Task Provisioning in Unreliable Edge Networks: Inferring Utility

Ibrahim M. Amer*, Sharief M. A. Oteafy*†, Sara A. Elsayed*, and Hossam S. Hassanein*
*School of Computing, Queen's University, Kingston, ON, Canada
†School of Computing, DePaul University, Chicago, Illinois, USA
ibrahim.amer@queensu.ca, soteafy@depaul.edu, {selsayed, hossam}@cs.queensu.ca

GLOBECOM 2023 - 2023 IEEE Global Communications Conference | 979-8-3503-1090-0/23/$31.00 ©2023 IEEE | DOI: 10.1109/GLOBECOM54140.2023.10437918

***Abstract*—Edge computing can satisfy the requirements of latency-critical and data-intensive applications by exploiting computational resources of end devices. However, such devices inherently suffer from dynamic user behavior, cyclic task-switching, varying link qualities which often impact their reliability. In addition, in incentivized systems, they may often over-estimate their advertised capabilities and consequently fail on delivering. In this paper, we propose the Reputation-based Task Assignment and Replication (RTAR) scheme. RTAR is the first scheme that uses a black box approach to perform cost-efficient task replication that accounts for workers' reliability and preserves workers' privacy by not requiring or soliciting any information about their devices. RTAR incorporates a reputation model using beta distribution to estimate the worker's reputation based on past performance. We formulate the problem as an Integer Linear Program (ILP) that strives to maximize the overall reputation of recruited workers, while abiding by a certain budget limit for each task. We also propose the RTAR-Heuristic (RTAR-H) scheme. RTAR-H uses matching theory to solve the optimization problem in a time-efficient manner. Extensive evaluations show that RTAR yields 63% and 68% reduction in recruitment cost and number of replicas, respectively, compared to a baseline scheme that blindly maximizes the number of replicas. Moreover, RTAR-H closely approaches the optimal solution, rendering a small gap of up to 1% and 1.2% in terms of task drop rate and recruitment cost, respectively.**

***Index Terms*—Edge Computing, Resources Provisioning, Task Replication, Matching Theory, Reliability**

I. Introduction

With the projected adoption of the Internet of Things (IoT), it is estimated that there will be 41.6 billion IoT devices connected to the Internet by 2025 [1]. Moreover, the global Artificial Intelligence (AI) market is expected to reach $1,811.75 billion by 2030 [2], while the natural language processing market is forecasted to grow to $49.4 billion by 2027 [3]. This massive proliferation is expected to trigger significant demands on computing resources to meet the strict Quality of Service (QoS) requirements of latency-critical and data-intensive applications, such as ChatGPT, Tactile Internet, metaverse, smart cities, and autonomous vehicles [4]–[7]. Cloud computing fails to satisfy such requirements, due to the transmission of large amounts of data to distant data centers, which can increase latency and create massive traffic influx at backhaul links [8].

Edge Computing (EC) has emerged as a propitious computing paradigm that can alleviate the aforementioned issues by harnessing the copious yet underutilized computational resources of end devices, such as smartphones, laptops, and connected vehicles. EC facilitates data processing much closer to end users, which can drastically diminish latency [9]. Moreover, in contrast to infrastructure-based edge computing paradigms that are solely governed by cloud service providers and/or network operators, EC paradigms that leverage the computational resources of end devices avoid such a monopoly and open a new tech market that permits more players to create and administer their own edge cloud [10].

Despite the promising prospects of device-based EC paradigms, their full potential can be impeded by the fact that end devices are user-owned, thus exposing them to a dynamic user access behavior, which can profoundly impact their reliability. For instance, at any given time, users may use their devices to stream a video or run any computationally intensive application. This can create a deficiency in their available computation and communication resources or cause the users to drop the execution of offloaded tasks in order to preserve their own convenience. Such factors can trigger significant fluctuations in the availability of end devices, which can lead to increased task drop rate and execution time. Thus, it is imperative for task allocation schemes to account for the intermittent availability and unreliability of workers.

Most existing resource allocation schemes fail to account for the reliability of workers. Some schemes resort to task migration to deal with service interruption after it occurs [11]. However, such schemes can trigger an additional delay, due to the need to transmit the service from one worker to another. In addition, poor network connectivity between unreliable workers can lead to increased task failure.

