Proactive Task Allocation in Extreme Edge Computing for Digital Twin Services

Rawan F. El-Khatib, *Graduate Student Member, IEEE,* Sara A. Elsayed, *Member, IEEE,* Nizar Zorba, *Senior Member, IEEE,* and Hossam S. Hassanein, *Fellow, IEEE,*

***Abstract*—Extreme Edge Computing (EEC) exploits the untapped computational power of end devices, referred to as Extreme Edge Devices (EEDs), and thus holds the potential to revolutionize the Digital Twin (DT) technology. However, traditional reactive task allocation approaches fail to address the complexities of DT processing tasks, where the execution of all the underlying subtasks is crucial. Additionally, these approaches suffer due to the intermittent availability of EEDs, which compromises the Quality of Service (QoS). In this paper, we propose the Proactive Maximum Weighted Service Capacity (P-MWSC) scheme. P-MWSC is the first scheme to employ a proactive approach, utilizing predictions of the dynamic resource usage and resource characterization of EEDs to tackle the intricacies of DT tasks while taming the effects of EEDs' dynamicity and intermittent availability. We formulate the task allocation problem as a Binary Integer Linear Program (BILP) that aims to maximize the service capacity, weighted by the achieved gain from each fully assigned task. We derive an analytical solution using the Karush–Kuhn–Tucker (KKT) conditions and Lagrangian relaxation, and use a top-down decomposition approach to provide a solution that achieves up to 80% runtime reduction. Additionally, we propose a heuristic scheme with a bottom-up decomposition approach that is suitable for certain practical scenarios, yielding up to 90% runtime reduction. Extensive performance evaluations using data from a realistic testbed demonstrate that P-MWSC outperforms representatives of prominent reactive and proactive schemes, achieving up to 70% increase in the task success rate and a 39% reduction in the average response delay.**

I. Introduction

The long-anticipated convergence of the physical and digital worlds is becoming a tangible reality, driven by emerging technologies that seamlessly integrate these systems and pave the way for comprehensive digital transformation. At the forefront of these technologies is the Digital Twin (DT) technology, which is projected to play a foundational role in reshaping industries, societies, and governments [1], [2].

DTs are virtual replicas that elaborately model elements of a physical system such as users, devices, wireless channels, and services based on historical and real-time data [3], [4]. By mimicking the behaviors and characteristics of their physical counterparts in a digital environment, DTs facilitate better decision-making, monitoring, analysis, and optimization [5], [6]. For instance, in a video streaming service, DTs can predict link disruptions and identify to video portions to be prefetched to ensure seamless service [7]. Similarly, DTs predict Line-of-Sight (LoS) blockages for beam-switching [8] and simulate multi-spectrum radio propagation for communication and sensing [9]. Achieving these DT tasks requires handling multi-modal data streams generated by distributed infrastructure and user-owned devices. Consequently, DT-assisted decision-making results in complex processing tasks that involve several resource-intensive computations over streams of sensory data from multiple sources.

Within the context of the complex and resource-intensive computations associated with DTs, it is imperative to consider that incomplete execution of DT tasks due to one or more failures in processing data streams may lead to inaccurate DT modeling, hinder real-time analytics, and compromise the reliability of decision-making. Thus, it is crucial to complete all the underlying subtasks of each DT task while abiding by their delay constraints. Such requirements necessitate not only pushing the computational resources closer to data sources but also managing multiple parallel subtasks to ensure successful service delivery, which requires unique computing solutions and paradigms. Extreme Edge Computing (EEC) is a computing paradigm that opportunistically utilizes the ample idle computational resources at the Extreme Edge [10]. EEC enhances Edge Computing capabilities with the resources of Extreme Edge Devices (EEDs) such as smartphones and laptops. Note that a modern EED (i.e., worker)[1] is equipped with powerful computing units and diverse wireless communication capabilities, rendering it capable of exchanging and computing data in real-time. Recent reports indicate that most battery-powered EEDs have more than 55% of their battery available and over two-thirds of their computing power unoccupied by local tasks [11]. Thus, EEC is an attractive computing paradigm that offsets the excessive infrastructure costs of EC and disrupts the monopoly that cloud service providers and network operators/telecommunications companies have over edge services [12]. Additionally, parallel processing of the task partitions of a DT-based service at multiple EEDs can boost computational power and provide computing services in closer proximity to end-users, drastically reducing latency. Note that harvesting such advantages of EEC requires making efficient task allocation decisions that satisfy the sophisticated requirements of DT-based services.

R. F. El-Khatib is with the Department of Electrical and Computer Engineering, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: rawan.elkhatib@queensu.ca).

S. A. Elsayed, and H. S. Hassanein are with the School of Computing, Queen's University, Kingston, ON K7L 2N8, Canada (e-mail: selsayed@cs.queensu.ca, hossam@cs.queensu.ca).

N. Zorba is with the College of Engineering, Qatar University, Doha, Qatar (e-mail: nizarz@qu.edu.qa).

[1]In this work, we use the terms EEDs and workers interchangeably.

Several schemes focused on task allocation in EC and EEC (e.g., [13]–[17]). However, existing schemes overlook the sophisticated requirements of DT-based services, which demand complete task execution [13]–[17]. Failing to execute a single subtask renders the entire parent task incomplete, reducing the task success rate and increasing the risk of wasted resources, since the computational resources allocated to successfully completed subtasks would be wasted if the parent task failed. This compromises the Quality of Service (QoS) and service requester satisfaction, as incomplete tasks hinder reliable decision-making. Consequently, existing schemes fall short when dealing with the unique requirements of DT-based services. This problem is further exacerbated by the high dynamicity and intermittent availability of EEDs, which render the reactive approach adopted by existing task allocation schemes inadequate. Since EEDs are user-owned devices, they are highly susceptible to intermittent availability, particularly due to their dynamic user access behavior [18]. For instance, users may abruptly start to locally use their own devices (EEDs) for a computationally intensive application, which can drastically alter the availability of computational resources to offloaded tasks, reducing the task success rate and increasing the response delay. This intermittent availability and dynamic user access behavior presents significant challenges for reactive schemes, particularly in contexts where tasks are divided into subtasks.

In reactive schemes, task allocation decisions are made based on the current system state [13]–[17]. Such schemes struggle to ensure full task execution due to their limited knowledge of the available resources that are dynamically changing. In order to fully execute a task, the orchestrator must confirm that sufficient workers are available within the current execution window. If resources are insufficient, the orchestrator faces two options: either to drop the entire task, which undermines the task success rate and service requester satisfaction, or to assign a portion of the task in the current round and attempt to complete it in subsequent rounds, which results in increased response delays. If resource shortages arise in subsequent rounds, the task remains incomplete, leading to significant wasted resources, reduced task success rate, increased response delay, and compromised QoS. Evidently, these negative effects are significantly amplified when multiple DT service requesters contend for limited workers (i.e., EEDs).

To address the aforementioned challenges, we propose the Proactive Maximum Weighted Service Capacity (P-MWSC) scheme. P-MWSC makes proactive task allocation decisions that ensure full task completion by incorporating predictions of future worker availability into the decision-making process. Note that recent research on real datasets and real testbeds has demonstrated the ability to yield high prediction accuracy in forecasting the dynamic resource usage and intermittent availability of workers under a highly dynamic user access behavior, providing reliable multi-step ahead predictions of the computational capabilities of workers [19]. Additionally, research on a real dataset capturing the dynamicity of workers in EEC has shown promising results in predicting worker availability [20]. P-MWSC leverages such predictions to account for the dynamicity and intermittent availability of workers and addresses the sophisticated nature of DT-based services by proactively making informed task allocation decision. We formulate the task allocation problem in P-MWSC as a Binary Integer Linear Program (BILP) and demonstrate that its decision problem is NP-complete. We derive an analytical solution using the Karush–Kuhn–Tucker (KKT) conditions and Lagrangian analysis. Our contributions can be summarized as follows.

- We propose a novel task allocation scheme (P-MWSC) that addresses the intricate nature of DT processing tasks and their reliance on multi-modal data sourced from distributed data producers, while considering the dynamic and intermittent availability of EEDs. P-MWSC is the first scheme that proactively allocates tasks to EEDs to account for the complex requirements of DT services, including the need to ensure the successful execution of all partitioned subtasks comprising each DT task, while adhering to their deadline requirements. P-MWSC aims to optimize service capacity, weighted by the profit procured from each fully assigned job, all while adhering to the tasks' deadline constraints, thus maintaining high QoS and performance standards.
- We reduce the complexity of the P-MWSC task allocation problem and propose a time-efficient solution. We achieve this by exploiting the bi-level hierarchical structure and applying the Logic-Based Benders Decomposition (LBBD) [21], [22] algorithm. The problem is decomposed into a relaxed master problem with reduced constraints and a slave problem that generates cuts to iteratively refine the master problem's search space. Ultimately, the master and slave problems converge towards the optimal solution in a time-efficient manner.
- We propose a new heuristic to further reduce the complexity of P-MWSC task allocation problem. The proposed heuristic retains high task success rates with low computational complexity solutions, making it suitable for practical system scenarios.

We use publicly available data from a realistic testbed representative of heterogeneous workers in different dynamic resource usage scenarios [23]. Extensive performance evaluations show that P-MWSC yields significant improvements, achieving up to an improvement of up to 48% and 70% in the task success rate, and 33% and 39% reduction in the average response delay compared to prominent representatives of reactive and proactive task allocation schemes, respectively. This reflects its ability to maintain high QoS and performance standards. Additionally, rigorous performance evaluations demonstrate that P-MWSC still manages to perform well under erroneous predictions.

The remainder of the paper is organized as follows. Section II reviews the related work. Section III presents the system model and Section IV introduces the proposed scheme (P-MWSC) and the exact top-down decomposition based approach. Section V introduces the heuristic bottom-up decomposition solution.

Section VI details our performance evaluation experiments. Section VIII summarizes our work and our conclusions.

II. Related Work

Task allocation in EEC is studied under various scenarios with different enabling technologies [24]. The main optimization goals are to a) minimize latency (e.g., [13]), b) minimize energy consumption (e.g., [14], [25]), and variations that examine trade-offs between these two objectives or their derivatives (e.g., [15]–[17], [26]). In [13], Liang et al. propose a Mixed Integer Non-Linear Program (MINLP) to minimize service requester latency under energy constraints, using data compression for bandwidth efficiency, and solve it with a near-optimal ADMM-based distributed algorithm. Khazali et al. [14] focus on minimizing energy consumption by jointly optimizing task assignment, power allocation, and node grouping. Fang et al. [15] optimize communication and computing resource allocation to maximize offloading benefits via game theory. Yuan et al. [16] develop a collaborative task and resource allocation scheme to maximize profit while meeting task deadlines, considering CPU utilization, service rate, and power usage. Xie et al. [17] frame the network utility maximization problem as a potential game, proving the existence of a Nash equilibrium and introducing two learning-based algorithms to find it in a distributed manner.

These studies can be categorized into: a) systems with a single requester handling one task [14] or multiple independent tasks [17], and b) systems with multiple requesters, each with a single task [13], [15], [16], [26]. On one hand, schemes designed for single-service requesters often struggle to manage the competition in multi-requester scenarios while maintaining QoS levels. Extending these results to multi-requester environments can be challenging and may lead to sub-optimal performance. Additionally, studies that focus on individual tasks overlook the cohesion required among subtasks from the same requester, which is crucial for complex DT-based services. Therefore, these approaches fall short in meeting the needs of complex DT-based services, where requester satisfaction relies on fulfilling both individual subtask requirements and the complete execution of the parent task. In contrast to these works, we propose a task allocation scheme (P-MWSC) that considers a multi-requester system where each requester has a complex task that comprises several subtasks. P-MWSC takes into account the individual subtask requirements and overall service requester satisfaction by ensuring the completion of the entire parent tasks. Hence, P-MWSC is suitable for DT-based services where full task execution is critical for accurate modeling and decision-making.

