Timely Processing and Offloading of Short Term Tasks in the Internet of Things

Umut Guloglu
Inst. of Comm. and Navigation
German Aerosp. Center (DLR), Germany
email: umut.gueloglu@dlr.de

Leonardo Badia
Dept. of Information Engin.
University of Padova, Italy
email: badia@dei.unipd.it

Andrea Munari
Inst. of Comm. and Navigation
German Aerosp. Center (DLR), Germany
email: andrea.munari@dlr.de

***Abstract*—Numerous Internet of things (IoT) applications demand accurate and timely information to enable effective actuation. Nonetheless, in computation-intensive status update systems or data gathering systems, the capabilities of IoT devices may fall short in computing/acquiring data with high accuracy, thereby requiring multiple trials before success. Offloading data computation or acquisition tasks to robust units mitigates this inaccuracy. However, these units can be positioned far from the user, and latency becomes an issue for offloading due to some factors, such as propagation delays and resource sharing with other users. This paper investigates the optimization of information freshness for short term tasks, introducing a cost function representing the staleness of the recently obtained data that is accurate enough for actuation. The problem is cast as a Markov decision process and solved through finite-horizon dynamic programming. We find interesting trends of the resulting optimal policy, which can lead to relevant applications for many IoT scenarios.**

***Index Terms*—Edge computing, Age of information, Dynamic programming, Internet of Things.**

I. Introduction

The increase in the digitization of the society pushes Internet of Things (IoT) applications to become more oriented towards more pervasive real-time services in many everyday situations [1]. For example, an increased availability of smart traffic management systems can be expected, using real-time data from connected road sensors to optimize traffic flow and reduce congestion [2]. Similarly, in production settings, an evolution towards Industry 4.0 is made possible by IoT-enabled machinery that monitors and reports performance metrics in real time, enabling predictive maintenance and minimizing downtime [3]. Finally, e-health applications may leverage monitoring devices continuously measuring patients' vital signs, allowing for immediate medical intervention when necessary [4].

These few examples also highlight that, while real-time applications may provide extremely useful services and improve the end user quality of life, they cause at the same time an increase of generated data at the network's edge. Not only do data become more and more variegate, but also they have requirements about timeliness, whose general direction can be expressed through age of information (AoI) [5]–[7]. This metric represents the time elapsed since the generation of the last successfully processed data. Indeed, real-time IoT applications must perform control operation in the cyber-physical realm based on up-to-date information [8].

Hence, this scenario poses a general challenge for data management and exchange, which becomes even more acute if preprocessing is required, e.g., for AI-empowered systems making real-time decision on the exchanged data [9]. This corresponds to whether IoT devices ought to process the information *in situ*, based on their only computational resources, which are often limited, or offload to more powerful remote servers [10]. The latter are sometimes seen as being located in the cloud, but for the purpose of our discussion, they can as well be edge servers, still more powerful and reliable than the individual IoT nodes that are strongly constrained in terms of computational resources.

For this reason, we will generically distinguish between local or remote processing [11]. The former makes only use of the computational resources available at the device, which is a fast but often inaccurate action that may require multiple trials for completing a processing task. Conversely, the latter calls up to a more powerful remote server for handling the task, which can be seen as guaranteeing success at the first attempt, yet implying longer delays. These can be due to longer propagation delays for physically distant servers, or to the fact that the remote computing unit is shared by other users and therefore may lead to queueing delays [12].

This choice between local processing and remote offloading is often addressed in the literature [13], but from the perspective of simpler objectives such as throughput maximization or minimization of the outage probability under delay constraints. While these are certainly important aspects for IoT systems, we investigate the problem through the lens of information freshness. Moreover, most of the existing investigations explore the case of an infinite horizon scheduling (and, even when they consider AoI, this often translates into the long-term average AoI minimization). We argue that, for the very same reasons, it would be more poignant to consider a finite-horizon, which may be more fitting for task-driven operations over a short time span.

