

A Framework for Sustainable Management of Autonomous Ground Robot Fleets

Syeda Tanjila Atik, Daniel Grosu, *Senior Member, IEEE*, and Marco Brocanelli, *Senior Member, IEEE*

Abstract—Ensuring low battery degradation in Autonomous Ground Robot (AGR) fleets operating in online environments (e.g., delivery services) is essential for enhancing their long-term sustainability. However, most existing studies either rely on offline methods—unsuitable for scenarios requiring real-time decisions—or focus solely on maximizing task allocation, resource utilization, or revenue, with limited consideration for battery health. Additionally, maximizing fleet sustainability requires bounded relative revenue losses from unassigned tasks within a user-defined acceptable limit to make it an attractive option for industry. To address these limitations, we propose an online task and charge allocation framework that jointly optimizes revenue generation and battery lifespan, while allowing users to explicitly constrain relative revenue losses. The framework includes three event-driven algorithms: **BTC-M**, which computes optimal decisions at each event, and two computationally efficient greedy variants, **BTC-G** and **BTC-WG**, which provide sub-optimal solutions with reduced overhead. We evaluate the performance of our approach under different task arrival distributions representative of real-world applications. Simulation results based on a real AGR, compared against multiple baselines, demonstrate that our framework can extend battery lifespan by up to 19% with minimal revenue loss.

I. INTRODUCTION

The presence of autonomous ground robots (AGRs) has increased significantly in various sectors of our society. For example, AGRs are employed to transport packages between various stations in Amazon Robotics sortation hubs [1] while Grubhub has partnered with robot delivery companies such as Starship [2] and Cartken [3] to deploy their full service in many US university campus areas [4]–[7], which exponentially increased the completed deliveries from 500K in 2020, 2M in 2021, to 5M in 2023 [2]. Unlike offline scenarios where the set of tasks is predetermined and known in advance, many scenarios operate under an online setting, where the characteristics of incoming tasks—such as task value, starting and ending locations—are only revealed upon their arrival, necessitating immediate allocation decisions. To ensure smooth and reliable operations, a critical aspect for any AGR is the management of its battery. Over time, the maximum capacity of rechargeable batteries, defined as the highest amount of energy they can store, gradually declines, eventually rendering the battery unusable when this capacity becomes too low. Conventionally, a battery is considered to

have reached the end of its useful life once its maximum capacity falls to approximately 80%, as subsequent usage tends to accelerate degradation at a faster rate [8]. The pattern of battery degradation is closely tied to how the task allocation and recharge scheduling of the AGR fleet are managed. If not carefully coordinated, the growing deployment of AGRs across various sectors could significantly increase the demand for battery replacements, leading to sustainability concerns. Therefore, it is essential to develop online algorithms that not only aim to maximize the total revenue but also minimize the adverse impact on the battery health of the entire AGR fleet.

The majority of the existing works [9]–[14] mostly focuses on optimizing task allocation, which is similar to maximizing the total revenue. However, these approaches are mainly designed for offline scenarios where task characteristics and other parameters are known in advance. A few other studies concentrate on maximizing task allocation in online scenarios [15], [16]. However, they often overlook the long-term impacts of the task scheduling decisions on the battery health. Our previous studies (e.g., [17], [18]) consider the joint task and charge scheduling problem focusing on both allocation and battery health maximization. However, they only consider offline scenarios and only use energy thresholds to reduce the impact of allocation and charging decisions on the battery health. In reality, different AGRs have different degradation levels, which cannot be captured accurately by such threshold-based approaches. Thus, none of the above works provide a solution to the problem of *online battery lifespan-aware task and charge scheduling for AGR fleets*, specially considering the actual estimation of how scheduling decisions affect the capacity degradation of each individual AGR over time.

In the conference version of this paper [19], we addressed this problem through the design of a family of algorithms **BTC-X**, which included an optimal but slower version **BTC-M** and a greedy but faster version **BTC-G** to coordinate fleet operations. However, the proposed solution has the following limitations. First, the framework does not allow users to define a maximum bound of revenue losses. To strengthen companies' confidence in integrating sustainable practices it would be beneficial to allow maximizing the fleet sustainability while bounding the revenue losses within a user-defined acceptable limit. Second, the proposed greedy algorithm does not adequately handle situations in which some AGRs cannot perform most available tasks due to energy limitations, leading to reduced revenues. Third, the performance evaluation in [19] considered only a single task arrival distribution (i.e., bimodal) inspired by a food delivery application. However, in other scenarios tasks may arrive at a different rate (e.g., uniform [20],

S. T. Atik and M. Brocanelli are with the Department of Electrical and Computer Engineering, Ohio State University, Columbus, USA. E-mail: atik.4@osu.edu, brocanelli.1@osu.edu. D. Grosu is with the Department of Computer Science, Wayne State University, Detroit, USA. E-mail: dgrosu@wayne.edu

This research was supported by the US National Science Foundation under Grant CNS-1948365

[21]) throughout the day. Therefore, it remains unclear how the performance of our algorithms may vary across different task arrival distributions.

In this paper, we present an improved version of the online Battery Degradation-Aware Task and Charge Allocation framework. The proposed framework integrates a model that helps maximize the fleet sustainability and revenues while ensuring that the relative revenue losses remain within the user-defined acceptable limit. Furthermore, we introduce a weighted greedy approach, referred to as BTC-WG, which enhances task allocation efficiency compared to the previously proposed greedy approach, BTC-G, by considering the variation in the number of executable tasks for each AGR. The solution found by BTC-WG is better than BTC-G and slightly worse than BTC-M. However, BTC-WG has an execution time that is faster than BTC-M but slightly slower than BTC-G. Finally, we conduct a comprehensive evaluation of the BTC-M framework under various task arrival distributions and relative revenue loss bounds to study their impact on the fleet lifespan.

Specifically, this paper makes the following contributions:

- We formulate the online sustainability-aware task and charge allocation problem and develop a family of algorithms, referred to as BTC-X, capable of computing a solution in polynomial time at each event occurrence.
- We design three algorithms of the family to find a solution to the formulated problem, BTC-M, BTC-G and BTC-WG. While BTC-M computes an optimal solution, it may incur higher computational costs compared to BTC-G and BTC-WG, which employ a greedy and weighted greedy strategy to efficiently obtain a sub-optimal solution.
- We leverage a real AGR to generate realistic simulated scenarios. Compared to several baselines, our extensive evaluation shows that both BTC-M and BTC-WG can achieve up to 19% longer battery lifespan while bounding revenue losses within acceptable limits.

The rest of the paper is organized as follows. Section II describes the related work, Section III presents the problem formulation and the solution design. Section IV shows the experimental results, Section V provides suggestions for future research and Section VI concludes the paper.