Recently, task replication has been explored to deal with service interruptions in device-based EC paradigms [12], [13]. By enabling multiple workers to execute the same task, the system can tolerate failures or unexpected events associated with individual workers. If one worker fails or drops out, redundant workers can continue the task without disruption, reducing the chances of task failure and minimizing the impact on task execution. However, existing task replication schemes tend to either blindly maximize the number of replicas [12], or use a random threshold to specify its upper bound [13]. Thus, such schemes fail to adequately control the number of

replicas, which can lead to excessive recruitment costs and wasted resources. Furthermore, existing schemes assume that the decision maker has information about the workers, such as their computational resources, location, network connectivity, data rates, and available power [12]–[14]. However, possessing such information can raise privacy concerns that make users reluctant to endow their computational resources to the service, since it can be used to monitor user behavior or even identify devices.

In this paper, we propose the Reputation-based Task Assignment and Replication (RTAR) scheme. RTAR uses a black box approach that fosters cost-efficient task replication without requiring or relying on any prior information about the workers. Instead, RTAR leverages the reputation of workers based on their performance to make informed decisions and account for their reliability. RTAR relies on the fact that poor or inconsistent performance of a worker is bound to lead to reduced trust and credibility in the worker, affecting their future task assignments and potential earnings, as well as task replication decisions. The higher the reputation of a worker, the lower the number of replicas needed for the task assigned to it. We formulate the task replication problem as an Integer Linear Program (ILP) and solve it using the Gurobi solver [15]. Our contributions can be summarized as follows:

1) We account for the reliability of workers by making informed and cost-efficient task replication decisions that strive to maximize the total reputation of recruited workers while adhering to a certain budget limit specified for each task.
2) We preserve workers' privacy and encourage their participation in the service by using a black-box approach that does not require any information about their devices. In this approach, we devise a reputation model for workers using beta distribution to assess their performance and reliability by incorporating reputation scores that are continuously updated.
3) We propose the RTAR-Heuristic (RTAR-H) scheme to solve the replication problem in a time-efficient manner. We utilize a variation of the Gale-Shapley algorithm [16], a prominent matching theory algorithm that relies on bipartite graph matching.

The remainder of the paper is organized as follows. Section II highlights some of the recent related works. Section III presents the proposed RTAR and RTAR-H schemes. Section IV discusses the performance evaluation and the simulation results. Section V concludes the paper's future directions.

II. Related Work and Motivation

Several schemes have been proposed to alleviate the effect of sporadic availability and unreliability of workers in edge computing. Such schemes involve the use of task migration and task replication techniques.

In task migration, tasks tend to be transferred from one edge node to another to deal with service interruptions and failure. Task migration can be classified into reactive and proactive migration [17]. In reactive migration, the migration process is triggered after a failure occurs or a need manifests and the goal is to resolve the problem or address the situation as quickly as possible [18], [19], whereas it is performed in advance in proactive migration [17]. In [18], Zhang and Zheng propose a reactive migration scheme that accounts for users' mobility and the risk that users may move too far away from the edge node executing the offloaded task, which can cause increased task failures or prolonged delays. In [19], Saleh and Shastry apply a migration scheme in peer-to-peer networks where workers may leave and join at any time. They reactively move the tasks from a departing worker to another one in the network. However, such migration schemes can trigger excessive latency, overhead, and energy consumption due to task and data transmission from one edge node to another [18], [19]. In addition, they deal with workers' unavailability after a failure already occurs.

Other methods explore the use of task replication in device-based EC to alleviate the effect of workers' unreliability. Amer et al. [12] propose a task replication scheme that strives to maximize the number of replicas assigned for each task. Note that maximizing the number of replicas helps increase the chances of successful task execution in such dynamic computing environments. However, it leads to significantly high recruitment costs and increased wasted resources.

In [13], Amer et al. propose a task replication scheme that minimizes the CPU gap between tasks and target workers. In this scheme, the number of replicas assigned for each task is controlled using a threshold parameter. However, this parameter is randomly specified and the scheme fails to consider the recruitment cost or the reputation of workers. The existing replication schemes also tend to assume some prior knowledge about the workers, thus overlooking their privacy and discouraging them from contributing their resources.