Thus, the contribution of the present paper is to propose a timeliness-based model, where we can employ both local and remote computing resources (as well as idling) and consider a

A. Munari acknowledges the support of the Federal Ministry of Education and Research of Germany in the programme of "Souverän. Digital. Vernetzt." Joint project 6G-RIC, project identification number: 16KISK022.

finite-horizon decision setting. We address the optimization as a Markov decision problem, in which the user leverages the current cost function value. We solve the problem through dynamic programming [14], obtaining an optimized setting that achieves superior performance, and we highlight whenever we need to resort to the entire array of actions at our disposal (remote processing, local processing, or wait).

However, the optimized policy can also be shown to be approached by much simpler randomized policies under certain network parametrizations. Thus, the result is further discussed comparing our stateful approach with a simpler blind, i.e., stationary policy for transmission scheduling. Therefore, depending on the system parameters, this comparison fosters practical implications on IoT applications, determining whether simpler scalable approaches are ultimately possible.

The rest of the paper is organized as follows. Related works are discussed in Section II. The system model is formalized in Section III and further solved in Section IV. Section V shows numerical results, and we conclude in Section VI.

II. Related Work

The concept of AoI was proposed in [5], considering a communication link modeled as a queueing system. This approach has been adopted by other papers in the literature, and blurs the actual scheduling of the updates by considering a more general approach where the offered traffic is the only parameter, and the other element of variability is the queueing discipline at the server [15], [16].

However, if the generation of status updates can be controlled, it may be convenient to apply scheduling to optimize the performance in terms of AoI performance. This may be investigated subject to throughput or energy constraints [6], [17], as well as the optimization itself of the scheduling policy depending on its available information, e.g., whether the scheduling is pre-set and immutable or can be adjusted depending on possible failures in the transmission or other random events [18], [19]. In particular, the main issues considered by these approaches are either random channel erasures due to noise or limited access to the medium, e.g., we are in the presence of multiple sources yet the transmission happens over a collision channel where only one source can transmit at a time.

It is also worth noting that the most common scenario involves users obeying the scheduling decision of a centralized unit. In reality, this may not hold if transmitters are owned by different operators and a game theoretic approach may be desirable instead. For example, [20] studies a competitive setup between two transmitter-receiver pairs, whereas [21] focuses on an adversarial scenario instead. In the former case, the additional agent considered is just symmetric in its desire to access the channel as well, and interference may prevent that they do it together. Instead, the latter paper considers again an additional agent, but with the sole purpose of destroying the communication of the other player. Similar approaches would be applicable to the problem at hand in the present paper, with the adjustments of considering equilibrium solutions instead of optima, as typical of game theory [7].

The further extension that can be considered is to reverse this scheduling perspective and include the choice of the actual server, thereby translating from multiple sources transmitting to the same server, to the case where a source can choose among different available end points of the communication [13], which would correspond to our problem of choosing between a local processor or an edge/cloud server. For this problem, the authors of [22] introduced age of processing (AoP) as an extension to AoI, as they consider a different delay for the servers, which must be accounted for when making the scheduling decision. AoP minimization under the choice of local vs. remote processing was also considered in other contributions, adopting different models and processing constraints.

In [23], a BS prompts the generation of data by one out of multiple users and also determines whether data must be processed locally or at an edge server. The problem is cast as a Markov decision process, which is similar to our approach. However, infinite-horizon is assumed and the problem is approached through Lyapunov optimization. The authors of [24] formalize a similar problem but under the assumption that the system dynamics are unknown to the scheduler, thus the problem is successfully tackled by applying reinforcement learning Finally, [11] considers another state-machine approach based on signal flow graphs, and gives different characteristics to the server options, the remote being farther but also more reliable, as we do here. However, this is again studied only in the context of an infinite horizon, thereby deriving stationary policies.

We argue that a finite horizon may give a more fitting representation for short term tasks, which are quintessential in real-time services. For example, vehicular control is mostly exerted on very specific fine-horizon tasks, relative to the road time of the car [2]. Similarly, industrial automation can be seen as a series of short-term production tasks, related to production cycles [3]. Even IoT scenarios with persistent background monitoring, such as agricultural [12] or e-health [4] applications, may contain short term tasks for specific observations.