The existing literature is predominantly based on the reactive task allocation approach [13]–[17], where allocation decisions are made based on the current system state, often resulting in increased delays. In addition, reactive approaches operate under the assumption that the workers are static and exist in a uniform computing platform. As mentioned earlier, this assumption is unrealistic in the EEC paradigm where workers are subject to their owners dynamic access behavior. Several recent works established that EEDs have predictable patterns that can be leveraged to improve task allocation policies [20], [23], [27]. Such predictors equip the orchestrator with *a priori* knowledge about time and resource availability of candidate EEDs, allowing it to overcome the drawbacks of reactive approaches by making proactive task allocation decisions.

Few studies have explored proactive task allocation in EEC. El Khatib et al. [10] propose a scheme that uses predictable task request patterns to preemptively map tasks to workers aiming to minimize the latency under workload constraints. Similarly, Liu et al. [26] introduce a proactive scheme for Vehicular Edge Computing (VEC) that incorporates predictions of vehicle speeds to adjust task density and improve system stability. However, such schemes serve static EEDs [10], [26] whose availability, resource usage, and computational capabilities are assumed to be fixed. Consequently, they fail to account for the dynamic availability and the fluctuating resource usage of EEDs. Additionally, they fail to ensure full task completion [26], which renders them impractical for the unique requirements of DT-based services. In contrast, our scheme accounts for the dynamicity of EEDs by incorporating predictions of the dynamic resource usage and intermittent availability of workers under highly dynamic user access behavior. Such predictions allows to proactively allocate tasks to EEDs to meet the intricate demands of DT services while maximizing the number of satisfied requests and ensuring the successful execution of all partitioned subtasks that make up each DT task. To the best of our knowledge, this is the first study to tackle the complexities of processing tasks arising from DT service requests in a dynamic EED environment. We introduce P-MWSC to ensure the success of DT tasks by executing all subtasks within the relevant performance constraints. By leveraging the predictability of EED dynamics, P-MWSC employs a proactive task allocation approach to form groups of EEDs, where each group collaborates to arrive at a decision for a DT-aided service.

III. System Model

Consider a set of K service requesters $\mathcal{S} = \{s_1, s_2, ..., s_K\}$, a set of J workers $\mathcal{W} = \{w_1, w_2, ..., w_J\}$, and a centralized entity, known as the orchestrator. The orchestrator, which acts as the scheduler, is hosted at an edge server or an Access Point (AP) that can exchange information with both the service requesters and workers, with the aim of creating appropriate subtask-worker allocation decisions.

Each service requester $s_k \in \mathcal{S}$ issues one DT-based service request. Based on the DT service, several data measurements from distributed sensors must be processed to make a decision, where each data measurement corresponds to a separate subtask. Service requesters submit details about the nature of these subtasks and the computational resources needed to the orchestrator, whereas workers lease out their computational resources to perform these subtasks in return for incentives. The orchestrator utilizes a prediction module to forecast the future availability of workers, including their available time periods and computational power. Note that dynamic worker availability prediction techniques have been shown to achieve a

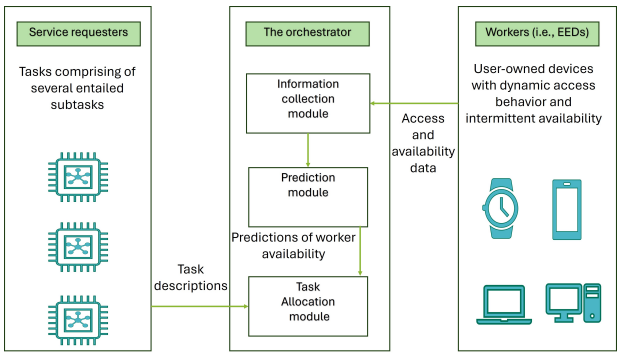

Fig. 1: The system model.

high prediction accuracy [20]. Additionally, it has been demonstrated that multi-step ahead prediction of dynamic resource usage in workers can achieve high prediction accuracy and facilitate efficient resource characterization [19]. We use such predictors to obtain this a priori knowledge. In order to ensure full DT task execution, all entailed subtasks must be executed. The orchestrator achieves this by assigning the subtasks to possibly different workers for parallel execution. Particularly, the orchestrator aims to create groups of collaborating workers, where each group is delegated the task corresponding to one DT service. Note that in P-MWSC, we do not enforce mutually exclusive groups. In other words, a worker can be delegated more than one subtask, where each subtask comes from a separate service requester, and hence groups of collaborating EEDs can be overlapping. Without loss of generality, we assume that the aforementioned predicted information is provided by the prediction module for a specified time window, during which the workers' dynamics are known. As a result, time is divided into sequential epochs $\{e_1, e_2, e_3, ..\}$, with each epoch corresponding to this time window. Before each epoch begins, the orchestrator proactively makes task allocation decisions based on the predictions and the available information about the workers. The system model is depicted in Fig. 1.

Let $\mathcal{T} = \{t_1, t_2, ..., t_I\}$ denote the set of all I subtasks in the system. Each service requester s_k issues a task request that comprises a batch of subtasks denoted by the set $\mathcal{B}_k$, resulting in a finite number $K < I$ of disjoint sets, where $\mathcal{B}_1 \cup \mathcal{B}_2 ... \cup \mathcal{B}_k = \mathcal{T}$. Each batch of subtasks is executable in parallel (i.e., no order is required), and each subtask t_i cannot be further divided. The profile of subtask t_i is characterized by a six-tuple of parameters: $\langle L_i^{in}, L_i^{out}, q_i, \delta_i^{max}, c_i, g_i \rangle$. In particular, L_i^{in} (in bits) specifies the amount of input data to be processed, L_i^{out} (in bits) specifies the amount of resulting data after processing the subtask, q_i (in cycles/bit) denotes the amount of computational intensity of t_i, and δ_i^{max} denotes the deadline within which t_i must be completed. Each subtask has a computational requirement c_i, specifying the number of CPU cycles per second that a worker must dedicate to execute subtask t_i. This value is set by service requesters to mitigate the effects of the dynamic user access behavior of workers. In P-MWSC, a contract is negotiated between a worker w_j and a service requester s_k regarding subtask t_i. Once the worker agrees to the contract, then the predetermined amount of its computational capacity c_i is locked and dedicated to subtask t_i, in exchange for a monetary gain denoted by g_i.

P-MWSC follows an all-or-nothing remuneration approach, where the gain is associated with full task execution, i.e., the reward of a task is paid if and only if all its subtasks are executed. Consequently, for each task fully executed, the amount of gain g_k achieved by the group is equivalent to the sum of gains of all subtasks issued by the service requester s_k.

A worker w_j has a single fast processor where all its processing power is concentrated. The problem can be easily extended to the case where multiple processors exist. The processor is equipped with Dynamic Voltage Frequency Scaling (DVFS) capability, and can adjust the device's operating frequency according to the subtask's computational demand c_i. Let α_j be the instant of time at which worker w_j becomes idle, allowing it to take on some subtasks. The maximum CPU capacity available for the worker w_j is indicated by f_j^{max}. In alignment with distributed model of data producers, let $d_{i,j}$ and $r_{i,j}$ be the distance and the data rate of the link between the data producer of task t_i and worker w_j respectively. Let v denote the speed of light. Then, the total response delay for the subtask-worker tuple $\langle i, j \rangle$ is expressed in Eq. 1.

$$D_{i,j}^{total} = \alpha_j + \frac{L_i^{in} q_i}{c_i} + 2\frac{d_{i,j}}{v} + \frac{L_i^{in} + L_i^{out}}{r_{i,j}} \tag{1}$$

The four terms correspond to the waiting delay for worker w_j, the execution delay, the propagation delay and the transmission delay, respectively.

IV. Proactive Maximum Weighted Service Capacity (P-MWSC)

We formulate the problem as a BILP. Let $\mathbf{X}$ denote the $I \times J$ matrix of the binary decision variables $x_{i,j}, \forall i, \forall j$ defined in Eq. 2.

$$x_{i,j} = \begin{cases} 1, & \text{if subtask } t_i \text{ is assigned to worker } w_j \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Additionally, let $\mathbf{y}$ denote the K-dimensional vector of the binary decision variables $y_k, \forall k$, defined by Eq. 3.

$$y_k = \begin{cases} 1, & \text{if task } k \text{ is selected for full assignment} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

From the perspective of service requesters, it is important to ensure that the maximum feasible number of service requests is fulfilled. On the other hand, workers are interested in maximizing their remunerations. Hence, P-MWSC aims to maximize the service capacity (i.e., the number of fulfilled service requests) weighted by their respective gains. This not only promotes high QoS for service requesters but also ensures the most profitable tasks are selected, which increases the average reward per worker. Note that the effects of satisfactory profits have been shown to be key in maintaining workers continuous participation and coping with the load imposed by

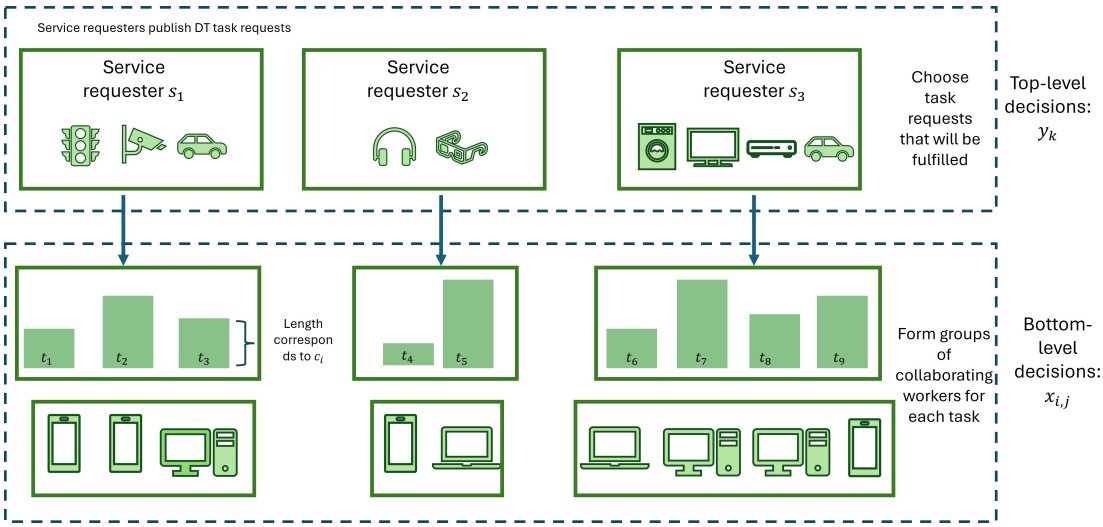

Fig. 2: The bi-level hierarchy of the P-MWSC problem.

incoming requests [28]. The problem formulation is shown in Eq. (4).

$$\textbf{(P1):} \quad \underset{\mathbf{X},\mathbf{y}}{\text{maximize}} \quad \sum_{\forall k} g_k y_k \tag{4a}$$

$$\text{subject to}$$

$$\sum_{\forall i} c_i x_{i,j} \leq f_j^{max} \quad ,\forall j, \tag{4b}$$

$$\sum_{\forall j} D_{i,j} x_{i,j} \leq \delta_i^{max} \quad ,\forall i, \tag{4c}$$

$$\sum_{\forall j} x_{i,j} = y_k \quad ,\forall i \in \mathcal{B}_k, \forall k, \tag{4d}$$

$$x_{i,j}, y_k \in \{0,1\} \tag{4e}$$

The objective is to maximize service capacity, weighted by the collected gain from each fully assigned task. Constraint 4b ensures that each subtask fulfills exactly its required computational capacity c_i, and that for each worker w_j, the cumulative computational loads of the subtasks assigned to it do not exceed its maximum available CPU capacity. Constraint 4c enforces the timeliness requirements of individual subtasks, by ensuring that the total response delay of any subtask-worker tuple $\langle i, j \rangle$ does not exceed the maximum completion deadline of t_i. This is crucial because many subtasks are highly sensitive to latency, and processing subtasks beyond their deadlines renders them meaningless for requesters. Lastly, to enforce task requirements, constraint 4d ensures that when task k is selected for processing, all its entailed subtasks must be assigned. This constraint also ensures that a subtask is assigned to at most one worker.