In contrast to existing schemes, we propose a cost-efficient task replication scheme that accounts for workers' reliability and eliminates their privacy concerns by making all decisions without relying on any information about their devices, including their capabilities and network parameters. Towards that end, we determine the reliability of each worker using a reputation score that is continuously updated based on past performances. We use such reputation scores to allocate enough number of workers for each task without exceeding the budget limit.

III. Reputaton-based Task Assignment and Replication (RTAR)

In this section, we present the system model of RTAR, the underlying problem formulation, and the heuristic scheme RTAR-H.

A. System Model and Overview

In RTAR, the system is comprised of three key components: workers, requesters, and an orchestrator. Workers, which are user-owned EEDs, endow their resources to the system in

exchange for incentives granted by the orchestrator. Requesters deploy tasks that need to be offloaded to the orchestrator. The latter is a central entity that is responsible for replicating and allocating incoming tasks to workers while adhering to certain budget constraints. It is worth noting that the orchestrator performs task replication without any prior knowledge about the workers, including their computation capabilities, communication capabilities, and battery levels.

Consider a set of M tasks denoted $\Gamma = \{\gamma_1, \cdots, \gamma_M\}$ and a set of N workers denoted $\mathcal{W} = \{w_1, \cdots, w_N\}$. Each task $\gamma_j \in \Gamma$ is associated with a data size γ_j^{data} in bits, processing density $\gamma_j^{\text{density}}$ in CPU cycles/bit, i.e., the number of CPU cycles required to process a single bit of task's data, a certain computation delay deadline $\gamma_j^{\text{deadline}}$, which is the maximum acceptable computation delay that the task can tolerate, and a certain budget limit γ_j^{budget}. We want to assign and replicate each task γ_j to workers from the set $\mathcal{W}$ if possible.

Each worker $w_i \in \mathcal{W}$ can execute a maximum number of parallel tasks at a time, denoted w_i^{tasks}. Task replication decisions are made by the orchestrator based on the reputation scores of workers. Each worker has a reputation score, denoted w_i^{rep}. The orchestrator assigns a pay value, denoted w_i^{pay}, to each worker based on their reputation score in order to incentivize them to improve their reputation. Intuitively, the worker's reputation score and the payment value assigned by the orchestrator are proportional. Reputation scores are dynamically acquired by the orchestrator based on the workers' past performances observed over time, thus eliminating the risk of fraudulent assessments or fake ratings provided by users.

We use the Beta distribution as a statistical model to estimate and update the reputation scores of workers. The Beta distribution is a flexible and widely used probability distribution that is well-suited for modeling uncertain and bounded variables, such as reputation scores [20]. The Beta distribution can be used to represent uncertainty and variability associated with reputation scores since it allows for the modeling of both the mean and variance of the scores. The Beta distribution can also be employed to update reputation scores in an iterative manner, incorporating new information as it becomes available, making it a valuable tool for dynamic and adaptive reputation systems.

As mentioned earlier, the reputation of the workers follows a probability distribution, specifically the Beta distribution. The reputation model constructs the reputations of workers based on historical ratings, which are initially set by the orchestrator, and continuously updated according to the results of tasks returned by workers. First, the orchestrator c initializes the reputation scores of each worker $w_i^{\text{rep}} \in \mathcal{W}$ to 50%. Then, if a worker w_i succeeds or fails to meet the task's deadline requirements, the worker's reputation is positively or negatively updated accordingly. Note that the allocation process is performed iteratively over rounds of operation, with updates made at each round based on the worker's performance in the previous round. The workers' reputation scores follow a Beta distribution. The random variable of the Beta distribution is denoted p, where $p \sim Beta(\alpha, \beta)$. The Probability Density Function (PDF) of the Beta distribution, denoted $f(p|\alpha, \beta)$, is given by (1) and can be expressed by the gamma function Γ.