We note that the approaches proposed for settings similar to ours, but considering stochastic decision-making over an infinite horizon, end up in assigning a probability to each action so as to minimize the long-term average cost. Such a solution does not require any knowledge on the current state of the system, and solely relies on average values (e.g., the completion time of a remote task and the success of a local computation). From this standpoint, such a policy may be easier to implement, also avoiding the memory and computational burden of determining the optimal online solution. Yet, a stationary policy may be inconvenient if the system dynamics matters, and ought to be kept into account [14]. In the following, we will argue whether this is the case or not, depending on the system parameters found in practical evaluations.

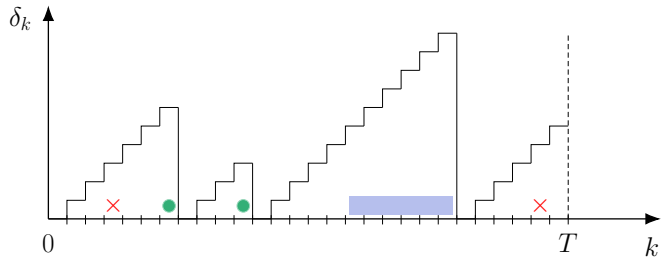

Fig. 1. Example evolution of the cost metric δ_k over time. Green circles (●) denote slots in which a local task is successfully performed, whereas red crosses (×) indicate performing a local task unsuccessfully. Finally, the blue rectangle (■) covers slots over which a task is offloaded to the remote server. Upon completion, the metric is reset.

III. System Model

To capture the key trade-offs in the setting under study, we focus on a slotted timeline. Each slot is of equal duration, and denoted by an index $k \in \mathbb{N}_0$. A reference user (node) aims at executing tasks, defined as atomic activities. A task may involve data collection, processing, or both, depending on the context. We assume that a new task arises as soon as the previous one is successfully completed, so that one atomic activity is always available, and that tasks can be completed either locally at the node or by resorting to a remote server.

Specifically, the user over a slot may be either *idle* or *busy*, where the former condition refers to being ready for processing the next task. Whenever idle at the beginning of a slot, the node may perform one of three actions: (i) attempt to process the next task locally, (ii) offload it, or (iii) wait and not deal with the task at the current slot. In the first case, the local execution is processed within the same slot, yet leads to a successful completion of the task with probability α. Failures may be due for example to lack of sufficient computational capabilities at the node, or to low quality of available information. Regardless of the outcome, the user will be idle again at the start of the next slot. Conversely, an offloaded task is always successfully completed thanks to the stronger capabilities of the remote server. On the other hand, the execution may require more than one slot, e.g., due to resource sharing policies with other nodes, forwarding/re-routing among edge/cloud components, or network propagation delays. To capture this, we assume that the remote task can eventually be completed at each slot with probability p. Accordingly, in case (ii), the node remains busy for a geometric time prior to returning idle. Finally, when a wait action is selected (case (iii)), the user clearly stays idle. The relevance of such choice will become clear later, when limitations to the other actions will be discussed.

To gauge the performance of a user, we introduce a cost metric δ_k, $k \in \mathbb{N}_0$, akin to AoI or AoP [5], [22], defined as

$$\delta_k := k - \sigma_k$$

where σ_k denotes the slot index of the last successful task completion, as of time k. The function portrays the staleness of the most recent task completion, increasing linearly until a new task is completed successfully, when it reset to 0. An example of the time evolution for δ_k is reported in Fig. 1, where actions are assumed to be chosen at the onset of time slots, while the effects are observed at the slot end. For the setting under study, we are interested in determining how the user shall optimally make decisions on which actions to take based on the experienced cost metric. We focus in particular on a finite time horizon of duration T slots, and aim to minimize the average penalty

$$\Delta = \frac{1}{T}\sum_{k=0}^{T} \delta_k. \tag{1}$$

As a practical consideration, we assume that the frequency for attempting local processing is bounded to ε. This could be due to the availability of a subset of the computational resources, e.g., need to process other tasks, as well as to duty cycle or energy consumption constraints. The limitation translates into a condition on the maximum number of local actions performed within the time horizon, which cannot exceed εT.

IV. Optimal Task Scheduling

The problem at hand can be conveniently captured resorting to the framework of Markov decision processes (MDPs) [25]. For the sake of simplicity, we first focus on the case in which the user can attempt local processing without limitations, later extending the approach to the constrained setting.