Clearly, **P1** is a BILP, which is well-known to be challenging to solve [29]. In fact, **P1** can be seen as an instance of the Multiple Knapsack Problem with Assignment Restrictions (MKAR), whose decision problem is known to be $\mathcal{NP}$-complete [30]. Note that the inequality constraints in (4b) and (4c) are both convex, and the equality constraint in (4d) is affine. However, the integrality constraints in (4e) are discrete and non-convex. As such, exact optimization techniques (e.g., exact BILP solvers) would be computationally prohibitive for large-scale instances [29]. To overcome this limitation, we convexify **P1** by replacing (2) and (3) with $0 \leq x_{i,j}, y_k \leq 1, \forall i, j, k$, making an analytical solution attainable. We use Lagrangian multipliers and Karush–Kuhn–Tucker (KKT) conditions to simplify the problems's complexity and reduce computational burdens [31]. Given that closed-form solutions for BILPs are generally not achievable, we derive a lower bound for the optimal values of $x_{i,j}$ and y_k.

Theorem 1. *The lower bound on the optimal value of $x_{i,j}$ which is responsible for assigning subtask i to worker j is written as:*

$$x_{i,j}^* = \begin{cases} 0 & ,u = 0 \\ 1 & ,-u = \lambda_{3,k,i}^{*(1)} - \lambda_{3,k,i}^{*(2)} \\ \frac{f_j^{max} - \sum_{\hat{i} \neq i} c_{\hat{i}} x_{\hat{i},j}}{c_i} & ,\lambda_{1,j}^* > 0 \\ \frac{\delta_i^{max} - \sum_{\hat{j} \neq j} D_{i,\hat{j}} x_{i,\hat{j}}^*}{D_{i,j}} & ,\lambda_{2,i}^* > 0 \\ y_k^* - \sum_{\hat{j} \neq j} x_{i,\hat{j}}^* & ,\lambda_{3,k,i}^{*(1)} > 0 \\ y_k^* - \sum_{\hat{j} \neq j} x_{i,\hat{j}}^* & ,\lambda_{3,k,i}^{*(2)} > 0 \end{cases} \tag{5}$$

Moreover, the lower bound on the optimal value of y_k which is responsible for selecting tasks is written as:

$$y_k^* = \begin{cases} 0, & \lambda_{5,k}^{*(1)} = 0 \\ 1, & \lambda_{5,k}^{*(1)} > 0, \\ \sum_j x_{i,j}^*, & \lambda_{3,k,i}^{*(1)} > 0 \\ \sum_j x_{i,j}^*, & \lambda_{3,k,i}^{*(2)} > 0 \end{cases} \quad g_k = \lambda_{3,k,i}^{*(1)} - \lambda_{3,k,i}^{*(2)} - \lambda_{5,k}^{*(1)} \tag{6}$$

Proof. The proof of Theorem 1 is given in the Appendix. □

According to the bounds obtained in Theorem 1, we conclude that the closed-form solution can not be attained because the bounds on the decision variables $\mathbf{X}$ and $\mathbf{y}$ are expressed in terms of other unknown variables. In the next subsection, we derive a decomposition-based solution.

A. Top-down Decomposition (TDD) of P-MWSC

The $\mathcal{NP}$-completenes of the decision problem **P1** is indicative of its inherent computational complexity and the difficulty of finding optimal solutions. Therefore, it is imperative to develop approaches that accelerate its solution. To achieve this, we exploit the hierarchy of P-MWSC, as shown in Fig. 2. Specifically, the optimization problem in P-MWSC involves two key decisions that can be envisioned in two layers: a) in the top layer, we decide which set of tasks are selected for full processing, and b) in the bottom layer, we decide how the batches of subtasks corresponding to the selected tasks are allocated to workers. This bi-level structure arises from P-MWSC's requirement for full task execution, which enforces a hierarchical relationship between the decision variables y_k and $x_{i,j}$.

The primary logic behind hierarchical decomposition is to handle complexity by structuring the problem into a sequence of interrelated subproblems, where each subproblem's solution informs and constrains the next. By employing this divide-and-conquer strategy, P-MWSC can be decomposed into two sub-problems, each corresponding to a different hierarchical level which allows for a more granular understanding of the problem space. Final solutions are obtained by integrating the partial solutions through a specified integration scheme. The complexity of the decomposed sub-problems and the optimality of the final solution are contingent upon the decomposition and integration methodologies employed. In this section, we present the top-down decomposition approach based on the Logic-Based Benders' Decomposition (LBBD) [21], [22].

The LBBD is an exact decomposition algorithm that decouples a complex optimization problem into a sequence of two sub-problems: the master problem and the slave problem. The master problem is derived by relaxing or omitting certain constraints from the original formulation, making its solution space smaller than that of the original problem and thus it becomes easier to tackle [21], [22]. The slave problem is informed by the master problem, and its solution allows to generate cuts that guide the master problem in its search space. The iterative LBBD algorithm has been shown to converge towards the optimal solution [21], [22]. The LBBD has been recently used in several works within the EC paradigm to circumvent the complexities of various $\mathcal{NP}$-hard problems and has been shown to provide satisfactory results [32], [33].

We apply the LBBD algorithm to P-MWSC as follows. In P-MWSC, the master problem is formulated to encompass the top-level decision variables $\mathbf{y}$. Specifically, all workers' individual capacities are aggregated into one fictitious 'meta' worker whose maximum available computing capacity is equal to the sum of individual capacities of all workers in $\mathcal{W}$. Let F_{max} denote the maximum available computing capacity in the system entirely: $F_{max} = \sum_j f_j^{max}$. Similarly, the computational requirements of individual subtasks in each batch $\mathcal{B}_k$ are consolidated into a single value C_k equivalent to the sum of the individual c_i values. Consequently, the Top Down Decomposition (TDD) master problem is written in 7.

$$\textbf{(P2.1):} \quad \underset{\mathbf{y}}{\text{maximize}} \quad \sum_{\forall k} g_k y_k \tag{7a}$$

$$\text{subject to}$$

$$\sum_{\forall k} c_k y_k \leq F_{max} \quad , \tag{7b}$$

$$y_k \in \{0,1\} \tag{7c}$$

In this manner, the top-level master problem aims to identify the set of tasks that maximizes the weighted service capacity, taking into account the system's computational capabilities. This problem represents a relaxation of **P1** as it does not account for the timeliness requirements of individual subtasks or the capacity constraints of individual workers. Consequently, the TDD master problem is a simplified version of **P1** and is easier to solve. Let the solution to the master problem at iteration h be denoted by $\mathbf{y}_h^M$. Note that the first time we solve **P2.1**, we obtain the upper bound of the original problem **P1**. This tentative solution is then used as input for the slave problem, which corresponds to the original problem formulated with the task selections determined by $\mathbf{y}_h^M$. Since the objective function solely depends on the gains of the selected tasks (i.e., top-level decision variables $\mathbf{y}$), it remains unaffected by the subtask-worker allocations decisions (i.e., bottom-level decision variables $x_{i,j}$). Consequently, the same objective function can be employed for both the master problem and the sub-problem. Once the collection of tasks has been determined, the slave problem aims to determine if there exists a feasible subtask-worker allocation decisions such that all the subtasks within the selected batches are assigned, ensuring that subtask timeliness requirements and individual worker capacities are satisfied. This problem is written in 8.

$$\textbf{(P2.2)}: \underset{\mathbf{X}}{\text{maximize}} \quad \sum_{\forall k} g_k y_k \tag{8a}$$

$$\text{subject to}$$

$$(4b), (4c), (4d), \tag{8b}$$

$$y_k = y_{h,k}^{M*} \quad , \forall k, \tag{8c}$$

$$x_{i,j} \in \{0,1\} \tag{8d}$$

The advantage of the LBBD approach lies in fixing the top-level decision variables $\mathbf{y}$, thereby significantly reducing the search space of the slave problem. Let $\mathbf{X}_h^S$ denote the slave problem solution at iteration h. There are two possible outcomes to attempting **P2.2**. The first outcome occurs if the slave problem yields a feasible solution, in which case the current solution is a global optimum, and the algorithm terminates. Alternatively, if the slave problem returns an infeasible solution, a Benders' cut, denoted by ψ_h, must be introduced to guide **P2.1** in its search in the next iteration $h+1$, such that it

excludes the task(s) causing the infeasibility. This is achieved by appending an additional constraint to **P2.1**. We formulate the cut by pinpointing the smallest subset of tasks within the current solution that triggers infeasibility [34]. Let $\mathcal{S}_h \subset \mathcal{S}$ be the subset of tasks chosen in $\mathbf{y}_h^M$. We arrange tasks in ascending order by their computational requirements and systematically remove the least demanding task from this set. The smallest subset of tasks that induces infeasibility in the slave problem is used to derive a cut as follows. Let this subset be denoted by $\mathcal{S}_h^{'}$, then the Benders' cut ψ_h is defined in Eq. 9.

$$\psi_h = \sum_{k \in \mathcal{S}_h^{'}} y_k \leq |\mathcal{S}_h^{'}| - 1 \tag{9}$$

By adding this cut, we exclude this combination of tasks and any superset of it in $\mathbf{y}_{(h+1)}^M$. As the cuts accumulate, the search space will become tighter, and the cuts more constructive, eventually converging towards the optimal solution [21], [22]. A formal statement of the TDD procedure appears in Algorithm 1.

Algorithm 1 The P-MWSC TDD Algorithm

Input: $\mathcal{T}, \mathcal{B}_k, \langle L_i^{in}, L_i^{out}, q_i, \delta_i^{max}, c_i, g_i \rangle, \forall i, \langle f_j^{max}, \alpha_j \rangle, \forall j, D_{i,j}, \forall i, \forall j$
Output: $\{\mathbf{X}, \mathbf{y}\}$

```
h ← 0
feasible ← 0
repeat
    y_h^M ← solve(P2.1)
    X_h^S ← solve(P2.2 ∧ y_k = y_{h,k}^M)
    if P2.2 is infeasible then
        generate ψ_h from Eq. 9
        (P2.1) ← (P2.1) ∧ ψ_h
        h ← h + 1
    else if P2.2 is feasible then
        y ← y_h^M
        X ← X_h^M
    end if
until feasible = 1
```

Theorem 2. *Algorithm 1 terminates after a finite number of steps.*

Proof. The proof follows from [21]. The domains for the master problem decision variables in $\mathbf{y}$ are defined in binary tuples of $\{0, 1\}$, and hence, can generate only finitely many values. Consequently, Algorithm 1 will terminate in a finite number of steps. □

Clearly, the worst-case scenario is when the number of iterations is equal to the number of possible combinations of the set of tasks $K - 1$. This occurs if one task is selected in each iteration h and excluded in Eq. 9, until all the tasks are iterated. In practice, this is rarely the case. In fact, when a combination of tasks is excluded in Eq. 9, all the solutions in which these tasks are selected will be removed from the search space, regardless of the individual subtask-worker allocation decisions. Hence, given a fixed $\mathcal{S}_h^{'}$, the number of possible combinations of these tasks grows exponentially with its size, greatly reducing the number of iterations required to achieve feasibility. In the next section, we propose a near-optimal decomposition approach that traverses the P-MWSC hierarchy in an upward direction.