$$f(p \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{1}$$

The feedback reported by the orchestrator regarding the outcome of task γ_j executed on worker w_i determines the reputation score update. The feedback can either be positive or negative and is denoted by y or $\bar{y}$, respectively. We denote the observed occurrences of outcome y as r and the observed occurrences of outcome $\bar{y}$ as s. The PDF of observing outcome y can be expressed as a function of the worker's history or past observations. The parameter α_i is updated by incrementing it by 1 whenever the orchestrator receives positive feedback about worker w_i. In particular, α_i is updated using the formula $\alpha_i = r_i + 1$. A positive feedback about some worker w_i is reported when the worker executes a task γ_j successfully within the task's deadline $\gamma_j^{\text{deadline}}$. On the other hand, the parameter β_i is updated by incrementing it by 1 whenever the orchestrator receives a negative feedback about worker w_i indicating the worker's failure to complete the assigned task before the task's deadline $\gamma_j^{\text{deadline}}$.In particular, β_i is updated using the formula $\beta = s_i + 1$. Consequently, the reputation score w_i^{rep} of worker w_i represents the expected value of the reputation model and is thus updated as given by equation (2).

$$\mathbb{E}[\Gamma] = \frac{\alpha}{\alpha+\beta} \tag{2}$$

B. Problem Formulation

In RTAR, the main objective is to maximize the total sum of the reputation scores of all recruited workers for all replicas. We formulate the problem as an Integer Linear Program (ILP), where the binary decision variable λ_{ij} is set to 1 if a replica of task γ_j is assigned to worker w_i, and 0 otherwise. The problem formulation is given by (3).

$$\max_{\lambda_{ij}} \quad \sum_{i=1}^{N}\sum_{j=1}^{M} w_i^{\text{rep}} \lambda_{ij} \tag{3a}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \lambda_{ij} w_i^{\text{pay}} \leq \gamma_j^{\text{budget}} \qquad \forall \gamma_j \in \Gamma \tag{3b}$$

$$\sum_{j}^{M} \lambda_{ij} \leq w_i^{\text{tasks}} \qquad \forall w_i \in \mathcal{W} \tag{3c}$$

$$\lambda_{ij} \in \{0,1\} \qquad \forall w_i \in \mathcal{W}, \ \forall \gamma_j \in \Gamma \tag{3d}$$

The objective given by equation 3a is subject to constraints 3b–3d. Constraint 3b ensures that the total recruitment cost paid to all the workers assigned to task γ_j does not exceed the task's budget limit γ_j^{budget}. This is to guarantee that the replicas associated with task γ_j do not incur more than the budget specified for that. To incentivize the workers to enhance their reputation, the central controller c assigns a pay value w_i^{pay} to each worker based on their reputation score. Clearly, the relation between the worker's reputation score and the payment value assigned by the controller is proportional. Constraint (3c) ensures that each worker is assigned no more than w_i^{tasks} tasks to be executed at a time. To avoid overloading workers, in our simulation, we set the value of w_i^{tasks} to 1. Constraints (3d)

represent the integrality constraints associated with the binary decision variables vector $\boldsymbol{\lambda}$.

The aforementioned problem is an ILP problem, and thus cannot be optimally solved in polynomial time, since ILP problems are generally NP-hard [21]. Thus, we solve it using the Gurobi solver [15]. We also propose the RTAR-H scheme to solve the problem in polynomial time.

C. RTAR-Heuristic (RTAR-H)

Although constrained optimization is a powerful mathematical tool for solving optimization problems optimally, it may not always be practical, especially when dealing with large problem spaces where the dimensionality of the decision variables is high. That is why heuristic approaches are often more practical, as there are many algorithms that can solve a wide range of problems in polynomial time, unlike constrained optimization, which can have exponential complexity.

In RTAR-H, we solve the problem formulated in equation (3) using a prominent algorithm in matching theory, which is based on bipartite graph matching with one-sided preference. Towards that end, we use a modification of the Gale-Shapley algorithm [16]. Note that the Gale-Shapley algorithm was originally developed to solve one-to-one problems. However, some variants of the algorithm are used to solve many-to-one or even many-to-many matching problems.