A. Unconstrained local actions ($\varepsilon = 1$)

The definition of a MDP requires specification of a Markovian system state, a set of control actions together with their rewards (or costs), as well as of existing noise components influencing the state evolution. In the unconstrained setting, the state at time $0 \le k \le T$ can be described by the sole pair $x_k = (\delta_k, \varsigma_k)$, where δ_k is the current value of the cost function, whereas $\varsigma_k \in \{I, B\}$ denotes the condition of the user at the start of the slot, with I indicating idle and B busy, respectively. Control u_k is a ternary choice among attempting local execution of the task (L), offloading it remotely (R), or waiting (W). Not all the actions can be taken in each state, as, for instance, $u_k \in \{L, R\}$ only when $\varsigma_k = I$, i.e., the node is idle. Finally, the noise component is fully characterized by the geometric duration of a remote task completion and the probability of successfully performing a task locally.

Accordingly, the evolution over time of the system state can be readily described, resorting for compactness to the binary r.v. $s_k \in \{0, 1\}$, taking value 1 if a task is completed (either locally or remotely) at the end of slot k, and 0 otherwise. Specifically, we have for $k \in \{0, \dots, T-1\}$

$$\begin{aligned} \delta_{k+1} &= \begin{cases} 0 & \text{if } s_k = 1 \\ \delta_k + 1 & \text{otherwise} \end{cases} \\ \varsigma_{k+1} &= \begin{cases} \varsigma_k & \text{if } \varsigma_k = B \wedge s_k = 0 \\ I & \text{otherwise} \end{cases} \end{aligned} \tag{2}$$

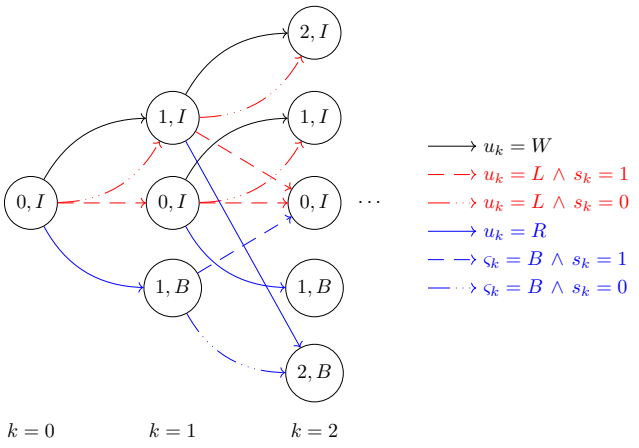

Fig. 2. State transitions for the MDP in the unconstrained case ($\varepsilon = 1$). Solid lines denote deterministic transitions, dashed and dash-dotted lines indicate stochastic transitions leading to a task being completed or not completed in the current slot, respectively. Black, blue and red colors are used to identify wait, remote and local actions, respectively.

The state space of the MDP is shown in Fig. 2, highlighting the possible transitions of the model. The system is initialized in $x_0 = (0, I)$, and the time index advances from left to right, with all potential states of a time slot arranged vertically. In the plot, colors are used to differentiate transitions that follow a wait (black), local (red) and remote (blue) action. Moreover, solid lines denote deterministic transitions, whereas dashed and dash-dotted lines are used to indicate probabilistic transitions that lead or not to a successful completion of the task in the current slot, respectively.

In this setting, the cost incurred over the k-th time slot is simply $g_k(x_k, u_k, s_k) = \delta_k$, with terminal condition $g_T(x_T) = \delta_T$. Accordingly, we aim to determine the optimal stateful control policy minimizing the expected value of the average cost over the finite time horizon T introduced in (1), i.e. $\mathbb{E}[\Delta]$, where the expectation is intended over all the possible realizations of the state transitions. The problem can be solved via dynamic programming for the presented MDP, obtaining the optimal solution via backward induction [25]. The solution has a computational complexity $O(T^2)$, and the size of the generated dynamic programming look-up tables is proportional to T^2.