II. RELATED WORK

Maximizing Task Allocation. Jeon et al. [9] aimed at maximizing the number of tasks completed by assigning each task to the nearest AGR, thus minimizing travel distance. Liu et al. [10] proposed two offline Multi-Agent Pickup-and-Delivery (MAPD) algorithms for task and path planning of AGRs based on the traveling salesman problem. In [11], task scheduling approaches are proposed for order picking and moving tasks in a warehouse considering the tasks to maintain a static order. Some solutions proposed approaches based on market [12]–[14], [22], reinforcement learning [23], and graphs [21] for task allocation. However, they are only designed for *offline* scenarios where task characteristics are known in advance.

A few other studies focused on *online* algorithms for maximizing task allocation [15], [16]. For example, Agarwal

and Sarkar [15] proposed an online task scheduling algorithm to maximize the completion of tasks within soft deadlines while minimizing penalties for late execution. However, none of the above offline and online approaches consider the recharge scheduling of the AGRs or the long term impact of task scheduling decisions on the AGR's battery lifespan.

Minimizing Battery Degradation. Considering the environmental sustainability in battery-operated mobile computing, several studies proposed various strategies aimed at extending the battery lifespan [24]–[26]. For example, some solutions [24] proposed algorithms that reduce the discharge rate and leverage voltage scaling to alleviate battery degradation while meeting task deadlines. Kwak et al. [25] proposed a task scheduling approach where the battery temperature is controlled to minimize the degradation rate. He et al. [26] proposed a charging strategy that adjusts rates based on user availability while allowing periodic relaxation to extend battery life. Unlike our approach, these studies focus on battery life extension without considering AGR coordination for online task and charge scheduling, which is vital for optimizing performance and battery health.

Joint Task and Charge Scheduling. Focusing on the joint task and charge scheduling problem, Chen and Xie [27] proposed an approach that integrates task allocation, routing, and charging to simultaneously reduce total energy consumption, overall service time, and total energy charged. Shi et al. [28] proposed an optimized charging schedule for robot fleets to maximize their overall operational profit with the consideration of the battery degradation, nonlinear charging profile, and electricity cost. However, their model simplifies the scenario by assuming that all tasks generate the same revenue and consume the same amount of energy, which may not reflect some real-world scenarios (e.g., AGR-based goods delivery). Our previous works in [17], [18] presented offline joint task and charge scheduling algorithms for AGRs considering only energy thresholds in the optimization problem to calculate the cost of battery degradation while recharging or discharging. However, they do not consider the actual capacity degradation of individual AGRs, failing to distinguish between AGRs with new and aging batteries. Moreover, these approaches assume full prior knowledge of tasks, limiting their applicability to online scenarios, and do not account for location-specific battery degradation effects while charging at charging stations, unlike our proposed method. *To the best of our knowledge, this is the first paper that provides a solution to the problem of **online** battery lifespan-aware task and charge scheduling for AGR fleets.*

III. PROBLEM FORMULATION AND SOLUTION DESIGN

We consider the general scenario of a fleet of AGRs performing tasks such as food or goods delivery within a defined area, either indoors (e.g., warehouse) or outdoors (e.g., university campus), with a fixed number of charging stations. A central coordination system assigns tasks to homogeneous AGRs, which autonomously follow optimal routes provided by a global navigation system (route generation is outside this paper's scope and covered in prior work [10], [29], [30]). Real-world examples include university campuses operating fleets

of AGRs (e.g., 40 AGRs [31]) for food delivery, and modern warehouses using AGRs to sort and transport packages [32], [33]. *In this paper, we propose to formulate the online battery lifespan-aware task and charge scheduling problem as a utility maximization problem, which jointly takes into account the total revenue derived from the allocated tasks along with the battery degradation of the AGRs due to either executing specific tasks or charging at specific times and locations.* In the rest of this section, we briefly describe the general parameters of the problem (Section III-A), how to calculate at runtime the energy (Section III-B) and battery degradation (Section III-C) of potential allocation choices to aid online decision making, and the proposed utility functions (Section III-D) before describing the proposed algorithm (Sections III-E and III-F). Finally, in Section III-G we describe the model that enables our algorithm to automatically adapt its parameters to bound the relative revenue losses.

A. General Description of the Parameters

AGR Parameters. A generic AGR i in the fleet at a time point k , i.e., $R_i(k) \in \mathcal{R}$, where $\mathcal{R}$ is the set of AGRs, is characterized by five parameters:

$$R_i(k) = \langle l_i^r(k), e_i(k), d_i(k), t_i^c(k), s_i^r(k) \rangle$$

where $l_i^r(k)$ represents the AGR's current location, $e_i(k)$ is the current remaining energy stored in the battery, $d_i(k)$ is the current level of battery capacity degradation, $t_i^c(k)$ is the current time elapsed from a charging request, and $s_i^r(k)$ is the operational state of the AGR at time k . Specifically, each AGR can be in any of the following four states: *idle*, *busy*, *charging*, and *waiting to recharge*. The *idle* state indicates that the AGR is currently neither charging, executing tasks, or waiting to recharge. A *busy* state indicates that the AGR is assigned and actively executing a task. A *charging* state denotes that the AGR is currently recharging at a charging station, while a *waiting-to-recharge* state implies that the AGR is currently waiting in the recharge queue of a charging station. While some of these parameters are easily obtainable from the AGR, e.g., location, state, current energy, and time elapsed from a charging request, some other parameters necessary for online decision making such as the estimated energy and the degradation due to potentially executing a certain task or charging at a specific charging station are non-trivial to obtain. We describe how to obtain them in Sections III-B and III-C.

Charging Station Parameters. Each charging station j , i.e., $C_j \in \mathcal{C}$, where $\mathcal{C}$ is the set of charging stations, is characterized by two parameters:

$$C_j(k) = \langle l_j^c, s_j^c(k) \rangle$$

where l_j^c is the location of the charging station (assumed to be fixed), and $s_j^c(k)$ is the status of the charging station at time k , which can be either *occupied* or *free* depending on whether an AGR is charging or not at that station, respectively.

Task Parameters. Each task h , i.e., $T_h \in \mathcal{T}$, where $\mathcal{T}$ is the set of tasks, arriving at a certain time has six parameters:

$$T_h(k) = \langle l_h^p, l_h^d, v_h, a_h, f_h, s_h^t(k) \rangle$$

where l_h^p and l_h^d represent the start and end locations of task T_h , respectively, v_h is the valuation of the task (e.g., the revenue to successfully execute it), a_h denotes the arrival time, f_h is the absolute deadline by which the task needs to be allocated to any AGR, and $s_h^t(k)$ is the task status at time k , which can be *active* when it arrives, *allocated* when it is being executed by an AGR, or *timed-out* if it cannot be allocated within its deadline. Due to the online nature of the considered scenarios, all these parameters are only known when a task actually arrives. In addition, in this paper we do not make any prediction of future demand, thus the designed algorithm is independent from any shape of task arrival over time. It is in our future work to embed demand prediction in the task and charge scheduling choices.