V. P-MWSC Heuristic using Bottom-Up Decomposition (BUD)

In this section, we propose an alternative solution that employs a Bottom-Up Decomposition (BUD) approach to address the P-MWSC problem. The fundamental principle of the BUD approach is to incrementally build from simple subtask assignments to tackle the complexities of task selections, beginning with the more granular decision-making at the lower level, where subtask-worker mappings are established based on individual subtask timeliness requirements and workers' capacity constraints. These subtask assignment decisions are made without regard to their contribution to a complete task assignment. Once the bottom-level decisions are made, they are used to construct the top-level final decisions for task selections. To this end, we introduce a two-stage approach: the first stage involves determining subtask-worker mappings, and the second stage involves making task selections by refining the solutions to comply with the full task selection requirement. The first stage BUD problem is formulated as follows:

$$\textbf{(P3.1):} \underset{\mathbf{X}}{\text{maximize}} \quad \sum_j \sum_i g_i x_{i,j} \tag{10a}$$

$$\text{subject to}$$

$$(4b), (4c), \tag{10b}$$

$$\sum_j x_{i,j} \leq 1, \forall i, \tag{10c}$$

$$x_{i,j} \in \{0, 1\} \tag{10d}$$

In **P3.1**, our objective is to maximize the total number of assigned subtasks weighted by their individual gains, subject to individual subtasks' timeliness requirements and individual workers' capacity limits. Constraint (10c) ensures that subtask t_i is assigned at most once. Note that **P3.1** is equivalent to an individual profit maximization problem without regard to the service requester's satisfaction with full task assignment. Let $x_{i,j}^{FS}$ denote the subtask-worker decisions determined in **P3.1**. This solution is sent to the second stage BUD problem to decide on task selections that maximize the weighted service capacity. Hence, the second stage BUD problem can be written as:

$$\textbf{(P3.2):} \underset{\mathbf{X}, \mathbf{y}}{\text{maximize}} \quad \sum_k g_k y_k \tag{11a}$$

$$\text{subject to}$$

$$(4d), \tag{11b}$$

$$\sum_j x_{i,j} - x_{i,j}^{FS} \leq 0, \forall i, \forall j, \tag{11c}$$

$$x_{i,j}, y_k \in \{0, 1\} \tag{11d}$$

In the formulation above, constraint (11c) states that a subtask that was allocated in the first stage can be either kept or removed. This constraint combined with constraint (4d) ensure that superfluous assignments that do not contribute to full task executions are removed from the final solution. Note that contrary to the TDD approach in Section IV-A, the BUD method only runs one iteration where the first and second stage problems are solved sequentially until an exact solution is achieved. Clearly, the resulting solution of P-MWSC-BUD is sub-optimal since it maximizes the weighted service capacity after the subtask-worker allocation decisions are made.

VI. Performance Evaluation

In this section, we evaluate the performance of P-MWSC in its TDD and BUD variants compared to two representatives of the state-of-the-art reactive task allocation approach. The first scheme, referred to as the Reactive Maximum Number of Tasks (RMNT) scheme, is representative of the prominent reactive strategy that aims to maximize the number of completed tasks and the associated profit [16]. At the beginning of each interval, RMNT assesses the system state and reactively allocates the tasks to workers with the aim of maximizing the number of assigned tasks under full task assignment and deadline constraints using the current information available about the workers, oblivious to any future changes. To ensure a fair comparison, we adapt RMNT to maximize the weighted service capacity while guaranteeing full task execution under time and workload constraints. The second scheme, referred to as the Reactive Minimum Response Delay (RMRD) scheme, aims to minimize latency [13]. Similar to RMNT, RMRD operates on a time-slotted basis but prioritizes minimizing the response delay in a greedy manner while adhering to workload constraints, without any regard to full task execution constraints. In addition, we compare P-MWSC to a representative of the state-of-the-art proactive task allocation schemes, referred to as the Proactive Static Minimum Number of Workers (P-SMNW) [10]. P-SMNW proactively assigns all available tasks to workers based on predicted computing capacities, subject to workload constraints. However, P-SMNW assumes a static environment and does not account for workers' intermittent availability. To ensure a fair comparison, we adapt P-SMNW to minimize the total number of recruited workers while guaranteeing all available tasks are assigned.

To corroborate the viability of our bi-level decomposition approach, we measure the runtime speedup of the TDD and BUD approaches compared to the extensive solution. Moreover, to assess performance, we employ the following performance metrics: 1) the Task Success Rate (TSR), defined as the number of fully assigned tasks divided by the total number of tasks; 2) the Subtask Denial of Admission (SDoA) rate, defined as the number of assigned subtasks divided by the total number of subtasks; 3) the average response delay, defined as the average time required to complete an assigned subtask; and 4) the reneging percentage, defined as the percentage of subtasks dropped due to deadline expiry.

A. Simulation Setup

P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD and P-SMNW are all implemented using MATLAB on an Intel i5-8250U at 1.6GHz. Similar to [35], our experiments are carried out in an area of 200m × 200m, within which service requesters and workers are uniformly distributed. The length of the execution epoch, which serves as the planning horizon for proactive schemes, is set to 300 seconds (i.e., 5 minutes) [27]. For reactive schemes (i.e., RMNT and RMRD), the 5 minute period is further divided into 1-minute intervals, with assignments made at the beginning of each interval. Each service requester issues one DT-based request whose aim is to predict future behaviors or test what-if scenarios pertaining to a certain phenomenon. The DT request is divided into several subtasks, each one of which requires processing of data obtained from a given data producer.

We use data from a realistic testbed that captures the dynamic use of worker resources [23]. The testbed utilizes a group of four Raspberry Pi 4B devices with CPU frequencies 1.2, 1.5, 1.5 and 1.8 GHz, respectively. To represent dynamic resource usage scenarios, a series of applications were executed sequentially while interspersed with idle periods. The applications include video gaming, YouTube video streaming, real-time augmented reality, and low-power cryptocurrency mining. Data was collected from various resource usage scenarios over 48-hour periods, with a monitoring interval of 5 seconds. The dataset provides detailed information that includes user CPU time, system CPU time, idle CPU time, memory usage percentage, network upload and download sizes, and rates, among other metrics. Given our focus on heterogeneous EEDs, we particularly emphasize the data capturing available CPU resources, which we leverage in our simulation experiments.

The dataset is preprocessed by aggregating CPU measurements to a 5-minute scale, aligning with the 5-minute epochs utilized for the proactive schemes. The number of workers in our setup ranges in $\{50, 75, ..., 150\}$, whose available computational frequency f_j^{max} is randomly sampled from the dataset, ensuring representation of the 4 different CPU capabilities. The arrival time of a worker α_j follows a Poisson process with $\lambda = 0.1$. The number of subtasks varies in $\{50, 75, ..., 150\}$, where each subtask has an input data size L_i^{in} uniformly distributed in $[200, 400] \times 10^3$ bits, and the output data size L_i^{out} is $0.05 \times L_i^{in}$. The computational intensity q_i and the computational requirement c_i are uniform in $[15, 25] \times 10^3$ cycles/bit and $[120, 180]$ MHz, respectively. The subtask gain g_i is set to $L_i^{in} \times q_i$, thereby allowing workers to earn higher gains for more demanding subtasks. Finally, the number of tasks varies in $\{4, 8, ..., 20\}$, with the number of subtasks per task randomly determined to ensure heterogeneity in the size of the tasks.

B. Simulation Results and Discussion

We evaluate the performance of our bi-level decomposition approach (P-MWSC-TDD and P-MWSC-BUD) against the extensive BILP solution in terms of runtime. All experiments are

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3524868

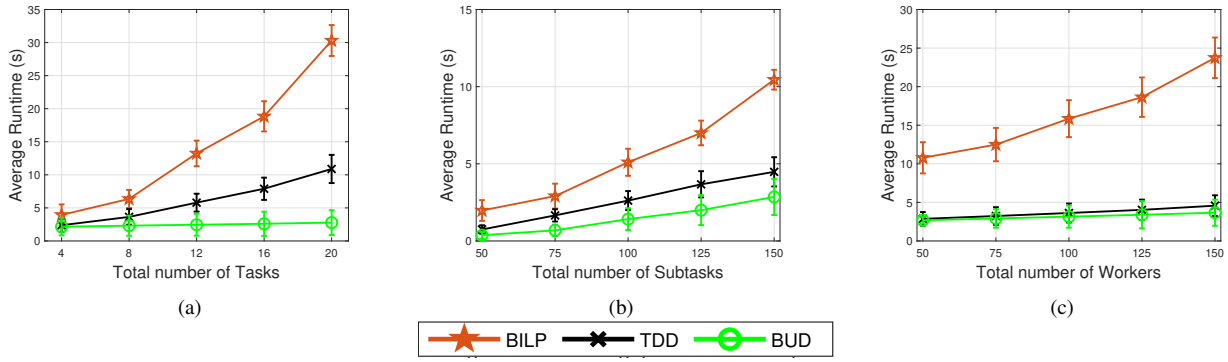

Fig. 3: Average runtime of BILP, TDD and BUD over varying numbers of tasks, subtasks, and workers, respectively.

repeated 50 times for each instance, and the simulation results are presented with a 95% confidence level. The confidence intervals are depicted in Figure 3. Due to their negligible size, confidence intervals are omitted from the other figures to enhance presentation clarity. Additionally, we evaluate the performance of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD and P-SMNW over varying numbers of subtasks, tasks, workers, and varying prediction error variance.

1- Runtime analysis.

Figure 3a depicts the average runtime when the number of subtasks and workers is 150 and 50, respectively, while the number of tasks varies. Recall that an increase in the number of tasks is equivalent to an increase in the number of requesters K. It can be seen that a growth in the number of tasks leads to longer runtimes for both the BILP extensive solution and P-MWSC-TDD. For the extensive solution, the runtime grows exponentially with more tasks, indicating that its execution scale does not perform well for larger instances and may incur runtimes that are unacceptable for practical implementations. In the case of P-MWSC-TDD, the growth in the number of tasks requires a larger number of iterations to converge to a feasible solution. P-MWSC-TDD renders a runtime reduction up to 64% compared to the BILP extensive solution due to its decomposition based approach. In contrast, P-MWSC-BUD shows minimal variation as the number of tasks grows. This stability is due to P-MWSC-BUD optimizing assignments in the first stage, where the search space size depends on the number of subtasks and workers, and does not account for tasks or their sizes. The second stage involves a simple pruning routine that requires minimal time. As such, P-MWSC-BUD renders a runtime speedup of up to 90% compared to the BILP extensive solution, and up to 74% runtime reduction when compared to the P-MWSC-TDD solution.

Figure 3b shows the average runtime when the number of tasks and workers is 10 and 50, respectively, while the number of subtasks varies. The figure reveals that the BILP extensive solution exhibits a rapid surge in runtime as the number of subtasks grows. P-MWSC-TDD consistently outperforms it, achieving up to 57% runtime reduction. This is because as the number of subtasks grows, the task sizes also increase, making Benders' cuts that exclude certain task combinations more effective in simplifying the slave problem, which in turn reduces the average runtime. P-MWSC-BUD also exhibits an upward trend because increasing the number of subtasks expands the search space of the first stage problem, albeit at a slower rate because subtask assignments are not complicated by the full task selections constraint. Accordingly, P-MWSC-BUD achieves up to a 72% and 36% runtime reduction compared to the BILP extensive solution and P-MWSC-TDD, respectively.