B. Constrained local actions ($\varepsilon < 1$)

Whenever $\varepsilon < 1$, the user faces a constraint on the number of times it can attempt to perform a task locally, which is bounded over the time horizon of interest to a maximum of $\Lambda := \lfloor \varepsilon T \rfloor$. To tackle this setup, we consider an extended version of the MDP introduced earlier, where the system state is captured by the triplet $x_k = (\delta_k, \varsigma_k, \lambda_k)$, and $\lambda_k \in \{0, \dots, \Lambda\}$ denotes the number of possible attempts at local completion of a task which are remaining at the start of slot k. The system has initial state $x_0 = (0, I, \Lambda)$. Whenever $u_k = L$, λ_k decreases by 1, regardless of whether the task is completed successfully or not during the slot, until no local actions remain. At that point, the sole options are to either wait or trigger a remote action when idle. The evolution of the system state is still described by (2), with the added condition

$$\lambda_{k+1} = \begin{cases} \max\{0, \lambda_k - 1\} & \text{if } u_k = L \\ \lambda_k & \text{otherwise.} \end{cases}$$

With these modifications, the problem can again be solved through backward induction, entailing a complexity of $O(T^3)$.

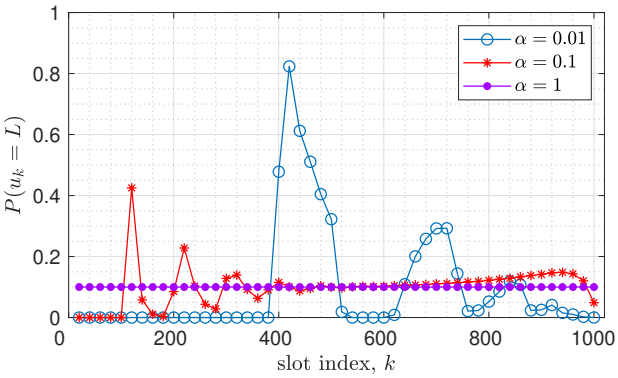

Fig. 3. Empirical probability distribution of selecting a local action at time slot $k \in \{0, \dots, T-1\}$. Different markers denote values of the local task success probability α. In all cases, $p = 0$ (i.e., no remote offloading available), $\varepsilon = 0.1$, and $T = 1000$ slots.

V. Results and Discussion

To understand the structure of optimal task allocation, as well as the achievable performance over a finite time horizon, dedicated numerical studies were performed. In particular, for a given configuration of the system parameters $(\alpha, p, \varepsilon, T)$, the optimal action policy has been computed via dynamic programming, and applied throughout Montecarlo simulations to gauge the expected average cost.

Initial insights on the behavior of the system are offered in Fig. 3, considering the degenerate case $p = 0$. In such conditions, no task would be completed remotely (infinite server response time), and the sole options available to the user at a slot are to attempt a local execution or to remain idle. The plot reports the empirical probability distribution of selecting the former action at each slot, i.e., $P(u_k = L)$, obtained by counting the number of times the action was chosen at time $k \in \{0, \dots, T-1\}$ and by normalizing to the number of Montecarlo runs. Results achieved under three values of the local task completion probability α are shown with different marker styles, considering a time horizon of $T = 1000$ slots, and a maximum number of local attempts $\Lambda = 100$ ($\varepsilon = 0.1$). When task completion within a slot is always successful ($\alpha = 1$), local actions are uniformly distributed over the hole horizon. In this setting, it is indeed easy to verify via simple geometric arguments that a fixed schedule regularly allocating $u_k = L$, for $k = iT/(\Lambda + 1)$, $i \in \{1, \dots, \Lambda\}$ minimizes the average (deterministic) penalty Δ. Conversely, if failures are possible ($\alpha < 1$), local actions tend to be triggered more often towards the center of the overall time interval.