B. Estimating Energy Consumption

In order to make informed decisions on how tasks and charging choices can affect an AGR battery degradation, it is of primary importance to estimate how a specific task or charging decision would vary the AGR's battery state-of-charge, i.e., the percentage of remaining energy stored. For the charging decisions, we need to consider the energy spent for the AGR to travel from its current location to a specific charging station location, and then the rate of charge. To simplify the problem, we do not consider the effect of variable charging rates on the battery degradation, which is a complementary branch of research, but assume an average charging rate $\bar{r}$. While we can easily profile the power consumption of a real AGR for different speed and CPU/GPU frequency levels, making informed scheduling decisions require the employment of an energy model. Thus, we leverage the following well-established model to estimate the energy consumption E [34]:

$$E = (p + mg\bar{s} \max\{\sin \phi, 0\})\delta \quad (1)$$

where δ is a period of time (e.g., to travel from the current location to the task start location and then to end locations), p is the average power consumption of the AGR while traveling at an average speed $\bar{s}$ using the highest CPU/GPU frequency on a flat surface, m is the mass of the AGR, g is the gravity acceleration, and ϕ is the average ground inclination angle of a specific traveling path. We assume that the ground inclination angle ϕ and the specific route of a task or the one to go to a charging station is known at decision time. In addition, in our experiments with a real AGR (shown in Figure 2) we found that the energy consumption while going downhill does not change much compared to that of going on flat surface, which is why we take the maximum between $\sin \phi$ and 0 to calculate the additional energy consumption due to inclination. Note that, in real-life, each task may lead to low-level variations in energy consumption due to variations in speed and route conditions while traveling a specific path. However, considering these factors in the allocation algorithm increases the problem complexity. Instead, it is in our future work to decompose the problem by handling these low-level variations on each AGR after an allocation decision, by executing the task while maximizing the AGR speed within a task energy budget, which is used for allocation decisions.

Using this energy consumption model and knowing the charging rate $\bar{r}$, we can now estimate at decision time what is the variation in state of charge of the battery due to the potential allocation of a specific task or to travel to a charging station and then recharge until the state of charge reaches a certain level (e.g., 80% to preserve health). If an AGR remains idle for a period of time, we consider the energy of the AGR to be equivalent to that of its computing node (i.e., zero speed). The historic variation of the energy stored in an AGR battery concatenated with a potential energy variation due to a scheduling choice is then used to make informed optimization decisions on which task or charging station to allocate for each AGR in the fleet.

C. Estimating Battery Degradation

A rechargeable battery capacity degrades due to cycle and calendar aging. Cycle aging accounts for the specific charge and discharge cycles during usage, while calendar aging occurs due to the passing of time. Key factors influencing degradation include the average State-of-Charge (SoC), the variance in SoC as measured by the standard deviation, the depth of discharge (DoD), and the operating temperature. There have been extensive studies to model the battery degradation of a specific chemistry. Proposing a new model falls outside of the scope of this paper. Thus, to assess the battery degradation of the AGRs, we use the model proposed in [35], which leverages linear and non-linear cycle and calendar aging models and the rainflow counting algorithm to analyze the amount of capacity faded due to a specific SoC trace of a lithium-ion battery, typical for AGRs. Due to the limited space, we leave out the description of these models and algorithm and remind the interested reader to the details provided in [35]. On the other hand, in our algorithm we abstract the degradation algorithm proposed in [35] through a generic function $D(\cdot)$, which takes as parameters the AGR's unique ID and the candidate task/charging station to be allocated to execute the following steps: (1) use the energy model described in Section III-B to estimate the change in SoC due to the potential allocation of a specific task or charging station, (2) concatenate such estimated trace to the historically recorded AGR SoC trace, thereby generating an *augmented* SoC trace, (3) run the rainflow algorithm [35] using the augmented SoC trace, and (4) return the estimated battery capacity degradation. Since we use the function $D(\cdot)$ as a *plug in* in our algorithm, any other degradation model (e.g., for a different battery chemistry) can also be used without any major changes.

D. Utility Functions

In order to find a solution to the online battery lifespan-aware task and charge scheduling problem, we propose to translate it into the problem of finding a maximum weighted matching between the edges of a bipartite graph. Specifically, we define a set of *utility functions* to be used as weights of the edges of a bipartite graph that has, on one bi-partition, one vertex for each AGR in the idle state at the time of an event (e.g., the arrival of a task), while, on the other bi-partition, it has one vertex for each active task, one vertex for each

free charging station, and one *idle vertex* for each idle AGR. The utility of a task allocation decision (i.e., the weight of the edge from an AGR vertex to a task vertex) evaluates the value of allocating a tasks to an AGR against the cost of the estimated battery degradation. The utility of a charging decision (i.e., the weight of the edge from an AGR vertex to a specific charging station vertex) determines the desired time to start charging and the charging station to minimize the AGR battery degradation. In addition, to account for situations where an AGR does not have tasks to execute and does not need a recharge, for example due to high SoC, we also include a utility for remaining in the idle state.

Upon the occurrence of an event (see Section III-E for a list of the events we consider), a utility matrix U of size $|\mathcal{A}^T| \times (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|)$ is initialized to 0 and is then filled for allocation decisions. The $|\mathcal{A}^T|$ rows correspond to the idle AGRs, the first $|\mathcal{A}^T|$ columns correspond to the available task vertices yet to be allocated, the $|\mathcal{A}^C|$ columns correspond to the free charging station vertices, and the last $|\mathcal{I}|$ columns correspond to the idle vertices (one for each idle AGR). Next, we define the utility functions and describe the algorithm.

Task Utility. The task utility is defined to consider the trade-off between the valuation of the task that can be achieved by executing it and the amount of battery degradation incurred due to executing that task. We thus propose to calculate the utility $U_{ih}(k)$ for allocating task T_h to the AGR R_i at time k as:

$$U_{ih}(k) = \begin{cases} v_h - \beta_1 \cdot D(R_i(k), T_h(k)), & \text{if } e_i(k) \geq e_h^{req} \\ NULL, & \text{otherwise.} \end{cases} \quad (2)$$

where v_h is the valuation of task T_h normalized over an estimated maximum task value. $D(R_i(k), T_h(k))$ is the amount of battery degradation incurred if AGR R_i performs task T_h (see Sections III-B and III-C for details of how we estimate it), which is normalized to the end-of-life capacity degradation, e.g., 20%. β_1 is a multiplicative factor used to adjust the relative importance of preserving battery degradation compared to task revenue maximization. e_h^{req} is the energy necessary to successfully carry out a specific task T_h (calculated as described in Section III-B). If the AGR energy is not enough, that utility is set to NULL, removing the edge between that AGR and that task in the bipartite graph.