Lastly, Figure 3c illustrates the average runtime when the number of subtasks and tasks is 150 and 10, respectively, while the number of workers varies. Recall that an increase in the number of workers is equivalent to an increase in the number of EEDs. Similar to the previous figure, an increase in the number of workers also causes a significant surge in the average runtime for BILP extensive solution, for the same reasons discussed earlier. On the other hand, varying the number of workers has a modest effect on P-MWSC-TDD and P-MWSC-BUD compared to the number of subtasks, inducing an incremental rise in the runtime. In fact, P-MWSC yields up to 80% runtime reduction compared to the BILP extensive solution, while P-MWSC-BUD achieves up to 84% and 19% runtime reduction compared to the BILP extensive solution and P-MWSC-TDD, respectively. This is because additional subtasks introduce conflicting constraints related to full task assignment, computational requirements, and deadline constraints, which significantly complicate the problem. By closely examining the results in Figure 3, we observe that P-MWSC-TDD sustains a stable runtime, validating its scalability and performance advantage over the extensive solution as the number of tasks, subtasks, and workers increases rapidly. As we have thoroughly shown the superior performance of P-MWSC-TDD, we will focus now on analyzing its behavior within different scenarios.

2- Impact of Varying Number of Subtasks.

In this experiment, the number of workers is set to 50 and the number of tasks is set to 20, while the number of subtasks varies from 50 to 150. Note that the number of subtasks per task is random, and thus tasks are of varying sizes.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3524868

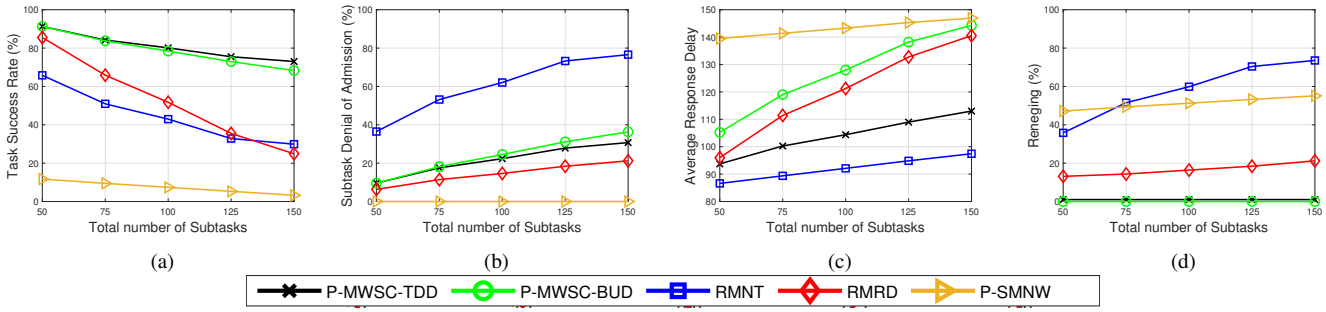

Fig. 4: Performance results of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD, and P-SMNW over varying numbers of subtasks.

This variability is common in practical systems, as DT service requesters may issue tasks with differing numbers of underlying subtasks based on their specific design. Figure 4a depicts the TSR of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD and P-SMNW over varying numbers of subtasks. It can be observed that all schemes exhibit a downward trend in the TSR as the number of subtasks grows. This is due to the growing task sizes coupled with a fixed number of workers, making it progressively more difficult to meet task requirements within specified deadlines. Both proactive schemes, P-MWSC-TDD and P-MWSC-BUD, demonstrate a significantly higher TSR compared to the reactive schemes RMNT and RMRD. Particularly, P-MWSC-TDD outperforms all other schemes due to its prioritization of maximizing the weighted service capacity, and its ability to exploit predictability through proactive subtask-worker allocation decisions, allowing it to fully capitalize on the future availability of EED resources. Consequently, P-MWSC-TDD boosts the TSR by 43% and 48% compared to RMNT and RMRD, respectively. P-MWSC-BUD ranks second in the achievable TSR, performing comparably to P-MWSC-TDD under low to medium loads. As the number of subtasks grows, the gap between P-MWSC-BUD and P-MWSC-TDD TSR reaches a maximum of approximately 5% at 150 subtasks. This is due to higher likelihood that P-MWSC-BUD selects subtasks that do not form complete tasks in the first stage, leading to significant pruning in the second stage. P-MWSC-BUD enhances the TSR by 38% and 43% compared to RMNT and RMRD, respectively.

Recall that, although RMNT solves the same optimization problem as MWSC, it operates reactively. RMNT assesses the system state at the beginning of each 1-minute interval to determine available workers and attempts to maximize service capacity based on this knowledge. Figure 4a shows that RMNT struggles to meet task requirements due to its reactive approach, which restricts knowledge about resource availability to the current state. On the other hand, P-SMNW performs poorly despite its proactive strategy that utilizes worker capacity predictions, as it overlooks the time-varying availability of these workers. As can be seen, P-MWSC-TDD and P-MWSC-BUD boost the TSR by up to 77.6% compared to P-SMNW. These findings support our assertion that merely ensuring full task execution is insufficient for the class of DT complex tasks that entail multiple subtasks. Proactive allocation that accounts for the intermittent availability and dynamic nature of EEDs is imperative to maintain acceptable levels of TSR for such tasks. Finally, RMRD assigns subtasks without consideration of whether their parent task is fully assigned, focusing instead on a minimum response delay objective. In very lightly loaded settings with fewer subtasks and sufficient workers, RMRD achieves up to an 85% TSR. However, a growth in the number of subtasks causes the TSR of RMRD dramatically declines, diminishing to approximately 25% when the number of subtasks is 150. This significant drop highlights its inability to manage complex task requirements under high-load conditions.

We conduct the same experiment to examine the SDoA in Figure 4b. Note that since tasks vary in size, the SDoA cannot be precisely assessed from the TSR metric. Recall that P-SMNW assigns all the existing tasks and entailed subtasks, and hence maintains a 0% SDoA. As can be seen in the figure, RMRD demonstrates lower SDoA rate compared to the other four schemes. This is attributed to RMRD's strategy of minimizing response delay until worker resources are exhausted, thereby allowing the admission of a substantial number of subtasks, albeit at the expense of optimal assignment. However, as illustrated in Figure 4a, this does not translate into a high TSR since it does not enforce the selection of full tasks. This suggests a lower level of satisfaction among DT service requesters even though a large percentage of subtasks are admitted. P-MWSC-TDD ranks second, decreasing the SDoA by up to 46% compared to RMNT. P-MWSC-BUD follows, improving the SDoA by up to 40% compared to RMNT. Note that P-MWSC-BUD exhibits a slightly higher SDoA compared to P-MWSC-TDD, reaching a maximum gap of 6% when the number of subtasks is 150. This is due to its two-stage process, where subtasks are initially selected and subsequently pruned. In contrast, RMNT shows a significantly higher SDoA because subtask admission is contingent on the complete assignment of tasks with limited foresight into available resources. This reactive approach limits its efficiency, particularly in heavily loaded scenarios, resulting in substantial SDoA rates, approaching 77% for 150 subtasks.

Figure 4c illustrates the average response delay across vary-

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3524868

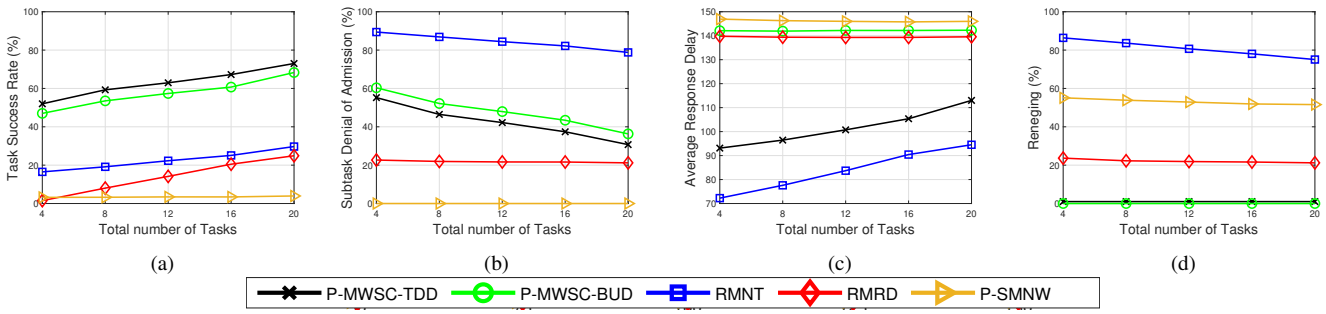

Fig. 5: Performance results of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD, and P-SMNW over varying numbers of tasks.

ing number of subtasks. All schemes exhibit an increasing trend in average response delay as the number of subtasks increases, primarily due to the accumulating waiting time caused by subtasks competing for the limited resources of the workers. P-MWSC-TDD provides the lower bound on the average response delay due to its proactive approach, which leverages foresight to minimize these delays effectively, reducing it by up to 18% and 33%, compared to RMRD and P-SMNW. P-MWSC-BUD exhibits a comparatively high average response delay compared to P-MWSC-TDD. The potential performance gains of the proactive strategy are hampered by its bottom-up approach, which involves identifying the largest number of feasible subtask-worker assignments before pruning them to ensure complete task assignments. This results in suboptimal subtask-worker allocation decisions with greater delays. P-SMNW underperforms in terms of the average response delay due to its assumption of static worker availability, leading to accumulated waiting delays when workers are unavailable to execute a subtask. RMRD also exhibits high average response delay because subtasks accumulate waiting delays in sequential 1-minute intervals until there are sufficient resources available for their execution. Although RMRD is capable of admitting a large number of subtasks, as shown in Figure 4b, the assignment is not optimal in terms of average response delays. RMNT operates similarly but only allows full task assignments while confining the window of considered resources in a single 1-minute interval. If a task cannot be fully assigned, it may be reconsidered in the next 1-minute interval unless one or more of its entailed subtasks are dropped due to deadline expiry. Consequently, as time progresses, it becomes increasingly difficult for RMNT to successfully assign full tasks. This results in few tasks being completed in the early 1-minute intervals, while later intervals see almost no assignments due to excessive reneging or a lack of sufficient workers. Due to this lossy nature, the average subtask response delay for RMNT is lower than that of all other schemes.

Figure 4d depicts the percentage of reneging against varying numbers of subtasks. Recall that the SDoA metric accounts for subtasks that are not considered for assignment during the 5-minute period, while reneging represents subtasks dropped due to their deadlines expiring before sufficient resources are available for their execution. P-MWSC-TDD and P-MWSC-BUD maintain a zero reneging percentage regardless of the number of subtasks, thanks to their proactive strategies that create subtask-worker allocation decisions based on the availability of forthcoming workers. RMNT, RMRD and P-SMNW show an upward trend of the reneging percentage as the number of subtasks increases. Compared to P-MWSC-TDD and P-MWSC-BUD, RMRD shows a rise in reneging to 21% when the number of subtasks is 150. On the other hand, RMNT exhibits a substantially higher reneging percentage, nearly doubling from 35% to 73% at 50 and 150 subtasks, respectively. These results align with the TSR, average response delay and SDoA trends discussed earlier. The high reneging percentages severely impact performance in RMNT, as tasks with any reneged subtasks become ineligible for assignment in any 1-minute interval regardless of resource availability. On the other hand, P-SMNW exhibits a maximum reneging percentage of 55% due to disregard of worker availability. While P-SMNW has a lower reneging percentage than RMNT, its dropped subtasks are scattered across all tasks, resulting in a much lower TSR.