The steps we use to replicate and allocate the set of tasks Γ to the set of workers $\mathcal{W}$ are illustrated in Algorithm 1. We first sort the workers' reputations in descending order at line (2). The sorted reputations are then used to build preferences matrix $\mathbf{P}$ of the tasks in lines (4–6), where each row in $\mathbf{P}$ represents a task's preferences (i.e., sorted workers' reputations). Note that we assume that all tasks have the same preferences, since each task needs to be assigned to the best available workers, and there are no priority considerations among the tasks. In addition, workers do not have any preferences over the tasks, and the only way for them to be considered in the assignment process is to compel the orchestrator to update their reputations positively. We initialize the set $\hat{\mathbf{w}}$ that is used to keep track of all workers that are yet to be recruited (line 3). The task assignment and replication process is performed iteratively until the stopping criterion is satisfied (lines 7–20). The stopping condition is given by the function ASSIGNABLE at line (7) and is defined by equation (4).

$$\text{ASSIGNABLE}(\mathbf{P}, \Gamma^{\text{budget}}, \hat{\mathbf{w}}) = \begin{cases} \text{True}, & \exists \Gamma_j^{\text{budget}} \in \Gamma^{\text{budget}} \ni \Gamma_j^{\text{budget}} > 0 \\ & \text{and } \exists \mathbf{P}_j \in \mathbf{P} \ni |\mathbf{P}_j| \neq 0 \\ & \text{and } |\hat{\mathbf{w}}| > 0 \\ \text{False}, & \text{otherwise} \end{cases} \tag{4}$$

The function ASSIGNABLE will return true if there is exist at least a task's budget $\Gamma_j^{\text{budget}} > 0$, a task with a non-empty preference list, and at least a non-recruited worker.

The loop (lines 8–19) iterates over each row in $\mathbf{X}$ and does the following: 1) Get the first preference of the task (line 9), 2) deletes the first preference of the task (line 10), 3) ensures that the worker of choice is not recruited yet and its recruitment payment is within the task's budget limit γ_j^{budget} (line 11), 4) assigns worker w to task i at the corresponding element in $\mathbf{X}$ (line 12), 5) updates the task's budget by subtracting the worker's payment value w_i^{pay} (line 13) 6) updates the maximum number of tasks that worker w_j can carry on (line 14), and 7) checks if the number of tasks that the worker can carry on is zero, then the worker should be marked unavailable hence it will be deleted (line 16).