Charge Utility. The charge utility is defined to capture the amount of battery degradation that would be incurred due to charging at a particular charging station based on the current location and state of charge of a particular AGR. We propose to calculate the utility $W_{ij}(k)$ for scheduling AGR R_i to charge at station C_j at time k as:

$$W_{ij}(k) = \begin{cases} \beta_2 \cdot (1 - D(R_i(k), C_j(k))) \cdot V(R_i(k)), & \text{if } e_i(k) < \alpha \\ NULL, & \text{otherwise.} \end{cases} \quad (3)$$

where $D(R_i(k), C_j(k))$ is the normalized amount of battery degradation if AGR R_i goes to recharge at charging station C_j , considering the estimated variation in SoC to go to the charging station location and then recharge to a maximum SoC (same for all AGRs). We use $1 - D(R_i(k), C_j(k))$ because

we want the allocation algorithm to minimize degradation by maximizing the total utility. β_2 is another multiplicative factor used to give more or less importance to the objective of charging or task allocation. $V(R_i(k))$ is a function that decreases linearly from 1 to a sufficiently small value (e.g., 0.05) as the SoC increases. This utility function thus helps considering three prioritization factors for charging allocation decisions, including (1) charging decisions that minimize degradation, (2) charging AGRs at a shorter distance to a charging station (considered through the degradation estimation), and (3) charging first AGRs at higher depth of discharges, i.e., with higher $V(\cdot)$ values due to a lower SoC at the event time. To prevent unnecessary recharges when an AGR's energy level exceeds the energy threshold α , the charging utility is set to NULL, removing the edge between that AGR and any free charging station in the bipartite graph. In such cases, the AGR is either allocated a task or remains idle awaiting a task. The value of α affects both generated revenue and AGR battery degradation. As described later in Section III-G, it is automatically regulated by our algorithm to limit the relative revenue losses to a desired bound set by the user.

Idle Utility. The idle utility is defined to keep the AGR in idle state when there is no need for task execution or recharge. For each AGR R_i , the utility of the idle vertex Z_i at time k is simply set to a sufficiently small value λ :

$$Z_i(k) = \lambda \quad (4)$$

This choice ensures that if an AGR's energy exceeds the threshold α , it will remain in the idle state when there are no tasks available. Additionally, if the energy falls below the threshold α and there are neither tasks nor available charging stations, the AGR will also remain in the idle state.

As explained next, after the elements of the utility matrix are determined using the above utility functions, the matrix is used to find the best allocation for the AGRs at the occurrence of a certain set of events.

E. Overview of the BTC-X Family of Online Algorithms

In this section, we provide a detailed description of our family of online Battery degradation-aware Task and Charge allocation algorithms, BTC-X, which leverages either an optimal but slower approach or a greedy but faster one to make decisions. Due to the online nature of the scenarios considered, BTC-X activates at the occurrence of certain scenarios that may require task and charge reallocation. We propose to focus on four main events: (E1) the arrival of a task, (E2) the completion of a task by an AGR, (E3) the completion of recharge by an AGR (by reaching a tunable max SoC level), and (E4) an AGR's SoC is too low to execute any tasks.

Figure 1 shows the high-level overview of the BTC-X during each event. The core of BTC-X is the allocation Algorithm 1, which makes (optimal or greedy) decisions on how to allocate tasks and/or charging stations to AGRs. We provide the details of this algorithm in Section III-F. There are three main ways for the triggering events to require execution of Algorithm 1. First, a new set of tasks arrives (i.e., E1). Second, an AGR completes the execution of a task (i.e. E2) and there

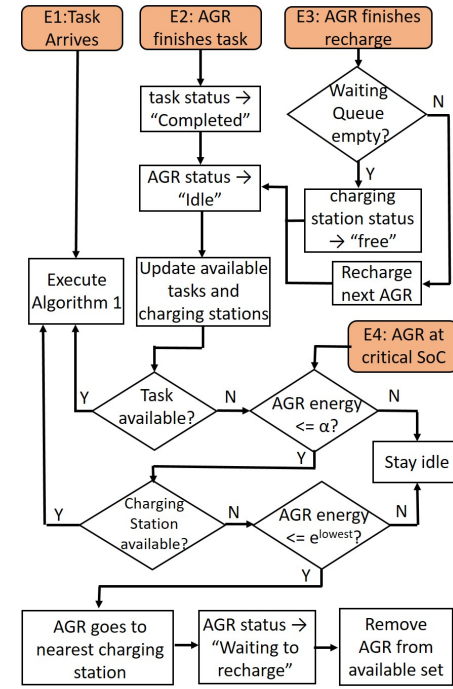


Fig. 1. High level logic of the proposed online BTC-X family of algorithms.

are tasks still available for allocation. This can happen when previous executions of Algorithm 1 left tasks unallocated due to several reasons, including the lack of idle AGRs or a relatively low task valuation compared to a high degradation of the currently idle AGRs. In such cases, the task remains in the set of available tasks until its deadline (and removed afterwards), waiting for a new AGR to become idle through events E2 or E3. This practice is supported in real scenarios such as food delivery (e.g., when requests may not find any drivers) and also in related work on online task allocation [36].

On the other hand, Algorithm 1 can also be executed when there are no tasks to be allocated and the newly idle AGRs have a sufficiently low energy level, i.e., below a threshold α to avoid unnecessary charges at high SoC (e.g., 70%). In such cases, BTC-X checks the energy remaining in the batteries of the idle AGRs to determine which ones could be scheduled for recharge, which can occur in Figure 1 through events E2, E3, or E4. Because multiple AGRs may need to recharge and multiple charging stations could be free, each combination leading to a different battery degradation, BTC-X makes charging allocation decisions by evaluating the utility function values through Algorithm 1. Note that, for such events, unavailability of free charging stations would simply lead the AGRs to (1) remain idle, or (2), if their energy level is *critical* (i.e., $\leq e^{lowest}$), reach the nearest charging station to wait in a queue for a recharge when E3 occurs. Here, e^{lowest} is the worst case energy consumption required for an AGR to travel from any point of the operational area to its nearest charging station based on the manhattan distance between those two points of interest.