3- Impact of Varying Number of Tasks.

In this experiment, the number of subtasks is set to 150 and the number of workers to 50, while the number of tasks (i.e., DT service requests) varies from 4 to 20. The total of 150 subtasks are randomly distributed across the tasks. Figure 5a depicts the TSR against increasing number of tasks. As can be seen in the figure, the TSR rises with the increasing number of tasks across all schemes. This can be attributed to the split of 150 subtasks among more tasks, inevitably resulting in smaller tasks that are more likely to find sufficient resources for execution. Consistent with the previous experiment, the P-MWSC-TDD scheme provides the upper bound on the TSR. P-MWSC-TDD achieves a peak TSR improvement of 43% and 48% compared to RMNT and RMRD, respectively. P-MWSC-BUD achieves a peak improvement of 38% and 43% compared to RMNT and RMRD, respectively. Due to its heuristic approach that approximates the solution, P-MWSC-BUD shows an average performance gap of 5% compared to the P-MWSC-TDD solution. RMRD performs poorly, particularly when the number of tasks is small. This is attributed to RMRD's

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3524868

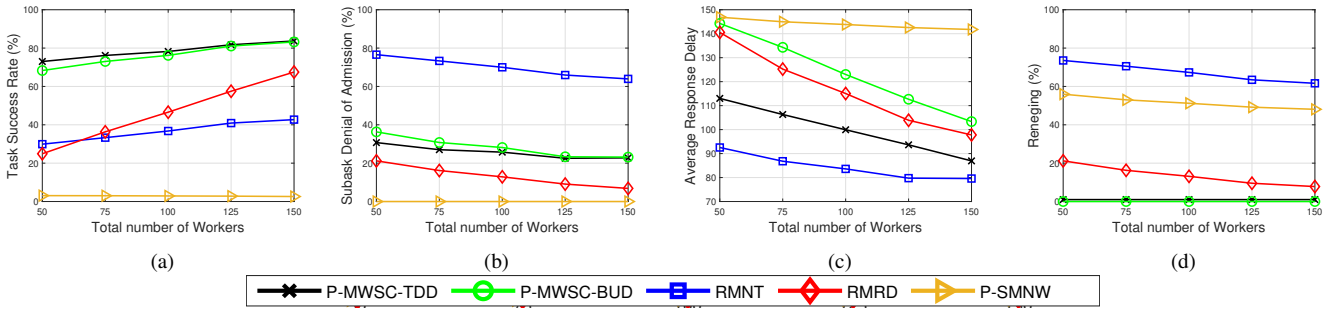

Fig. 6: Performance results of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD, and P-SMNW over varying numbers of workers.

disregard for full task execution requirements, as discussed earlier. As the number of tasks increases, RMRD gradually improves its TSR, achieving a maximum of only 25% at 20 tasks. RMNT continues to struggle in matching the performance of its proactive counterparts, with TSRs ranging from 17% to 30% for 4 to 20 tasks, respectively. Lastly, P-SMNW exhibits extremely low TSR, due to its disregard of dynamic worker availability.

Figure 5b shows the SDoA over varying numbers of tasks. The results align with those shown in the previous experiment. Both P-MWSC-TDD and P-MWSC-BUD show a notable decline in the SDoA. P-MWSC-TDD and P-MWSC-BUD reduce the SDoA by up to 47% and 41%, respectively, compared to RMNT. P-MWSC-BUD has an average performance gap of 6% compared to P-MWSC-TDD due to its heuristic approach. Note that this gap does not shrink by varying the number of tasks, because P-MWSC-BUD makes subtask-worker allocation decisions in the first stage regardless of the size or number of tasks present. In contrast, the SDoA of RMNT is slightly affected by varying the number of tasks. As explained earlier, RMNT only admits subtasks that contribute to complete task execution within a limited 1-minute interval, due by its myopic knowledge of resource availability. These negative effects persist even as tasks become less demanding, resulting in a mere 11% improvement in SDoA. P-SMNW and RMRD maintain the lowest and second lowest SDoA among all schemes, respectively, due to their assignment strategy.

Figure 5c shows the average response delay over varying numbers of tasks. The results align with those presented in the previous experiment. As the number of tasks grows, P-MWSC-TDD shows an upward trend in the average response delay, with a reduction of up to 33% compared to RMRD. P-MWSC-BUD, RMRD and P-SMNW show little change with varying number of tasks, because they make subtask-worker allocation decisions without regard to the number of tasks or their sizes. RMNT is expected to benefit from shrinking task sizes, as it enhances its ability to meet the task requirements in 1-minute intervals. However, even when the 150 subtasks are distributed among 20 tasks, RMNT fails to approach the performance of the proactive schemes, as shown in Figures 4a and 4b. Finally, Figure 5d shows the reneging percentage over varying numbers of tasks. As expected, P-MWSC-TDD and P-MWSC-BUD consistently maintain a zero reneging percentage. The results for RMNT, RMRD and P-SMNW show slight change when varying the number of tasks. For RMNT,this is because smaller tasks have a higher probability of being fully assigned. For RMRD and P-SMNW, the subtask-worker allocation decisions have a better chance of producing full task assignments when tasks are smaller.

4- Impact of Varying Number of Workers.

In this experiment, the number of subtasks is set to 150 and the number of tasks is set to 20, while the number of workers varies from 50 to 150. Figure 6a depicts the TSR of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD and P-SMNW over varying numbers of workers. It can be observed that P-MWSC-TDD and P-MWSC-BUD show an improvement in the TSR as the number of workers rises. Intuitively, increasing the number of workers enhances the quality of subtask-worker allocation opportunities, which in turn boosts the number of fully satisfied service requesters. Compared to RMNT, RMRD, and P-SMNW, P-MWSC-TDD achieves notable TSR improvements of 44%, 48%, and 70% respectively. P-MWSC-BUD performs similarly to P-MWSC-TDD, with a maximum gap of 6% when there are 50 workers. This gap closes as the number of workers increases because the first stage of P-MWSC-BUD has fewer subtask rejections, which reduces pruning in the second stage. P-MWSC-BUD improves the TSR by up to 38%, 43% and 65% compared to RMNT and RMRD, respectively. RMNT exhibits a slightly sharper upward pattern, increasing from 30% to 42% as the number of workers grows from 50 to 150, respectively. This indicates that even with a high number of workers, RMNT cannot match the performance of proactive schemes due to the reasons previously discussed. RMRD benefits the most from an increased number of workers, showing a steep rise in TSR from 24% to 67.5% when the number of workers is 50 and 150 workers, respectively. However, even with abundant resources, there remains a 15.5% gap in the TSR compared to the proactive schemes. Lastly, P-SMNW significantly underperforms in terms of the achieved TSR. Note that due to its objective of minimizing the total number of recruited workers, P-SMNW does not achieve performance gains as the number of workers increases.

This article has been accepted for publication in IEEE Internet of Things Journal. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/JIOT.2024.3524868

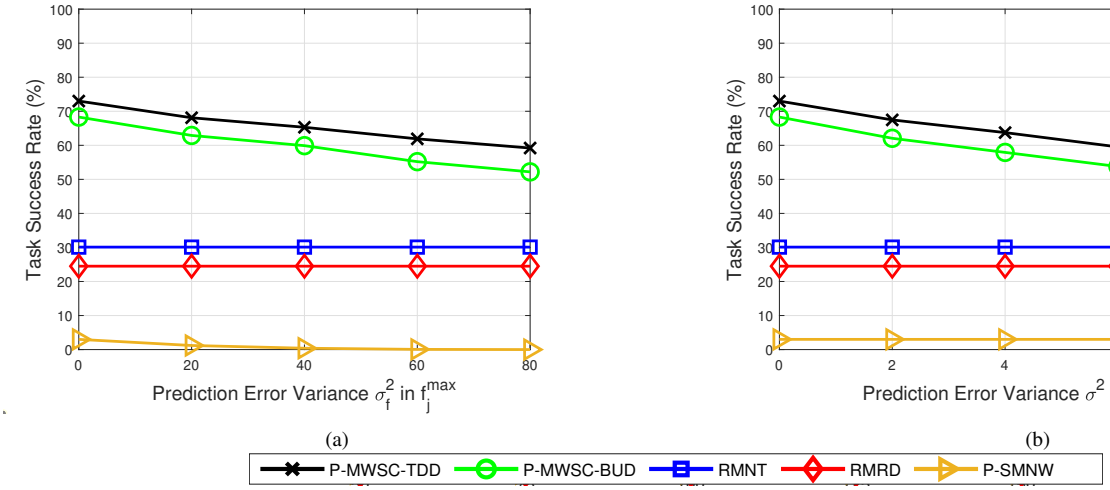

Fig. 7: Performance results of P-MWSC-TDD, P-MWSC-BUD, RMNT, RMRD, and P-SMNW over varying prediction error variance in f_j^{max} and α_j, respectively.

Figure 6b depicts the SDoA over varying numbers of workers. P-MWSC-TDD, P-MWSC-BUD, RMNT and RMRD exhibit a downward trend in SDoA as the number of workers increases, due to the improvement of subtask-worker allocation opportunities. Consistent with the previous experiments, P-SMNW admits all tasks, while RMRD exhibits the smallest SDoA values due to its task-blind admission policy. In contrast, RMNT shows the highest SDoA among all schemes for the same reasons explained above. In particular, RMNT's SDoA improves only by 13% when the number of workers increases from 50 to 150, highlighting the limitations of its reactive approach. Compared to RMNT, P-MWSC-TDD and P-MWSC-BUD reduce the SDoA by up to 41% and 34%, respectively. Lastly, the performance gap between P-MWSC-BUD and P-MWSC-TDD is most notable when the number of workers is 50, but quickly diminishes to zero as the number of workers increases.

Figure 6c illustrates the average response delay over varying numbers of workers. All schemes demonstrate a decline in the average response delay as the number of workers rises due to the greater resource availability, which reduces waiting times. Due to its excessively low TSR, RMNT maintains the lowest values of average response delay, for the reasons discussed earlier. P-MWSC-TDD provides the lower bound on average response delay. In contrast, P-SMNW experiences the highest response delays, resulting from its policy of minimizing recruited workers without accounting for their dynamic availability, which extends waiting times. On the other hand, P-MWSC-BUD exhibits high average response delay, because it prioritizes maximizing the number of assigned subtasks in the first stage, which can lead to longer waiting delays. This is the most notable shortcoming of P-MWSC-BUD compared to P-MWSC-TDD. Although P-MWSC-BUD approaches the TSR values of P-MWSC-TDD, it does so at the cost of higher average response delays. RMRD performs similarly to P-MWSC-BUD, showing slightly lower average response delay values due to its prioritization of minimal delay. We note that P-MWSC-TDD, P-MWSC-BUD and RMRD approach convergence in response delays as the number of workers grows significantly. However, P-MWSC-BUD and RMRD underperform in the TSR and SDoA, as discussed earlier, confirming the superiority of our P-MWSC-TDD scheme.

