,$$
$$x_{i,n}^L(t)y_{i,n}(t) \leqslant \hat{x}_{i,n}(t) \leqslant x_{i,n}^U(t)y_{i,n}(t), \forall i,\forall n,\forall t,$$
$$\tilde{a}_{i,m,n}^L(t)y_{i,n}(t) \leqslant \hat{a}_{i,m,n}(t) \leqslant \tilde{a}_{i,m,n}^U(t)y_{i,n}(t), \forall i,\forall m,\forall n,\forall t,$$
$$y_{i,n}(t) \in \{0,1\}, \forall n,\forall t.$$

Then, we prune $x_i(t)$ and $\tilde{a}_{i,m}(t)$ to tighten their relaxed bounds. By traversing each partition n of $x_i(t)$, we first determine the lower bound $\tilde{a}_{i,m,n}^L(t)$ and upper bound $\tilde{a}_{i,m,n}^U(t)$ of $\tilde{a}_{i,m}(t)$ by solving the following linear programming problem:

$$\begin{aligned}
&\tilde{a}_{i,m,n}^L(t) = \min \tilde{a}_{i,m}(t) \text{ or } \tilde{a}_{i,m,n}^U(t) = \max \tilde{a}_{i,m}(t)\\
&\text{s.t. } (2a),(2b),(2c),\\
&u_{i,m}(t)\geq 0, \forall i\in\mathcal{I},\forall m\in\mathcal{M},\forall t\in\mathcal{T},\\
&u_{i,m}(t)\geq x_i(t)+a_{i,m}(t)-1, \forall i\in\mathcal{I},\forall m\in\mathcal{M},\forall t\in\mathcal{T},\\
&u_{i,m}(t)\leq a_{i,m}(t), \forall i\in\mathcal{I},\forall m\in\mathcal{M},\forall t\in\mathcal{T},\\
&u_{i,m}(t)\leq x_i(t), \forall i\in\mathcal{I},\forall m\in\mathcal{M},\forall t\in\mathcal{T},\\
&obj(\mathcal{P}_4^1)\leqslant z',\\
&x_i^L(t) = x_{i,n}^L(t) \leqslant x_i(t) \leqslant x_{i,n}^U(t) = x_i^U(t).
\end{aligned} \tag{7}$$

Besides, the range of $x_i(t)$ is updated as $x_i^L(t) = \min_n x_{i,n}^L(t)$ and $x_i^U(t) = \max_n x_{i,n}^U(t)$ after traversing all the partitions of $x_i(t)$. Then, after pruning all $x_i(t)$ and $\tilde{a}_{i,m}(t)$, we can solve problem $\mathcal{P}_4^1$ in the pruned partition with software-based optimization solvers (e.g., CVX), and obtain its solution $(\boldsymbol{x}^R, \tilde{\boldsymbol{a}}^R)$. Lastly, we round continuous variables $\tilde{\boldsymbol{a}}^R$ to binary forms for obtaining integer solutions.

3) Solution for Small-Timescale Decisions: Small-timescale subproblem $\mathcal{P}_5$ is also non-convex in general, but by relaxing the integer decision $z_i(\tau)$ (i.e., $z_i(\tau)\in\{0,1\} \rightarrow z_i(\tau)\in[0,1]$), it becomes a block multi-convex problem, which motivates us to design a BCD-based algorithm. Specifically, by fixing arbitrarily three decision variables and optimizing the remaining one, $\mathcal{P}_5$ is divided into four convex subproblems, which can be easily solved by existing software-based optimization solvers (e.g. CVX). Note that all these problems are solved iteratively, and the iteration finishes when the objective of problem $\mathcal{P}_5$ converges.

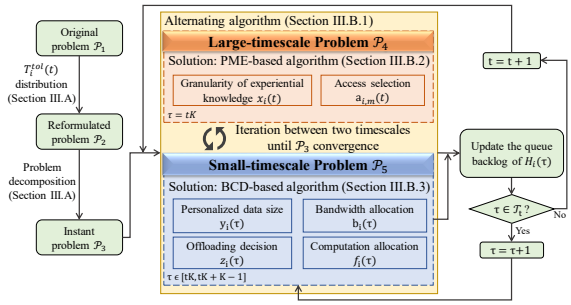

Fig. 2: Flowchart of the proposed TACO approach.