F. Algorithm 1: Allocation

As mentioned in the previous section, the allocation Algorithm 1 is executed for several reasons that require evaluating

Algorithm 1 Allocation

```

1: Upon Arrival of tasks  $\mathcal{N}^T$  (E1), AGRs completing tasks (E2),
   and/or charging stations becoming free (E3) do
2: Query minimum SoC threshold  $\alpha^{min}$  and (%) desired maximum
   relative revenue loss  $\Delta r$ 
3: Calculate  $\alpha$  according to Equation 5
4: Query energy  $e$ , location  $l^r$ , time  $t^c$ , status  $s^r$  from AGRs
5: Query status of charging stations
6: Update  $\mathcal{A}^R$  ▷ set of idle AGRs
7: Update  $\mathcal{A}^C$  ▷ set of free charging stations
8:  $\mathcal{A}^T \leftarrow \mathcal{A}^T \cup \mathcal{N}^T$  ▷ set of active tasks
9: if  $\mathcal{A}^R \neq \emptyset$  then
10:  $U_{|\mathcal{A}^R| \times (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|)} \leftarrow 0$  ▷ Utility matrix
11: Let  $\sigma(i)$  be a function that maps  $i = 1, \dots, |\mathcal{A}^R|$  to an
   AGR ID  $\in \mathcal{A}^R$ 
12: Let  $\rho(j)$  be a function that maps  $j = 1, \dots, |\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|$ 
   to either an ID of a task  $\in \mathcal{A}^T$ , charging station  $\in \mathcal{A}^C$ , or
   idle vertex  $\in \mathcal{I}$ 
13: for  $i = 1, \dots, |\mathcal{A}^R|$  do
14:   for  $j = 1, \dots, |\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|$  do
15:     if  $\rho(j) \in \mathcal{A}^T$  then ▷ task vertex
16:       if  $e_{\sigma(i)}$  is sufficient to complete  $\rho(j)$  then
17:          $U_{i,j} = v_{\rho(j)} - \beta_1 D(\sigma(i), \rho(j))$ 
18:       else
19:          $U_{i,j} = \text{NULL}$ 
20:     else if  $\rho(j) \in \mathcal{A}^C$  then ▷ charge vertex
21:       if  $e_{\sigma(i)} > \alpha$  then ▷  $\alpha$  is energy threshold
22:          $U_{i,j} = \text{NULL}$ 
23:       else
24:          $U_{i,j} = \beta_2(1 - D(\sigma(i), \rho(j)))V(\sigma(i))$ 
25:     else ▷ idle vertex
26:        $U_{i,j} = \lambda$ 
27:  $\mathcal{X} \leftarrow \text{SELECTION}(U)$  ▷  $\mathcal{X}$  is a set of tuples
28: for each  $(\sigma(i), \rho(j)) \in \mathcal{X}$  do
29:   if  $\rho(j) \in \mathcal{A}^T$  then
30:      $s_{\sigma(i)}^r \leftarrow \text{Busy}$ 
31:      $s_{\rho(j)}^t \leftarrow \text{Allocated}$ 
32:      $\mathcal{A}^R \leftarrow \mathcal{A}^R \setminus \{\sigma(i)\}$ 
33:      $\mathcal{A}^T \leftarrow \mathcal{A}^T \setminus \{\rho(j)\}$ 
34:   else if  $\rho(j) \in \mathcal{A}^C$  then
35:     if  $e_{\sigma(i)} \leq e^{lowest}$  or  $t_{\sigma(i)}^c \geq t^{max}$  then
36:        $s_{\sigma(i)}^r \leftarrow \text{Charging}$ 
37:        $s_{\rho(j)}^c \leftarrow \text{Occupied}$ 
38:        $\mathcal{A}^R \leftarrow \mathcal{A}^R \setminus \{\sigma(i)\}$ 
39:        $\mathcal{A}^C \leftarrow \mathcal{A}^C \setminus \{\rho(j)\}$ 

```

the trade off between battery degradation and task revenues when certain conditions occur through the defined events, including the arrival of tasks (E1), AGRs completing tasks (E2), or recharge (E3).

The algorithm first takes as input the minimum energy threshold α^{min} from the user. As a default value, it can be set to $\alpha^{min} = e^{lowest}$ to preserve the revenue as much as possible, since all the available AGR energy would be used for executing tasks until they reach their critical energy level before going for recharge. To improve the sustainability of the AGR fleet, the user is then requested to provide a desired maximum revenue loss Δr . For example, a user may want to bound the maximum reduction in total revenues from task allocation by 2% at most. Afterwards, in Line 3, the value of threshold α regulating when an AGR can remain idle or go to recharge (see definition of charge utility in Equation 3) is

calculated according to the model described in more details in Section III-G. In short, the selected α value ensures that the rest of the proposed algorithm does not exceed the defined bound in relative revenue losses.

The algorithm then updates the system parameters of the available tasks, AGRs, and charging stations, by querying them through network communications (Lines 4-8 of Algorithm 1). If there are idle AGRs (Line 9), i.e., waiting to be allocated to a task or charging station, the algorithm first updates the utility matrix as described in Section III-D (Lines 10-26). We use two hash functions $\sigma(\cdot)$ and $\rho(\cdot)$ to map each row index i and column j of the utility matrix U to the respective unique AGR ID in the set $\mathcal{A}^R$, task ID in the set $\mathcal{A}^T$, charging station ID in the set $\mathcal{A}^C$, and idle vertex ID in the set $\mathcal{I}$ (Lines 11-12). The bipartite graph's edge weights, i.e., utility values of the matrix, are then calculated according to Equations 2, 3, and 4 (Lines 13-26).

At this point, the **SELECTION()** function (described in details in the next paragraph) analyzes the utility values to finalize the allocation choices (Line 27). It returns a set $\mathcal{X}$ of tuples indicating an allocation of an AGR to a selected task, charging station, or idle vertex. If an AGR is allocated to a task (Lines 29-33), then the status of the AGR s^r is updated to *Busy*, the status of the selected task is updated to *Allocated*, and the respective AGR ID and task ID are removed from the sets $\mathcal{A}^R$ and $\mathcal{A}^T$, respectively. If an AGR is allocated to a charging station (Lines 34-39), the algorithm sets the AGR status to *Charging* to the selected charging station, which changes status to *Occupied*, and the sets $\mathcal{A}^R$ and $\mathcal{A}^C$ are updated accordingly. However, it may occur that the allocation algorithm, due to lack of tasks, allocates an AGR to a charging station even though a new task may be coming shortly thereafter. Unless the AGR's energy level is critical, rather than relying on demand predictions we propose to implement a *deferred start of charging* approach that monitors the time elapsed from an AGR first request for a charging $t_{\sigma(i)}^c$. If no task arrives before the end of such configurable period t^{max} (Line 35), then it will be allocated for charging at the requested charging station (following the same path as E4 in Figure 1). Otherwise, the AGR remains idle and can be included for another allocation round at a subsequent task arrival.

SELECTION(): Optimal, Greedy, and Weighted Greedy.