Lastly, Figure 6d shows the reneging percentage over varying numbers of workers. Due to their proactive strategy and consideration of dynamic worker availability, both P-MWSC-TDD and P-MWSC-BUD consistently maintain a zero reneging percentage. In contrast, RMRD shows a decline in reneging as the number of workers increases, reaching almost 8% at 150 workers, thanks to improved subtask-worker allocation opportunities. However, as revealed in Figure 6a, the reneged subtasks contribute to an overall 15.5% TSR gap compared to P-MWSC-TDD, reflecting the limitations of RMRD even when a large number of workers is available. P-SMNW shows a gradual decrease in the percentage of reneged subtasks as the number of workers increases. As mentioned earlier, the reneged subtasks are dispersed over all available tasks, leading to an extremely low TSR, regardless of resource availability. RMNT displays a significantly higher reneging percentage, which decreases as the number of workers rises. Even with 150 workers, RMNT achieves a mere 12% improvement in reneging percentages. This clearly demonstrates that even with ample resources, the reactive strategy severely undermines performance gains.

5- Impact of Prediction Errors.

Figure 7a depicts the TSR over varying prediction error variance in f_j^{max}. To evaluate the impact of prediction errors, we introduce a Gaussian random variable with a mean of zero and a variance σ_f^2 to the predicted worker CPU availability

and denote the resulting value as $\tilde{f}_j^{max}$. We conduct the experiment with the number of subtasks, workers, and tasks set to 150, 50, and 20, respectively. P-MWSC-TDD, P-MWSC-BUD and P-SMNW use $\tilde{f}_j^{max}$ to proactively make subtask-worker allocation decisions, while the actual CPU availability f_j^{max} is determined during execution. If worker's w_j predicted availability is overestimated, we iteratively remove the least profitable subtasks until worker w_j is no longer overloaded. Note that the yielded TSR of RMNT and RMRD remains the same, since they are both reactive schemes that do not use any predictions. In contrast, the TSR of both P-MWSC-TDD and P-MWSC-BUD decreases as the prediction error variance σ^2 increases. It can be observed that when $\sigma^2 = 0$, P-MWSC-TDD and P-MWSC-BUD have perfect foresight about worker's availability, providing the upper bound on the achievable TSR. As σ^2 becomes larger, we notice a decline in the TSR due to subtasks being dropped from overloaded workers. This is because a rise in σ^2 causes $\tilde{f}_j^{max}$ to deviate more significantly from the actual CPU availability f_j^{max}, making the predictions less accurate, which leads to higher subtask dropping. However, it can be observed that the decline is small, where P-MWSC-TDD and P-MWSC-BUD experience up to a 13% and 16% drop in TSR, respectively. This is because both schemes distribute the workload among the workers, which drastically reduces the probability of overloading and excessive subtask dropping. The TSR of P-SMNW is further impacted by prediction errors. This is due to the fact that P-SMNW minimizes the number of recruited workers, leading to increased subtask droppings and a significantly lower TSR.

Figure 7b depicts the TSR over varying prediction error variance in α_j. We conduct the experiment with the number of subtasks, workers, and tasks set to 150, 50, and 20, respectively. A Gaussian random variable with a mean of zero and a variance σ_α^2 is introduced to the predicted worker arrival time α_j, and the resulting value is denoted by $\tilde{\alpha}_j$. P-MWSC-TDD and P-MWSC-BUD employ the time arrival predictions $\tilde{\alpha}_j$ to make subtask-worker allocation decisions. Note that P-SMNW is unaware of dynamic worker availability, therefore, it does not take $\tilde{\alpha}_j$ into account. RMNT and RMRD are reactive schemes that decide subtask-worker allocations after a worker has arrived. Consequently, the achieved TSR of RMNT, RMRD and P-SMNW remains unchanged as σ_α^2 varies. When $\sigma_\alpha^2 = 0$, P-MWSC-TDD and P-MWSC-BUD possess perfect foresight regarding worker availability, establishing the upper bound on the achievable TSR. The presence of prediction errors may lead to reneged subtasks due to deadline expiry before a worker commences the subtask execution, which lowers the achieved TSR. Hence, an increase in σ_α^2 causes a decline in the TSR, as observed in the figure. P-MWSC-TDD and P-MWSC-BUD experience up to a 17% and 18% drop in TSR, respectively. Therefore, it can be concluded from the results in Figure 7 that despite the presence of prediction errors, P-MWSC-TDD and P-MWSC-BUD outperform both the reactive and proactive baselines, leading to higher satisfaction among DT service requesters.

VII. Practical Considerations.

In the context of EEC being a volunteer computing paradigm, a persistent challenge is the risk of insufficient participating workers, which jeopardizes the capability to meet the computational demands of service requesters [36]. Addressing worker participation requires the implementation of robust cooperation incentives, where task and resource allocation are optimized according to personalized incentive design, to ensure that workers receive satisfactory compensation, encouraging sustained participation in the system [28]. On the other hand, security and privacy concerns pose as significant hindrances to worker involvement. This warrants the integration of reputation trackers into the task allocation process, which allows up-to-date, continuous and reliable assessments of worker trustworthiness and identification malicious behavior [37]. On the other hand, socially-based cooperative strategies can enable the formation of communities of trusted requesters and/or workers, which aids in ensuring the privacy and security of all parties involved [38]. Finally, the blockchain technology can be incorporated to ensure user privacy and data security [39].

From the scheduler's perspective, the effectiveness of proactive task allocation can be significantly challenged by random and unpredictable user behavior that does not follow recurrent patterns. Users exhibiting high variability can lead to low prediction accuracy, undermining the performance gains of proactive schemes, such as P-MWSC. A possible solution is to use sophisticated prediction techniques, such as the Hierarchical Dirichlet Process-Hidden Semi-Markov Model (HDP-HSMM) [19]. This allows to produce high accuracy predictions even under high levels of randomness and abrupt changes. In addition, incorporating hybrid proactive-reactive strategies can preserve the performance gains of predictive proactive task allocation schemes. Specifically, employing adaptive recovery mechanisms that respond to sudden changes in worker availability at runtime can boost the robustness of proactive scheduling in dynamic environments. Such strategies collectively aim to overcome the practical limitations of worker engagement and random user behavior in EEC environments.

VIII. Conclusions

In this paper, we proposed a novel proactive task allocation scheme to meet the sophisticated requirements of DT services due to their reliance on multi-modal data from distributed data producers. P-MWSC makes proactive task allocation decisions that ensure full task completion by incorporating predictions of future worker availability into the decision-making process, thereby mitigating the effects of worker dynamicity and intermittent availability. To address the high computational complexity of the problem, we proposed an iterative top-down decomposition approach exploiting the bi-level hierarchical structure of P-MWSC to solve the problem in a time-efficient manner. We also proposed a heuristic solution based on a bottom-up decomposition approach to provide near-optimal solutions with increased time-efficiency. We evaluated the performance of the proposed solutions against the extensive BILP solution

to validate the effectiveness of our decomposition approach, demonstrating significant runtime reduction. We compared the performance of P-MWSC to two prominent reactive task allocation approaches and a prominent proactive task allocation approach, using publicly available data from a realistic testbed, which represents heterogeneous workers in various dynamic resource usage scenarios. Simulation results showed that P-MWSC significantly outperformed baselines in terms of task success rate, subtask denial of admission rate, and average response delay. P-MWSC also demonstrated its scalability as it maintains significant superiority with varying numbers of tasks, subtasks, and workers. Moreover, it maintained performance even under erroneous predictions, highlighting that our proposed approaches are robust to errors that can happen in realistic systems.

In the future, we plan to enhance the robustness of proactive task allocation by using stochastic modeling techniques to quantify prediction uncertainties arising from the spatiotemporal variation in requesters' demands and worker availability, incorporating these uncertainties into the task allocation decision. This approach will enable the development of a more resilient task allocation framework that preserves predictive performance gains while mitigating the adverse effects of uncertainty.

ACKNOWLEDGEMENT

This research is supported by a grant from the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant number: ALLRP 549919-20, and a grant from Distributive, Ltd. This work was also supported in part by Qatar University under Grant IRCC-2024-494.

REFERENCES

[1] Y. Wang, Z. Su, S. Guo, M. Dai, T. H. Luan, and Y. Liu, "A survey on digital twins: Architecture, enabling technologies, security and privacy, and future prospects," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 14 965–14 987, 2023.

[2] L. F. Rivera, M. Jimenez, N. M. Villegas, G. Tamura, and H. A. Muller, "The forging of autonomic and cooperating digital twins," *IEEE Internet Computing*, vol. 26, no. 5, p. 41–49, Sep. 2022.

[3] Q. Guo, F. Tang, T. K. Rodrigues, and N. Kato, "Five disruptive technologies in 6g to support digital twin networks," *IEEE Wireless Communications*, vol. 31, no. 1, p. 149–155, Feb. 2024.

[4] B. Mao, X. Zhou, J. Liu, and N. Kato, "Digital twin satellite networks toward 6g: Motivations, challenges, and future perspectives," *IEEE Network*, vol. 38, no. 1, pp. 54–60, 2024.

[5] H. Guo, X. Zhou, J. Wang, J. Liu, and A. Benslimane, "Intelligent task offloading and resource allocation in digital twin based aerial computing networks," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3095–3110, 2023.

[6] F. Tang, X. Chen, T. K. Rodrigues, M. Zhao, and N. Kato, "Survey on digital twin edge networks (diten) toward 6g," *IEEE Open Journal of the Communications Society*, vol. 3, pp. 1360–1381, 2022.

[7] A. Alkhateeb, S. Jiang, and G. Charan, "Real-time digital twins: Vision and research directions for 6g and beyond," *IEEE Communications Magazine*, vol. 61, no. 11, p. 128–134, Nov. 2023.

[8] B. Clerckx, Y. Mao, Z. Yang, M. Chen, A. Alkhateeb, L. Liu, M. Qiu, J. Yuan, V. W. S. Wong, and J. Montojo, "Multiple access techniques for intelligent and multi-functional 6g: Tutorial, survey, and outlook," 2024.

[9] D. He, K. Guan, D. Yan, H. Yi, Z. Zhang, X. Wang, Z. Zhong, and N. Zorba, "Physics and ai-based digital twin of multi-spectrum propagation characteristics for communication and sensing in 6g and beyond," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 11, pp. 3461–3473, 2023.

[10] R. F. El Khatib, S. A. Elsayed, N. Zorba, and H. S. Hassanein, "Optimal proactive resource allocation at the extreme edge," in *ICC 2022 - IEEE International Conference on Communications*. IEEE, May 2022.

[11] D. Chatzopoulos, C. Bermejo, E. u. Haq, Y. Li, and P. Hui, "D2d task offloading: A dataset-based q&a," *IEEE Communications Magazine*, vol. 57, no. 2, p. 102–107, Feb. 2019.

[12] R. Tourani, S. Srikanteswara, S. Misra, R. Chow, L.-L. Yang, X. Liu, and Y. Zhang, "Democratizing the edge: A pervasive edge computing framework," *ArXiv*, vol. abs/2007.00641, 2020.

[13] J. Liang, B. Ma, Z. Feng, and J. Huang, "Reliability-aware task processing and offloading for data-intensive applications in edge computing," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4668–4680, 2023.

[14] A. Khazali, A. Bozorgchenani, D. Tarchi, M. G. Shayesteh, and H. Kalbkhani, "Joint task assignment, power allocation and node grouping for cooperative computing in noma-mmwave mobile edge computing," *IEEE Access*, vol. 11, p. 93664–93678, 2023.

[15] T. Fang, F. Yuan, L. Ao, and J. Chen, "Joint task offloading, d2d pairing, and resource allocation in device-enhanced mec: A potential game approach," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3226–3237, 2022.

[16] H. Yuan and M. Zhou, "Profit-maximized collaborative computation offloading and resource allocation in distributed cloud and edge computing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1277–1287, 2021.

[17] J. Xie, Y. Jia, W. Wen, Z. Chen, and L. Liang, "Dynamic d2d multi-hop offloading in multi-access edge computing from the perspective of learning theory in games," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 305–318, 2023.