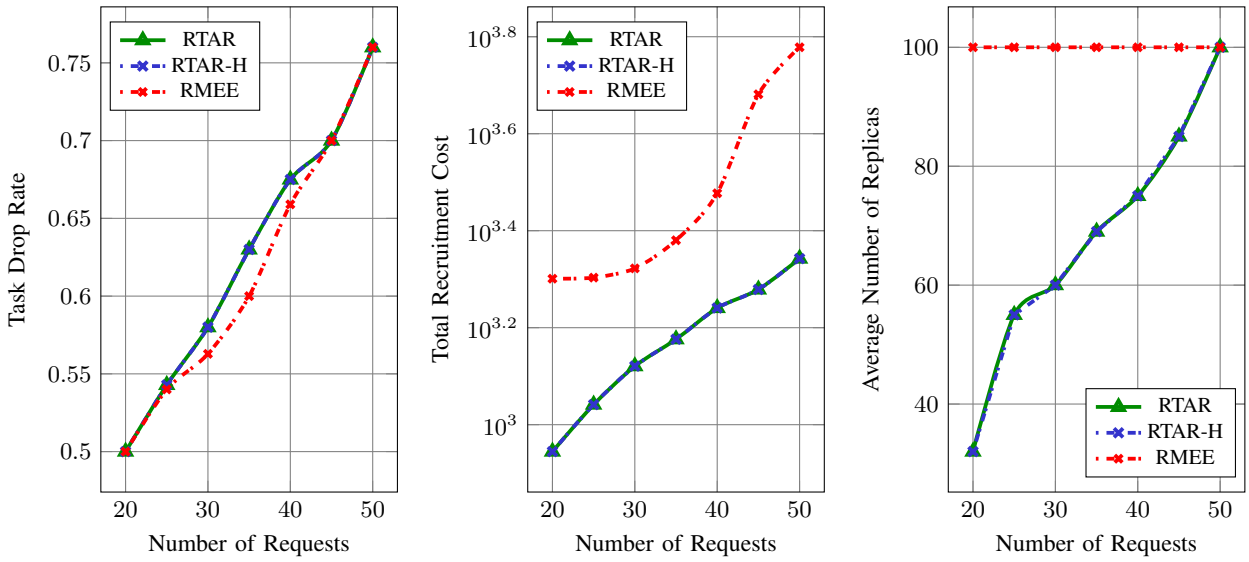
Algorithm 1 RTAR-H at the Orchestrator

Input:
N: number of workers
M: number of tasks
$\mathcal{W}^{\text{rep}}$: a vector of workers' reputations
Γ^{budget}: a vector of tasks' budgets
$\mathcal{W}^{\text{pay}}$: a vector of workers' payment values
$\mathcal{W}^{\text{tasks}}$: a vector representing the maximum number of tasks each worker can carry on
Output:
$\mathbf{X}$: a matrix of assignments of tasks to workers, where the entry $\mathbf{X}_{ji}$ equals 0 or 1

1: **function** RTAR_HEURISTIC(N,M,$\mathcal{W}^{\text{rep}}$,Γ^{budget},$\mathcal{W}^{\text{pay}}$, $\mathcal{W}^{\text{tasks}}$)
2: $s \leftarrow \text{SORT}(\mathcal{W}^{\text{rep}})$
3: $\hat{\mathbf{w}} \leftarrow \{1, \cdots, N\}$ // In inverse order
4: **for** $j \leftarrow 1$ **to** M **do**
5: $\mathbf{P}_j \leftarrow s$ // Populate preference vector
6: **end for**
7: **while** ASSIGNABLE($\mathbf{P}, \Gamma^{\text{budget}}, \hat{\mathbf{w}}$) **do**
8: **for each** $j \in \mathbf{X}$ **do**
9: $w \leftarrow \mathbf{P}_{j0}$
10: DELETE($\mathbf{P}_{j0}$)
11: **if** $w \in \hat{\mathbf{w}}$ **and** $\Gamma_j^{\text{budget}} \geq {\mathcal{W}^{\text{pay}}}_w$ **then**
12: $\mathbf{X}_{jw} \leftarrow 1$
13: $\Gamma_j^{\text{budget}} \leftarrow \Gamma_j^{\text{budget}} - {\mathcal{W}^{\text{pay}}}_w$
14: $w_j^{\text{tasks}} \leftarrow w_j^{\text{tasks}} - 1$
15: **if** $w_j^{\text{tasks}} = 0$ **then**
16: DELETE($\hat{\mathbf{w}}_w$)
17: **end if**
18: **end if**
19: **end for**
20: **end while**
21: **return** $\mathbf{X}$
22: **end function**

IV. Performance Evaluation

We evaluate the performance of RTAR compared to the Replica Maximization at the Extreme Edge (RMEE) scheme [12]. RMEE is a baseline scheme that blindly maximizes the number of replicas without considering the workers' reputation or the task's budget. We also assess the performance of RTAR-H compared to the optimal solution rendered by RTAR. We use the following performance metrics: 1) the task drop rate, which is the ratio of the number of tasks that have not been successfully completed due to workers' failure to the total number of tasks, 2) the total recruitment cost, which is the total amount paid to workers in exchange for their resources, 3) the average number of replicas assigned per task.



(a) (b) (c)

Figure 1: Performance results of RTAR, RTAR-H, and RMEE over varying number of tasks (M).

A. Simulation Setup

We implement RTAR and RMEE schemes using MATLAB and we use the Gurobi solver [15] to obtain the solution of the underlying optimization problem. All the simulation parameters are uniformly distributed. The number of workers is set to 100. The budget allotted to tasks γ_j^{budget} is in the range of [100, 300]. The data size of tasks is in the range of [1,20] bits. The workload intensity of tasks $\gamma_j^{\text{density}}$ is in the range of [1 – 5]$\times 10^2$ cycle/bit, and their deadline $\gamma_j^{\text{deadline}}$ is in the range of [3 – 5] ms. The payments of the workers are determined and updated at each round based on their reputation scores. First, all reputation scores are normalized between 1 and 100 then the payment value is determined based on the reputation score level. For instance, if the reputation score of the worker is between 0 and 20, then the payment value will be 20. An exponential probability of failure is set for each worker, which is associated with the battery drainage of the device. We use the reliability model adopted by Amer et al. [13] to implement the failure probability.