The **SELECTION()** function takes as input the bipartite graph defined by matrix U after calculating all the edge weights (described in Algorithm 1). It returns a set $\mathcal{X}$ of tuples consisting of AGR and the selected task, charging station or idle vertex that maximizes the total sum of the selected utilities. To design the **SELECTION()** function, we have used three procedures that define the three algorithms members of the BTC-X family, BTC-M, BTC-G, and BTC-WG. The first one is the weighted matching algorithm based on the Kuhn-Munkres approach [37] (called **WEIGHTED_MATCHING()** here), which finds the optimal allocation for the current event but incurs the largest overhead in the family, particularly in large-scale scenarios. For the second one, corresponding to BTC-G, we propose an algorithm called **GREEDY_SELECTION()**, which utilizes a greedy approach to provide a sub-optimal allocation with lowest overhead in the BTC-X family. However, this approach

Algorithm 2 GREEDY_SELECTION()

```

1: Input: Utility matrix  $U$  of size  $|\mathcal{A}^R| \times (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|)$ 
2: Tuple set  $\mathcal{X} \leftarrow \emptyset$ 
3:  $A_{|\mathcal{A}^R|} \leftarrow 1$ 
4:  $B_{|\mathcal{A}^T|+|\mathcal{A}^C|+|\mathcal{I}|} \leftarrow 1$ 
5: while  $\sum_{i=1}^{|\mathcal{A}^R|} A_i \geq 1$  do
6:    $(i^*, j^*) = \underset{\substack{i=1, \dots, |\mathcal{A}^R|; \\ j=1, \dots, |\mathcal{A}^T|+|\mathcal{A}^C|+|\mathcal{I}|}}{\operatorname{argmax}} \{U_{i,j} \mid A_i = 1 \ \& \ B_j = 1\}$ 
7:    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(i^*, j^*)\}$ 
8:    $A_{i^*} \leftarrow 0$ 
9:    $B_{j^*} \leftarrow 0$ 
10: Output:  $\mathcal{X}$ 

```

may not handle well scenarios where some AGRs cannot execute all the available tasks. For instance, consider three AGRs (AGR 1, AGR 2, AGR 3) are available with three allocatable tasks (T1, T2, T3). Let's consider also that AGR 1 and AGR 2 can execute any tasks but AGR 3 only have enough energy to execute task T2. According to the utility values, it could happen that BTC-G allocates T2 to either AGR 1 or AGR 2, leading T3 to remain unallocated and AGR 3 idle. To improve performance we thus design BTC-WG, our third algorithm, which is a weighted variant of BTC-G called WEIGHTED_GREEDY(). This variant shows better performance than BTC-G but slightly larger overhead, as we will show in Section IV, thus representing a middle-way solution between BTC-M and BTC-G.

WEIGHTED_MATCHING(). The weighted matching algorithm solves the problem of maximum weight matching in the bipartite graph defined by the utility matrix U in Algorithm 1, where the rows and the columns refer to the two distinct sets (bi-partitions) of vertices in the bipartite graph. This algorithm leverages the Kuhn-Munkres algorithm (also known as the Hungarian method) to find the maximum matching in a weighted bipartite graph so that the total weight (utilities) among the matched edges is maximized. At first, the algorithm subtracts the smallest value in each row from all other values in the same row and repeats the same for each column to reduce the matrix. After that, it tries to use the fewest possible lines (horizontal or vertical) to cover every zero element in the matrix. If all the zeros are not covered, it tries to adjust the uncovered elements by finding the smallest uncovered element, subtracting it from all uncovered ones and then adding it to the elements that are covered two times (the elements where two lines intersect). These steps are repeated until each row element is matched with an unique column element ensuring the sum of the selected edge weights is maximized.

Time Complexity of BTC-M. The core of our BTC-X algorithm is the Algorithm 1 Allocation(). The major steps for executing Algorithm 1 first involve the computation of the utility matrix elements by executing Lines 11-24 which takes $O(|\mathcal{A}^R| \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$. After that the SELECTION() function in Line 25 is executed, which uses the WEIGHTED_MATCHING() algorithm with a running time $O((\max\{|\mathcal{A}^R|, |\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|\})^3)$. Finally, it takes $O(|\mathcal{A}^R|)$ to execute the allocation decisions

Algorithm 3 WEIGHTED_GREEDY()

```

1: Input: Utility matrix  $U$  of size  $|\mathcal{A}^R| \times (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|)$ 
2: if  $|\mathcal{A}^T| \neq \emptyset$  then
3:    $W_{|\mathcal{A}^R|} \leftarrow 1$ 
4:   for  $i = 1, \dots, |\mathcal{A}^R|$  do
5:      $T_{alloc} \leftarrow 0$ 
6:     for  $j = 1, \dots, |\mathcal{A}^T|$  do
7:       if  $U_{i,j} \neq \text{NULL}$  then
8:          $T_{alloc} \leftarrow T_{alloc} + 1$ 
9:       if  $T_{alloc} > 0$  then
10:         $W_i \leftarrow |\mathcal{A}^T|/T_{alloc}$ 
11:   for  $i = 1, \dots, |\mathcal{A}^R|$  do
12:     for  $j = 1, \dots, |\mathcal{A}^T|$  do
13:        $U_{i,j} \leftarrow U_{i,j} \times W_i$ 
14: GREEDY_SELECTION( $U$ )

```

in Lines 26-37. Hence, the total running time of BTC-M is $O((\max\{|\mathcal{A}^R|, |\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|\})^3)$.

GREEDY_SELECTION(). The pseudo-code of the greedy selection approach is provided in Algorithm 2. It considers a greedy method to find a sub-optimal allocation decision that maximizes the total utility of the edge weights of the input bipartite graph. To do so, the algorithm first initializes the tuple set $\mathcal{X}$ to the empty set in Line 2. Then, it initializes the elements of two arrays A and B to 1 in Lines 3-4. Array A is used to keep track of the row indices (AGRs) that have been matched. Array B is used to keep track of the column indices (tasks, charging stations or idle vertex) that have already been matched. The algorithm then proceeds by continuously finding the maximum element of the matrix (highest edge weight). Once identified, the corresponding row and column indices (i^*, j^*) are added to the tuple set $\mathcal{X}$ (Line 7). In addition, it sets the indices i^* and j^* in the array A and B , respectively, to zero (Lines 8-9), which eliminates them from the following rounds of selection. This process is repeated until every element i (i.e., every AGR) in array A has been matched.

Time Complexity of BTC-G. BTC-G executes Algorithm 1 where the SELECTION() function in Line 25 uses GREEDY_SELECTION() (given in Algorithm 2) which has a running time $O(|\mathcal{A}^R|^2 \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$. Earlier we, have shown that the running time to execute Algorithm 1 without the SELECTION() function is $O(|\mathcal{A}^R| \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$. Thus the execution time of BTC-G is $O(|\mathcal{A}^R|^2 \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$, i.e., lower than that of BTC-M.