TABLE I: Simulation Parameters

Param	Value	Param	Value	Param	Value
p_i	500 mW	B_m	5 MHz	$S_i(\tau)$	[6.1, 12.2] Mbits
p^c	5 W	F_m	20 GHz	$D_i(t)$	[73.2, 97.6] Mbits
θ	4	C_m, C_i	300 cycles/bit	$\lambda_i(\tau)$	[10, 20] Mbits
K	10	ρ_m, ρ_i	10^{-27}	N_0	-174 dBm/Hz
F_i	1 GHz	r^c	50 Mbps		

C. Analysis of Proposed TACO Approach

In summary, the proposed TACO approach consists of problem reformulation, decomposition and alternation between two timescales. The flowchart of TACO is shown in Fig. 2.

***Theorem** 2:* The proposed TACO approach can converge with limited alternations and iterations.

Proof: This proof is omitted due to the page limit. ■

***Theorem** 3:* Given Lyapunov control parameter V, the optimality gap between the solution obtained by the proposed TACO approach and the theoretically optimal solution to problem $\mathcal{P}_1$ can be expressed as

$$\sum_{t\in\mathcal{T}}\sum_{\tau\in\mathcal{T}_t}\mathbb{E}[\sum_{i\in\mathcal{I}} A_i(\tau)|\boldsymbol{\Theta}(\tau)]/KT - \mathcal{O} \leq G/V + (\Lambda+\Gamma)/VT, \tag{8}$$

where $\mathcal{O}$ stands for the theoretically optimal solution, Λ and Γ are respectively optimality gaps of the PME-based and BCD-based algorithms, and G is defined in (5).

Proof: This proof is omitted due to the page limit. ■

IV. Simulation Results

Consider an HDT system in a $1km\times 1km$ square area with $M=10$ ESs and $I=40$ PTs. The average accuracy of edge execution and local execution for any PT i' s tasks are approximated as $g_i^{edge}(d_i(\tau)) = 1-[1-d_i(\tau)/(D_i(t)+S_i(\tau))]^2$ and $g_i^{local}=0.5$, respectively. Similar to [5], [10], simulation parameters are listed in table I. Furthermore, to show the superiority of the proposed TACO approach, we compare it with i) LOT [11], which focuses on generic VT placement and task offloading without considering customized VT update in a single-timescale only. ii) CRO [12], which optimizes both

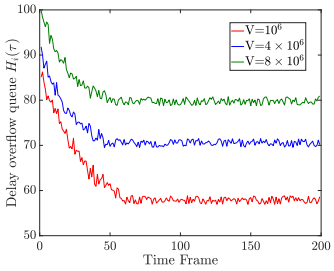

Fig. 3: Queue backlogs w.r.t V.

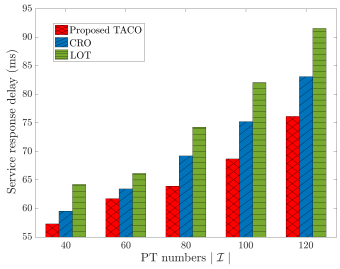

Fig. 4: Service delay w.r.t | I |.

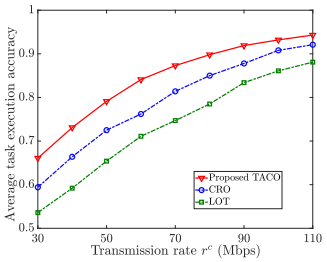

Fig. 5: Accuracy w.r.t r^c.

generic VT placement and customized VT update along with task offloading in a single-timescale.

Fig. 3 demonstrates the stability of the proposed TACO approach by showing the performance of the queue brought by the Lyapunov decomposition by varying V. From this figure, we can see that the delay overflow queue backlog decreases and quickly stabilizes over the time, because TACO focuses on controlling service response delay to minimize the objective function of $\mathcal{P}_3$, thereby shrinking the queue backlog, and can eventually achieve a well balance between the task execution accuracy and service response delay. Besides, it is intuitive that the queue backlog stabilize on higher values with the increase of V as more emphasis is on maximizing the task execution accuracy, resulting in the growth of delay.