[18] I. M. Amer, S. M. Oteafy, S. A. Elsayed, and H. S. Hassanein, "Task provisioning in unreliable edge networks: Inferring utility," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*, 2023, pp. 3015–3020.

[19] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein, "Rump: Resource usage multi-step prediction in extreme edge computing," *Computer Communications*, vol. 210, p. 45–57, Oct. 2023.

[20] M. Kantardjian, S. A. Elsayed, and H. S. Hassanein, "Dynamic worker availability prediction at the extreme edge," in *GLOBECOM 2023 - 2023 IEEE Global Communications Conference*. IEEE, Dec. 2023.

[21] J. Hooker and G. Ottosson, "Logic-based benders decomposition," *Mathematical Programming*, vol. 96, no. 1, p. 33–60, Apr. 2003.

[22] J. N. Hooker, *Logic-Based Benders Decomposition for Large-Scale Optimization*. Springer International Publishing, 2019, p. 1–26.

[23] R. Kain, S. A. Elsayed, Y. Chen, and H. S. Hassanein. Resource usage of applications running on raspberry pi devices. [Online]. Available: http://dx.doi.org/10.5683/SP3/GOZAJE

[24] M. Mehrabi, D. You, V. Latzko, H. Salah, M. Reisslein, and F. H. P. Fitzek, "Device-enhanced mec: Multi-access edge computing (mec) aided by end device computation and caching: A survey," *IEEE Access*, vol. 7, p. 166079–166108, 2019.

[25] H. Guo and J. Liu, "Uav-enhanced intelligent offloading for internet of things at the edge," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2737–2746, 2020.

[26] J. Liu, N. Liu, L. Liu, S. Li, H. Zhu, and P. Zhang, "A proactive stable scheme for vehicular collaborative edge computing," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10 724–10 736, 2023.

[27] H. Flores, A. Zuniga, S. Tarkoma, L. Tonetto, T. Braud, P. Hui, Y. Li, M. Ammar, and P. Nurmi, "Collaboration stability: Quantifying the success and failure of opportunistic collaboration," *Computer*, vol. 55, no. 8, p. 70–81, Aug. 2022.

[28] M. De'bas, S. A. Elsayed, and H. S. Hassanein, "Multitiered worker-oriented resource allocation: Mitigating worker attrition at the extreme edge," *IEEE Internet of Things Journal*, 2023.

[29] Schrijver, Alexander. Combinatorial optimization: Polyhedra and efficiency. [Online]. Available: https://link.springer.com/book/9783540443896

[30] M. Dawande, J. Kalagnanam, P. Keskinocak, F. S. Salman, and R. Ravi, "Approximation algorithms for the multiple knapsack problem with assignment restrictions." *Journal of Combinatorial Optimization*, vol. 4, p. 171–186, Jun. 2000.

[31] A. Frangioni, "About lagrangian methods in integer optimization," *Annals of Operations Research*, vol. 139, p. 163–193, 2005.

[32] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency iot services in multi-access edge computing," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 3, pp. 668–682, 2019.

[33] T. T. Vu, D. N. Nguyen, D. T. Hoang, and E. Dutkiewicz, "Qos-aware fog computing resource allocation using feasibility-finding benders decomposition," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[34] J.-F. Côté, M. Dell'Amico, and M. Iori, "Combinatorial benders' cuts for the strip packing problem," *Operations Research*, vol. 62, no. 3, p. 643–661, 2014.

[35] Y. Sun and X. Zhang, "A2c learning for tasks segmentation with cooperative computing in edge computing networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*. IEEE, Dec. 2022.

[36] S. B. Azmy, R. F. El-Khatib, N. Zorba, and H. S. Hassanein, "Extreme edge computing challenges on the edge-cloud continuum," in *2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2024, pp. 99–100.

[37] M. Bradbury, A. Jhumka, and T. Watson, "Trust trackers for computation offloading in edge-based iot networks," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications*, 2021, pp. 1–10.

[38] A. A. Moustafa, S. A. Elsayed, and H. S. Hassanein, "Community-oriented resource allocation at the extreme edge," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 5583–5588.

[39] P. Lang, D. Tian, X. Duan, J. Zhou, Z. Sheng, and V. C. M. Leung, "Blockchain-based cooperative computation offloading and secure handover in vehicular edge computing networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 7, pp. 3839–3853, 2023.

APPENDIX

Let $\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5$ denote the Lagrangian multipliers of the workload, deadline, service capacity and relaxed integrality constraints in **P1**. Then, the Lagrangian function associated with the relaxed program of **P1** is given by:

$$\begin{aligned}
&L(\mathbf{x}, \mathbf{y}, \lambda_1, \lambda_2, \lambda_3^{(1)}, \lambda_3^{(2)}, \lambda_4^{(1)}, \lambda_4^{(2)}, \lambda_5^{(1)}, \lambda_5^{(2)}) = \\
&\sum_{\forall k} g_k y_k + \sum_j \lambda_{1,j}\Big(\sum_i c_i x_{i,j} - f_j^{max}\Big) + \\
&\sum_i \lambda_{2,i}\Big(\sum_j D_{i,j} x_{i,j} - \delta_i^{max}\Big) + \\
&\sum_k \sum_{i \in \mathcal{B}_k} \lambda_{3,k,i}^{(1)}\Big(\sum_j x_{i,j} - y_k\Big) + \\
&\sum_k \sum_{i \in \mathcal{B}_k} \lambda_{3,k,i}^{(2)}\Big(y_k - \sum_j x_{i,j}\Big) + \\
&\sum_i \sum_j \Big(\lambda_{4,i,j}^{(1)}(x_{i,j} - 1) - \lambda_{4,i,j}^{(2)} x_{i,j}\Big) + \\
&\sum_k \Big(\lambda_{5.k}^{(1)}(y_k - 1) - \lambda_{5,k}^{(2)} y_k\Big)
\end{aligned} \tag{12}$$

Since the objective function in **P1** is linear, the KKT conditions are necessary and sufficient for optimality. Applying the KKT conditions to the constraints in **P1** for complementary slackness and solving each one for $x_{i,j}^*$ and y_k^* yields the following set of equations.

$$\lambda_{1,j}^*\Big(\sum_i c_i x_{i,j}^* - f_j^{max}\Big) = 0, \quad \forall j \tag{13a}$$

$$x_{i,j}^* = \frac{f_j^{max} - \sum_{\hat{i} \neq i} c_{\hat{i}} x_{\hat{i},j}^*}{c_i}, \quad \lambda_{1,j}^* > 0 \tag{13b}$$

$$\lambda_{2,i}^*\Big(\sum_j D_{i,j} x_{i,j}^* - \delta_i^{max}\Big) = 0, \quad \forall i \tag{13c}$$

$$x_{i,j}^* = \frac{\delta_i^{max} - \sum_{\hat{j} \neq j} D_{i,\hat{j}} x_{i,\hat{j}}^*}{D_{i,j}}, \quad \lambda_{2,i}^* > 0 \tag{13d}$$

$$\lambda_{3,k,i}^{*(1)}\Big(\sum_j x_{i,j}^* - y_k^*\Big) = 0, \quad \forall i \in \mathcal{B}_k, \forall k \tag{13e}$$

$$x_{i,j}^* = y_k^* - \sum_{\hat{j} \neq j} x_{i,\hat{j}}^*, \quad \lambda_{3,k,i}^{*(1)} > 0 \tag{13f}$$

$$y_k^* = \sum_j x_{i,j}^*, \quad \lambda_{3,k,i}^{*(1)} > 0 \tag{13g}$$

$$\lambda_{3,k,i}^{*(2)}\Big(y_k^* - \sum_j x_{i,j}^*\Big) = 0, \quad \forall i \in \mathcal{B}_k, \forall k \tag{13h}$$

$$x_{i,j}^* = y_k^* - \sum_{\hat{j} \neq j} x_{i,\hat{j}}^*, \quad \lambda_{3,k,i}^{*(2)} > 0 \tag{13i}$$

$$y_k^* = \sum_j x_{i,j}^*, \quad \lambda_{3,k,i}^{*(2)} > 0 \tag{13j}$$

$$\lambda_{4,i,j}^{*(1)}(x_{i,j}^* - 1) = 0, \quad \forall i, \forall j \tag{13k}$$

$$x_{i,j}^* = 1, \quad \lambda_{4,i,j}^{*(1)} > 0 \tag{13l}$$

$$\lambda_{4,i,j}^{*(2)} x_{i,j}^* = 0, \quad \forall i, \forall j \tag{13m}$$

$$x_{i,j}^* = 0, \quad \lambda_{4,i,j}^{*(2)} > 0 \tag{13n}$$

$$\lambda_{5.k}^{*(1)}(y_k^* - 1) = 0, \quad \forall k \tag{13o}$$

$$y_k^* = 1, \quad \lambda_{5.k}^{*(1)} > 0 \tag{13p}$$

$$\lambda_{5,k}^{*(2)} y_k^* = 0, \quad \forall k \tag{13q}$$

$$y_k^* = 0, \quad \lambda_{5.k}^{*(2)} > 0 \tag{13r}$$

The gradient of the Lagrangian w.r.t. the decision variables yields the following two equations.

$$\begin{aligned}
\frac{\partial L}{\partial x_{i,j}} &= \lambda_{1,j}^* c_i + \lambda_{2,i}^* D_{i,j} + \lambda_{3,k,i}^{*(1)} \\
&- \lambda_{3,k,i}^{*(2)} + \lambda_{4,i,j}^{*(1)} - \lambda_{4,i,j}^{*(2)} = 0, \forall i, \forall j
\end{aligned} \tag{14a}$$

$$\frac{\partial L}{\partial y_k} = g_k - \lambda_{3,k,i}^{*(1)} + \lambda_{3,k,i}^{*(2)} + \lambda_{5,k}^{*(1)} - \lambda_{5,k}^{*(2)} = 0, \quad \forall k \tag{14b}$$

By multiplying 14a with $x_{i,j}^*$ and using Eq. (13k) and (13m), we get:

$$x_{i,j}^*\Big(\lambda_{1,j}^* c_i + \lambda_{2,i}^* D_{i,j} + \lambda_{3,k,i}^{*(1)} - \lambda_{3,k,i}^{*(2)}\Big) + \lambda_{4,i,j}^{*(1)} = 0 \tag{15}$$

By substituting Eq. (13a), (13c), (13e), and (13h) in Eq. (15), the upper bound on $x_{i,j}^*$ is derived in Eq. (16).

$$x_{i,j}^* = \frac{-u}{\lambda_{3,k,i}^{*(1)} - \lambda_{3,k,i}^{*(2)}} \tag{16}$$

where $u = \lambda_{4,i,j}^{*(1)} + \lambda_{1,j}^* f_j^{max} + \lambda_{2,i}^* \delta_i^{max}$. Similarly, multiplying Eq. (14b) with y_k^* and using 13o and 13q yields the following:

$$y_k^*\Big(g_k - \lambda_{3,k,i}^{*(1)} + \lambda_{3,k,i}^{*(2)}\Big) = -\lambda_{5,k}^{*(1)} \tag{17}$$

Using equations 13e13h, 13o, and 13q the upper bound on y_k^* is derived in Eq. (6.

$$y_k^* = \frac{-\lambda_{5,k}^{*(1)}}{g_k - \lambda_{3,k,i}^{*(1)} + \lambda_{3,k,i}^{*(2)}} \tag{18}$$

The upper bounds on $x_{i,j}^*$ and y_k^* shown in equations 5 and 6 are obtained from the analysis of the above set of equations.