B. Simulation Results and Analysis

We assess the performance of the RTAR, RTAR-H, and RMEE schemes over varying the number of tasks requests.

Fig. 1a depicts the task drop rate of RTAR, RTAR-H, and RMEE. It can be observed that as the number of tasks increases, the drop rate increases for all schemes. This can be attributed to the fact that the number of replicas per task available decreases as the number of tasks increases. Note that RMEE renders the lowest drop rate among all schemes. This is because RMEE disregards the recruitment cost and focuses solely on assigning the maximum possible number of replicas per task. Intuitively, the more replicas per task, the lower the drop rate, since recruiting more replicas reduces the risk of task failure. In contrast, RTAR and RTAR-H account for the recruitment cost and limit the number of replicas based on the available budget for each task. However, RTAR and RTAR-H still provide a comparable drop rate to RMEE, where they closely approach the latter, with a small gap that reaches only 2.4%. This is due to their ability to strike a balance between ensuring reliability by providing multiple replicas per task and avoiding over-provisioning of replicas at the expense of increased recruitment costs and wasted resources. It can also be observed that RTAR-H almost coincides with the optimal solution rendered by RTAR, where it exhibits only a slight discrepancy of up to 1%.

Figure 1b shows the total recruitment cost of each scheme. The total recruitment costs of the RTAR and RTAR-H schemes are lower than that of the RMEE scheme. This is because both schemes recruit workers up to certain budgets which reduces the overall recruitment cost. The RMEE scheme always results in high recruitment costs because the scheme tends to maximize the number of recruited workers. Our scheme yields a significant reduction in recruitment costs by up to 63%. It is also observed that the RTAR-H scheme renders a small gap with the optimal solution of up to 1.2%.

Figure 1c depicts the impact of varying the number of task requests on the number of recruited workers. We observe that the number of replicas recruited by the RMEE scheme is constant for any number of task requests. This is because the RMEE scheme maximizes the number of recruited workers per task and as long as nothing can prevent the scheme from recruiting workers, it will recruit all the available workers within the area. The RTAR and RTAR-H schemes will recruit less number of workers because recruitment is affected by the

budget limits adopted by the schemes. Our scheme outperforms the RMEE scheme resulting in a reduction in the number of workers by up to 68%.

Based on our findings, we have determined that increasing the number of replicas may not always be the best course of action as a surplus of redundant replicas can be superfluous. There are disadvantages to maximizing the number of replicas, such as over-provisioning resources and increasing recruitment costs. Additionally, we have concluded that adopting a recruitment scheme that relies solely on workers' reputation scores, without considering their capabilities, can be advantageous.

V. Conclusion

In this work, we formulated the task assignment and replication problem in a highly uncertain edge computing environment where information about the workers' capabilities is unknown. The performance of the workers is evaluated using inferred reputation scores based on the Beta distribution. The problem was formulated as an Integer Linear Program (ILP), and a polynomial time-efficient heuristic approach was designed based on matching theory. We compared and assessed the performance of our scheme against a baseline scheme. Extensive simulations show that our scheme yields 63% and 68% reduction in recruitment cost and number of replicas, respectively, compared to a baseline scheme that blindly maximizes the number of replicas. Moreover, RTAR-H closely approaches the optimal solution, rendering a small gap of up to 1% and 1.2% in terms of task drop rate and recruitment cost, respectively.

In future work, we plan to include task priorities, which can significantly affect the distribution of workers across tasks.

Acknowledgment

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.

References

[1] "The Growth in Connected IoT Devices is Expected to Generate 79.4ZB of Data in 2025." [Online]. Available: https://www.businesswire.com/news/home/20190618005012/en/The-Growth-in-Connected-IoT-Devices-is-Expected-to-Generate-79.4ZB-of-Data-in-2025-According-to-a-New-IDC-Forecast,%202019.