WEIGHTED_GREEDY(). The pseudo-code of the weighted greedy approach is provided in Algorithm 3. This algorithm operates on the utility matrix U constructed in the previous stage and aims to optimize the allocation of tasks to available AGRs by incorporating a weighting mechanism. The algorithm begins by checking whether there are tasks available for allocation. If tasks are available, it initializes a per-AGR weight array W with all elements set to 1 (Line 3). The weight W_i for each AGR i is determined as the total number of available tasks divided by the number of tasks that can be executed by that AGR. To compute the weight values, the algorithm iterates over each row of the matrix, analyzing the corresponding task nodes (columns). It initializes a counter variable, T_{alloc} , to zero, which keeps track of the distinct

tasks that can be executed by each AGR. For each task node (column index) associated with a given AGR (row index), the algorithm verifies whether the corresponding matrix entry is non-NULL, indicating that the AGR is capable of executing the task. If so, T_{alloc} is incremented by 1. When all task nodes for a given AGR i have been evaluated, and if the AGR is able to execute at least one task, its corresponding weight is computed (Lines 4-10). The weights are then used to update the corresponding AGR-Task utility values, giving more importance to AGRs able to execute a lower number of the available tasks. Finally, the GREEDY_SELECTION() algorithm is executed using the updated utility matrix U as input.

Time Complexity of BTC-WG. BTC-WG executes Algorithm 1 where the SELECTION() function in Line 25 uses WEIGHTED_GREEDY() (given in Algorithm 3) and the GREEDY_SELECTION() (given in Algorithm 2). As a result, the running time of BTC-WG is $O(|\mathcal{A}^R| \cdot |\mathcal{A}^T|) + O(|\mathcal{A}^R|^2 \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$, which becomes $O(|\mathcal{A}^R|^2 \cdot (|\mathcal{A}^T| + |\mathcal{A}^C| + |\mathcal{I}|))$, lower than that of BTC-M.

G. Bounding the Relative Revenue Losses

In our conference paper [19], BTC took as an input the value of threshold α , which is a parameter used to avoid unnecessary recharges of an AGR when its battery has more than α Joules of energy stored. However, we believe AGR fleet managers would likely find that providing a maximum bound of acceptable revenue losses to gain in sustainability as input to our algorithm is easier than guessing a good α value. As a result, in this extended paper we have explored how this parameter affects the generated revenue. As we will show in Figure 9a in Section IV-E, we have found that the above BTC algorithms lead to a linear relationship between this parameter and the amount of revenue losses relative to a minimum threshold value of reference, e.g., $\alpha^{min} = e^{lowest}$ for maximum revenues in general while allowing a minimum remaining energy to go back to any charging station. This is because keeping a lower threshold for recharge allows the AGRs to use more energy to execute tasks, which in turn generates higher revenue at the cost of a lower lifespan and vice versa. Because companies interested in embedding sustainability in their fleet management are likely willing to only give up a bounded percentile of their revenues, our revised strategy is to automatically tune up the parameter α such that the proposed framework still maximizes the total utility without exceeding a desired relative revenue loss. Thus, BTC-X takes as input the relative acceptable loss of generated revenue (%) to find the optimized SoC threshold α using the following linear regression model:

$$\alpha = \alpha^{min} + \Delta r / M \quad (5)$$

where, Δr is the percentage of revenue loss accepted by the user and M is the slope of the sampled data. In a practical deployment, an effective approach to estimating the slope M involves operating the fleet with a few values of α , starting from α^{min} , within the same day across a few consecutive weeks. During this period, the differences in generated revenues can be recorded to estimate M . As

TABLE I
SIMULATION PARAMETER VALUES

Parameter	Value	Parameter	Value
CPU Frequency	2.26GHz	Charge Threshold (α/E^{max})	50%
AGR Speed ($\bar{s}$)	1.6m/s	Task Relative Deadline	5min
AGR Mass (m)	15kg	Util. Weights (β_1, β_2)	[0.1, 10]
Ground Angle (ϕ)	[-3, 3]°	Max Charge Delay (t^{max})	3min
Power (p)	38W	Max SoC Recharge	80%

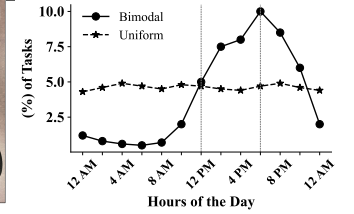
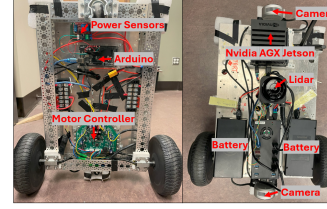


Fig. 2. Our prototype used to profile Fig. 3. Task arrival distribution per hour: Bimodal vs. Uniform.

additional data becomes available over time, the model can also be periodically refined to further increase accuracy. In Section IV-E, we will show that this model achieves high accuracy for controlling the relative revenue losses.

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

For our experiments, we use simulations to feasibly run multi-year tests for battery degradation evaluations. However, we have used our AGR prototype in Figure 2 to profile the power consumption for different speed and CPU frequency levels of the computing board ensuring realistic simulations. It features a ROS-based navigation stack running on an NVIDIA Jetson AGX Xavier, two 250W hoverboard motors, a 36V 20Ah Li-ion battery, a Slamtec RPLidar S1 lidar, an Arduino Mega 2560, and two INA3221 and INA260 power sensors. We have used the measured power consumption at highest frequency and 1.6 m/s speed to estimate the AGRs' energy consumption in our simulations. Table I shows the rest of the parameters used for our experiments.

Experiment Scenario. To ensure the experiments reflect real-world conditions, we consider practical scenarios where task arrivals throughout the day follow two distinct distributions: bimodal and uniform. A bimodal distribution is characterized by two distinct peaks, representing two frequent periods of high activity [38]. In contrast, a uniform distribution assumes tasks are evenly distributed across multiple time intervals. In the food delivery application, task arrivals (i.e., food orders) naturally follow a bimodal distribution, with demand peaking during lunch and dinner hours [39]. Meanwhile, in warehouses where the AGRs are primarily employed for tasks such as loading, unloading, and transporting goods, the task arrivals are generally uniformly distributed, as also assumed in related work [21] throughout the day, ensuring a steady and consistent workload as shown in Figure 3. For our experiments, we have considered an area of 1km² divided in squares of 1m². In addition, we have introduced variability in the hourly distribution while maintaining the overall trend in order to simulate the task arrivals over multiple years.