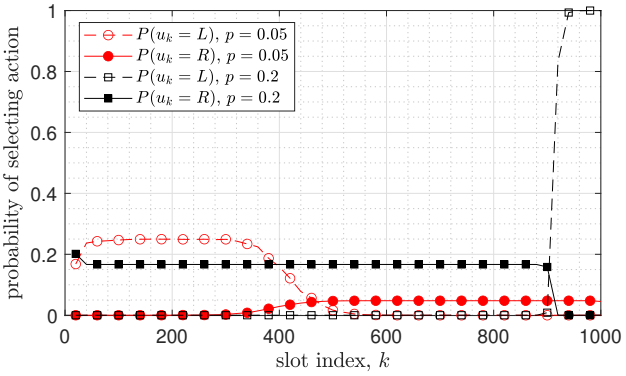

Fig. 4. Empirical probability distribution of selecting a local (empty markers) or remote (filled markers) action at time slot $k \in \{0, \ldots, T-1\}$. Squared and circle markers denote distinct values of p. In all cases, $\alpha = 0.2$, $\varepsilon = 0.1$,$T = 1000$ slots.

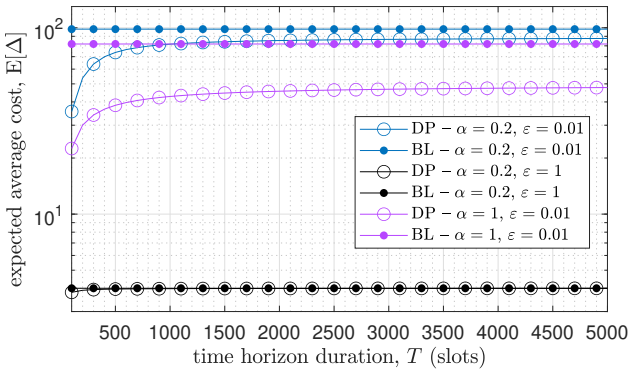

Fig. 5. Expected average cost vs. duration of the time horizon T. Empty markers refer to the optimal strategy derived in Sec. IV, whereas filled markers report the optimal performance attained with the offline stochastic solution in [11]. In all cases, $p = 0.01$.

The trend stems from the attempt of avoiding long periods without successful task completions that could be experienced by favoring placement of (potentially failed) attempts in the initial or final part of the horizon. Incidentally, the trend is consistent with similar results obtained for the scheduling of wireless transmissions over lossy links in [14].

The picture radically changes when task offloading is available ($p > 0$), as pinpointed by Fig. 4. The plot reports again the empirical probability distribution of an action choice, reporting both local processing (empty markers) and remote offloading (filled markers). Two different values of p are considered. In this case, the same time horizon considered before has been used ($T = 1000$ slots), setting $\alpha = 0.2$ and $\varepsilon = 0.1$. Let us first consider the case $p = 0.05$ (circle markers), corresponding to having a longer average response time from the remote server ($1/p = 20$ slots) compared to the average time needed to successfully completing the task locally by repeated attempts ($1/\alpha = 5$ slots). In such conditions, local actions are favored in the initial phases of the horizon. The strategy aims to avoid long time bursts spent in busy conditions, waiting for a task completion without the possibility to react by attempting a local processing, which would lead to high costs from the beginning.[1] As time progresses, and the impact of such bursts on the average cost diminishes, remote actions start being triggered more often, also in view of the potential exhaustion of the limited local processing opportunities. The structure of the optimal solution changes as p increases. This is exemplified by the setting $\alpha = p$ (square markers), where equal average performance of local and remote computations leads to lean more on the latter, saving local actions for the final part of the time horizon to avoid an early depletion. The situation would be further exacerbated for even shorter average completion times of offloaded tasks ($p > \alpha$). This case, not reported in the plot for the sake of clarity, results in a vanishing probability of triggering local actions, resorting to the more convenient remote resources.

[1]Recall that a local action can be triggered only if the node is idle.

Now, we consider the expected average cost that can be attained by the optimal task allocation policies derived in Sec. IV. To gather proper insights, we compare the informed decision making obtained via dynamic programming (DP) to a reference benchmark in the literature [11], which considers an infinite horizon and derives stationary (stateless) policies. This will thereafter be denoted as the blind (BL) scheme.