[2] "Artificial Intelligence Market Worth $1,811.75 Billion By 2030." [Online]. Available: https://www.grandviewresearch.com/press-release/global-artificial-intelligence-ai-market

[3] "Global Natural Language Processing (NLP) Market Size to." [Online]. Available: https://www.globenewswire.com/en/news-release/2023/01/11/2587125/0/en/Global-Natural-Language-Processing-NLP-Market-Size-to-Reach-49-4-billion-by-2027-Growing-at-a-CAGR-of-25-7-Report-by-MarketsandMarkets.html

[4] R. W. Coutinho and A. Boukerche, "Design of Edge Computing for 5G-Enabled Tactile Internet-Based Industrial Applications," *IEEE Communications Magazine*, vol. 60, no. 1, pp. 60–66, 1 2022.

[5] H. Peng, P. C. Chen, P. H. Chen, Y. S. Yang, C. C. Hsia, and L. C. Wang, "6G toward Metaverse: Technologies, Applications, and Challenges," *APWCS 2022 - 2022 IEEE VTS Asia Pacific Wireless Communications Symposium*, pp. 6–10, 2022.

[6] C. Yang, P. Liang, L. Fu, G. Cui, F. Huang, F. Teng, and Y. A. Bangash, "Using 5G in smart cities: A systematic mapping study," *Intelligent Systems with Applications*, vol. 14, p. 200065, 5 2022.

[7] S. Hakak, T. R. Gadekallu, P. K. R. Maddikunta, S. P. Ramu, P. M, C. De Alwis, and M. Liyanage, "Autonomous vehicles in 5G and beyond: A survey," *Vehicular Communications*, vol. 39, p. 100551, 2 2023.

[8] D. De, A. Mukherjee, and D. Guha Roy, "Power and Delay Efficient Multilevel Offloading Strategies for Mobile Cloud Computing," *Wireless Personal Communications*, vol. 112, no. 4, pp. 2159–2186, 6 2020. [Online]. Available: https://link-springer-com.proxy.queensu.ca/article/10.1007/s11277-020-07144-1

[9] S. M. Oteafy and H. S. Hassanein, "IoT in the Fog: A Roadmap for Data-Centric IoT Development," *IEEE Communications Magazine*, vol. 56, no. 3, pp. 157–163, 3 2018.

[10] "Distributive." [Online]. Available: https://kingsds.network/

[11] C. Li, Y. Zhang, X. Gao, and Y. Luo, "Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing," *Journal of Parallel and Distributed Computing*, vol. 166, pp. 15–31, 8 2022.

[12] I. M. Amer, S. M. Oteafy, S. A. Elsayed, and H. S. Hassanein, "QoS-based Task Replication for Alleviating Uncertainty in Edge Computing," *2022 IEEE Global Communications Conference, GLOBECOM 2022 - Proceedings*, pp. 5147–5152, 2022.

[13] I. M. Amer, S. M. Oteafy, and H. S. Hassanein, "Task Replication in Unreliable Edge Networks," *Proceedings - Conference on Local Computer Networks, LCN*, pp. 173–180, 2022.

[14] P. Dai, B. Han, X. Wu, H. Xing, B. Liu, and K. Liu, "Distributed Convex Relaxation for Heterogeneous Task Replication in Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, 2022.

[15] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2021. [Online]. Available: https://www.gurobi.com

[16] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[17] A. M. Zaki and S. Sorour, "Proactive Migration for Dynamic Computation Load in Edge Computing," *IEEE International Conference on Communications*, vol. 2022-May, pp. 4275–4280, 2022.

[18] C. Zhang and Z. Zheng, "Task migration for mobile edge computing using deep reinforcement learning," *Future Generation Computer Systems*, vol. 96, pp. 111–118, 7 2019.

[19] E. Saleh and C. Shastry, "Task Migration in Volunteer Computing Systems," *Proceedings - 2022 4th International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2022*, pp. 2076–2079, 2022.

[20] A. Josang and R. Ismail, "The beta reputation system," in *Proceedings of the 15th bled electronic commerce conference*, vol. 5, 2002, pp. 2502–2511.

[21] P. Raghavan, "Probabilistic construction of deterministic algorithms: Approximating packing integer programs," *Journal of Computer and System Sciences*, vol. 37, no. 2, pp. 130–143, 10 1988.