In Fig. 4, the performance comparisons of service response delay with different numbers of PTs I are illustrated. This figure shows that the service response delay increases exponentially with I for all schemes, because a larger I implies more demands for VT construction with potentially larger amount of data in competing limited communication and computation resources. Besides, the proposed TACO approach achieves the best performance. This is because TACO builds VTs in two timescales which only requires to download experiential knowledge at the beginning of each time frame significantly reducing the total data size in the process of model placement. Meanwhile, CRO and TACO outperform LOT, especially when I becomes larger, which reveals the necessity of customized VT model update in addition to generic VT model placement.

Fig. 5 illustrates the performance comparisons of task execution accuracy by varying transmission rate r^c. It is shown that, the average task execution accuracy increases with r^c for all three schemes. The reason is that, when r^c is large, more experiential knowledge and personalized data can be transmitted for VT construction and more tasks can be offloaded for HDT-assisted edge execution. Furthermore, we can see that LOT has the worst performance because it only considers generic VT placement, and CRO is better than LOT thanks to the joint consideration of both generic VT placement and customized VT update. Moreover, TACO achieves the best performance because it further strikes the balance of generic VT placement and customized VT update by conducting these two processes asynchronously in two different timescales.

V. Conclusion

In this paper, the optimization of HDT deployment at the network edge has been studied. Particularly, aiming to maximize the accuracy of complex task execution assisted by HDT under stringent delay constraint, a two-timescale online optimization problem is formulated for jointly determining VTs' construction (including dynamic generic model placement and customized model update) and PTs' task offloading together with access selection, and communication and computation resource allocations. Then, we propose an accuracy-aware online optimization approach, called TACO, which efficiently solves the formulated problem under various system uncertainties. Theoretical analyses and simulation results verifies the feasibility and superiority of proposed TACO.

Acknowledgments

This work was supported by the State Key Laboratory of Massive Personalized Customization System and Technology under grant No. H&C-MPC-2023-04-01, and Postgraduate Research & Practice Innovation Program of NUAA under grant No. xcxjh20231601.

References

[1] S. D. Okegbile, J. Cai *et al.*, "Human digital twin for personalized healthcare: Vision, architecture and future directions," *IEEE Netw.*, 2022.

[2] J. Chen, C. Yi *et al.*, "Networking architecture and key supporting technologies for human digital twin in personalized healthcare: A comprehensive survey," *IEEE Commun. Surv. Tutor.*, 2023.

[3] Y. Shi, Y. Yang, C. Yi, B. Chen, and J. Cai, "Towards online reliability-enhanced microservice deployment with layer sharing in edge computing," *IEEE Internet Things J.*, 2024.

[4] M. Lauer-Schmaltz, P. Cash *et al.*, "Designing human digital twins for behaviour-changing therapy and rehabilitation: a systematic review," *Proc. Design Soc.*, vol. 2, pp. 1303–1312, 2022.

[5] Y. Shi, C. Yi, R. Wang, Q. Wu, B. Chen, and J. Cai, "Service migration or task rerouting: A two-timescale online resource optimization for MEC," *IEEE Trans. Wirel. Commun.*, 2023.

[6] W. Wu, P. Yang *et al.*, "Accuracy-guaranteed collaborative DNN inference in industrial IoT via deep reinforcement learning," *IEEE Trans. Ind. Inform.*, vol. 17, no. 7, pp. 4988–4998, 2020.

[7] M. Neely, *Stochastic network optimization with application to communication and queueing systems*. Springer Nature, 2022.

[8] P. M. Castro, "Tightening piecewise mccormick relaxations for bilinear problems," *Comput. Chem. Eng.*, vol. 72, pp. 300–311, 2015.

[9] R. Karuppiah and I. E. Grossmann, "Global optimization for the synthesis of integrated water systems in chemical processes," *Comput. Chem. Eng.*, vol. 30, no. 4, pp. 650–673, 2006.

[10] C. Yi, J. Cai *et al.*, "A queueing game based management framework for fog computing with strategic computing speed control," *IEEE Trans. Mob.*, vol. 21, no. 5, pp. 1537–1551, 2020.

[11] X. Lin, J. Wu, J. Li, W. Yang, and M. Guizani, "Stochastic digital-twin service demand with edge response: An incentive-based congestion control approach," *IEEE Trans. Mob.*, 2021.

[12] Y. Zhang, H. Zhang, Y. Lu, W. Sun, L. Wei, Y. Zhang, and B. Wang, "Adaptive digital twin placement and transfer in wireless computing power network," *IEEE Internet Things J.*, pp. 1–1, 2023.