The results obtained with the DP (empty markers) and BL (filled markers) strategies are reported in Fig. 5. Here, the expected average cost $\mathbb{E}[\Delta]$ is shown against the duration of the finite time horizon T, considering different values of α and ε. In all cases, $p = 0.01$. The values shown in the plot for BL have been obtained picking the optimal action choice probabilities as computed in [11], and correspond to the expected value of the long term average of the cost. The performance of the scheme remains constant along the x-axis, as the solution is not affected by T. Consider first the unconstrained case ($\varepsilon = 1$), where the same behavior is exhibited by DP and BL. For the configuration under study, local actions are convenient over remote offloading, and, in the absence of limitations, both solutions consistently resort to computing locally, attaining minimum average cost. Interesting trends emerge in a constrained setting ($\varepsilon = 0.01$). First, the average cost by DP increases with T. This is due to the initial transient (recall that the system is initialized with $\delta_0 = 0$), and progressively vanishes as the time horizon widens. More importantly, the plot reveals the power of informed decision-making, as the user can strategically utilize local actions. For both $\alpha = 0.2$ and $\alpha = 1$, DP outperforms the blind decision approach, with a cost reduction of up to 60% in the former case for large values of T.

Finally, we explore the impact of different system parameters on the expected average cost under the optimal task scheduling policy. We do so in Fig. 6, reporting $\mathbb{E}[\Delta]$ against α, considering for DP a time horizon of $T = 3000$ slots and $\varepsilon = 0.01$. Focus first on the reference case $p = 0$ (solid lines), where the user solely resorts to (a finite number of) local actions. Here, besides DP and BL, we also

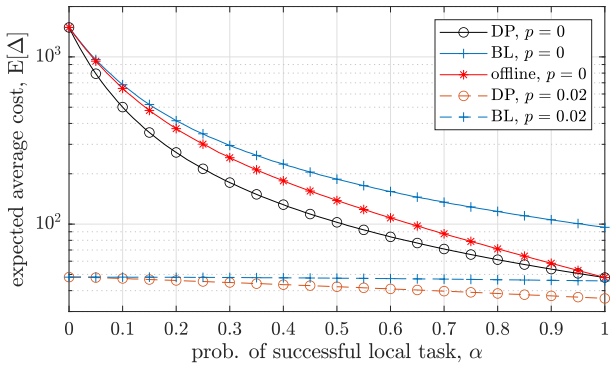

Fig. 6. Expected average cost vs. local success probability α for different values of p. In all cases, $\varepsilon = 0.01$. For the BP and *offline* schedule, $T = 3000$.

consider as additional benchmarks an *offline* method, which simply schedules beforehand local computations, spreading them evenly over the time horizon. In extreme cases ($\alpha \to 0$, $\alpha = 1$), the offline approach behaves as the DP, as the lack of stochastic fluctuations does not provide any advantage to an online adaptation based on the current cost level. Conversely, even when local tasks are always successful, the BL solution offers worse performance, due to the probabilistic allocation of tasks, which departs from the optimal (regular) schedule. In the more practical and interesting intermediate cases ($0 < \alpha < 1$), the advantage of an online adaptation is apparent. Fig. 6 also reports results for the case $p = 0.2$. Notably, BP and BL offer similar performance as long as $\alpha < p$, as the user constantly resorts to task offloading. However, as α increases and local actions become more advantageous, the DP approach starts to yield better results compared to blind decision-making.

VI. Conclusions

We investigated finite horizon minimization of freshness metrics in the presence of multiple available endpoints for data processing. We formalized the problem as a MDP and we solved it via dynamic programming.

Albeit the resulting optimized policy is clearly superior to the alternatives, depending on the scenario it can also be approximated by much simpler randomized policies under certain network configurations. This reveals practical implications for IoT applications, indicating that, depending on system parameters, simple, more scalable approaches may be feasible.

As for future work, other original aspects related to discording control agents, thereby leading to game theoretic approaches [7], [20], can be included in the analysis. Another interesting avenue for further exploration is the application of reinforcement learning or similar techniques [24] to capture the variability of network parameters, which may not be know a priori.

References

[1] X. He, Q. Ai, J. Wang, F. Tao, B. Pan, R. Qiu, and B. Yang, "Situation awareness of energy Internet of things in smart city based on digital twin: From digitization to informatization," *IEEE Internet Things J.*, vol. 10, no. 9, pp. 7439–7458, 2022.

[2] A. Rolich, I. Turcanu, A. Vinel, and A. Baiocchi, "Impact of persistence on the age of information in 5G NR-V2X sidelink communications," in *Proc. IEEE MedComNet*, 2023.

[3] X. Xie, H. Wang, and X. Liu, "Scheduling for minimizing the age of information in multisensor multiserver industrial Internet of things systems," *IEEE Trans. Ind. Informat.*, vol. 20, no. 1, pp. 573–582, 2024.

[4] Z. Ning, P. Dong, X. Wang, X. Hu, L. Guo, B. Hu, Y. Guo, T. Qiu, and R. Y. Kwok, "Mobile edge computing enabled 5G health monitoring for Internet of medical things: A decentralized game theoretic approach," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 2, pp. 463–478, 2020.

[5] S. K. Kaul, R. D. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. IEEE Infocom*, 2012.

[6] C. Kam, S. Kompella, G. D. Nguyen, J. E. Wieselthier, and A. Ephremides, "On the age of information with packet deadlines," *IEEE Trans. Inf. Theory*, vol. 64, no. 9, pp. 6419–6428, 2018.

[7] L. Badia and A. Munari, "A game theoretic approach to age of information in modern random access systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM) Workshops*, 2021.

[8] M. A. Abd-Elmagid, N. Pappas, and H. S. Dhillon, "On the role of age of information in the Internet of things," *IEEE Commun. Mag.*, vol. 57, no. 12, pp. 72–77, 2019.

[9] Y. Xu, M. Xiao, Y. Zhu, J. Wu, S. Zhang, and J. Zhou, "AoI-guaranteed incentive mechanism for mobile crowdsensing with freshness concerns," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4107–4125, 2023.

[10] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 1–23, 2019.

[11] A. Munari, T. De Cola, and L. Badia, "Local or edge/cloud processing for data freshness," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2023, pp. 01–06.

[12] L. Badia, A. Zancanaro, G. Cisotto, and A. Munari, "Status update scheduling in remote sensing under variable activation and propagation delays," *Ad Hoc Networks*, vol. 163, p. 103583, 2024.

[13] C. Li, S. Li, Y. Chen, Y. T. Hou, and W. Lou, "Minimizing age of information under general models for IoT data collection," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2256–2270, 2019.

[14] A. Munari and L. Badia, "The role of feedback in AoI optimization under limited transmission opportunities," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2022.

[15] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Trans. Inf. Theory*, vol. 65, no. 8, pp. 5215–5232, 2019.

[16] M. Moltafet, M. Leinonen, and M. Codreanu, "On the age of information in multi-source queueing models," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 5003–5017, 2020.

[17] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1359–1372, 2019.

[18] Y.-P. Hsu, E. Modiano, and L. Duan, "Scheduling algorithms for minimizing age of information in wireless broadcast networks with random arrivals," *IEEE Trans. Mobile Comput.*, vol. 19, no. 12, pp. 2903–2915, 2020.

[19] A. Arafa, J. Yang, S. Ulukus, and H. V. Poor, "Timely status updating over erasure channels using an energy harvesting sensor: Single and multiple sources," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 1, pp. 6–19, 2021.

[20] G. D. Nguyen, S. Kompella, C. Kam, J. E. Wieselthier, and A. Ephremides, "Information freshness over an interference channel: A game theoretic view," in *Proc. IEEE Infocom*, 2018.

[21] S. Banerjee and S. Ulukus, "Age of information in the presence of an adversary," in *Proc. IEEE Infocom Wkshps*, 2022.

[22] R. Li, Q. Ma, J. Gong, Z. Zhou, and X. Chen, "Age of processing: Age-driven status sampling and processing offloading for edge-computing-enabled real-time IoT applications," *IEEE Internet of Things Journal*, vol. 8, no. 19, pp. 14 471–14 484, 2021.

[23] S. Jayanth and R. V. Bhat, "Age of processed information minimization over fading multiple access channels," *IEEE Trans. Wireless Commun.*, vol. 22, no. 3, pp. 1664–1676, 2022.

[24] X. He, S. Wang, X. Wang, S. Xu, and J. Ren, "Age-based scheduling for monitoring and control applications in mobile edge computing systems," in *Proc. IEEE Infocom*, 2022.

[25] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont, MA, USA: Athena Scientific, 2005, vol. I.