For each task, we randomly generate start and end locations from the cells within the area and measure the Manhattan

distance, select an average slope between -3 to 3 degrees, and assign a task valuation from the range [\$10, \$100] denoting the potential revenue for completing the task. We also randomly generate 2-4 charging locations, each with multiple charging stations based on a given station-to-AGR ratio (0.5 unless stated otherwise). Initially, each AGR's current location is generated randomly and the SoC is selected to be between 60% and 90%. We also consider a heterogeneous level of battery capacity degradation for each AGR in the beginning of each simulation, ranging from 0 to 10%, depending on their prior operational history. This degradation level is computed using the historical SoC data of each AGR, which we assume is accessible to the fleet manager. The simulator, developed in Python, was tested on a system with an Intel Core i7-8750H CPU at 2.20GHz and 24GB DRAM. *The source code is available on GitHub [40].*

Problem Instances. To evaluate our algorithm's performance we have conducted a series of experiments across various problem instance sizes. For small instances, we consider 2-5 AGRs, 20-100 tasks per day, and 2 charging locations. For medium sized instances, we increase the number of AGRs from 5 to 20 and the number of tasks per day between 100 to 400, accompanied by 3 charging locations. For large instances, we further vary the number of AGRs between 20 to 45, number of tasks per day from 400 to 900, and consider 4 charging locations. Most of the experiments are simulated over 2 years to properly evaluate the battery degradation trends.

Baselines. Because previous studies have not focused on online algorithms that take into account the long-term impact of task allocation and recharging decisions on battery lifespan, we have implemented two representative baselines to evaluate the performance of our proposed approach. The first baseline, REV, is similar to the state-of-the-art approaches that solely focus on maximizing the number of allocated tasks [15], [16], which can be translated into purely maximizing the total revenue generated without considering battery lifespan. To further help maximize the obtained revenues, REV considers as task value the ratio of task revenue over the travel distance. The second baseline, DEG, prioritizes minimizing battery degradation while allocating tasks to the AGRs without focusing on high revenue generation. We implement both baselines using our proposed framework. However, they calculate their task utilities either with only the value maximization target (for REV) or only with the degradation minimization target (DEG). For charging decisions, REV charges only when an AGR energy level reaches the minimum e^{lowest} to maximize task allocation. For DEG, we tested charging thresholds from 20% to 70%, recharging AGRs when their SoC dropped to these levels. Figure 4 shows the average lifespan increase under different thresholds. In our experiments, we chose the threshold that produced battery degradation similar to our proposed methods (e.g., 35% for uniform distribution). This choice demonstrates that revenues can be increased

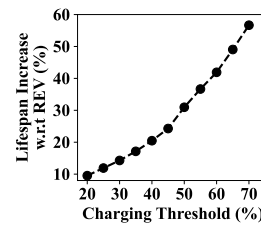


Fig. 4. Lifespan impact of different recharge thresholds for the DEG baseline.

for comparable degradation by jointly optimizing revenue and lifespan. For the baselines REV and DEG, after the utilities are calculated, decisions are made using the weighted matching approach (i.e., optimal) described in Section III-F with input the baseline-specific utility matrix. We compare the performance of these baselines with the three variants of our proposed approach, BTC-M, BTC-G, and BTC-WG.

Performance Metrics. To evaluate and compare the performance of our algorithm with the baselines, we have considered three performance metrics. The **Generated Revenue** is calculated as the percentage revenue generated from allocating tasks over the total revenue of all arrived tasks. The **Battery Degradation** is calculated as the average percentage of battery capacity lost across AGRs over the 2-years simulated time. However, a small reduction in battery degradation can significantly extend the lifespan, defined as the number of days taken to reach a *reference battery degradation* level. We use the degradation level reached by DEG, which typically results in the lowest degradation, as our reference. Since REV leads to highest revenue and degradation, we thus quantify the **Increase in Lifespan** as the percentage increase in number of days with respect to (w.r.t.) REV for the sustainability-aware schemes (BTC-X and DEG) to reach the defined *reference battery degradation*.

B. Example Case Study

In this section, we provide a detailed description of the operation of the baselines, REV, DEG and our proposed algorithms BTC-M and BTC-WG on an example case study with uniform task arrival distribution. For space reasons, we do not show BTC-G in this experiment, but we will compare its performance for wider experimental analysis in the next section. Also, for our BTC-X algorithms here we use a charge threshold value of α that allows AGRs charging only when their state of charge relative to the maximum energy capacity E^{max} is below 50%. This is because we have experimentally found that, for uniform task arrival distribution case, this value of α leads to a good balance between generated revenue and lifespan increase of the AGRs. We will show the general results for different α values and distributions in Sections IV-E and IV-F. In this illustrative case study, we simulate an environment which consists of 4 AGRs, 150 daily tasks, and 2 charging locations, each equipped with two charging stations. The AGR fleet is composed of a mix of AGRs with varying battery conditions e.g., AGRs 1 and 2 are equipped with nearly new batteries, while AGRs 0 and 3 have relatively aged batteries, resulting in differing initial degradation levels. The experiment is conducted over a two-year period to observe and analyze the progression of battery capacity degradation among the AGRs.

To illustrate the differences more clearly, Figure 5 presents the SoC profiles of the AGRs over a single day for each of the compared approaches. In the case of REV (Figure 5a), the focus is on maximizing task allocation, which causes AGRs to continue executing tasks until their battery levels are critically low, at which point they are forced to recharge. While this method yields higher revenue by completing more tasks, it significantly accelerates battery degradation, especially for AGRs equipped with older batteries (e.g., AGR 0) that experience frequent recharging at a deeper depth of discharge. On the

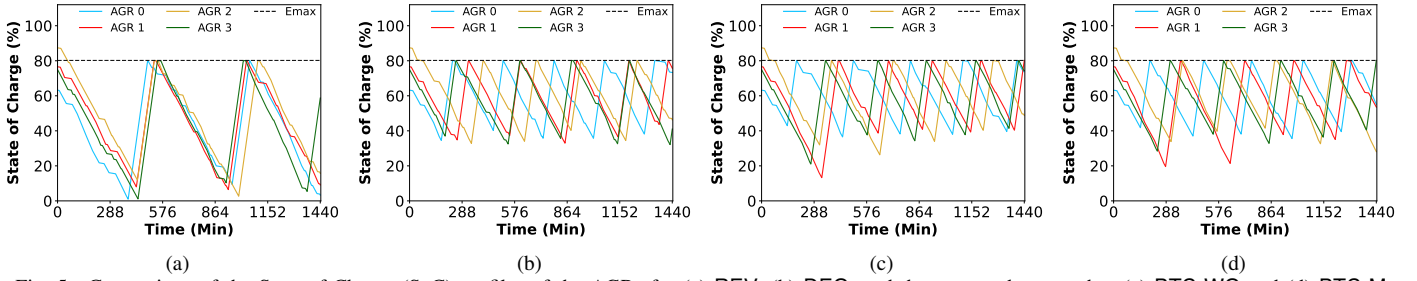


Fig. 5. Comparison of the State-of-Charge (SoC) profiles of the AGRs for (a) REV, (b) DEG, and the proposed approaches (c) BTC-WG and (d) BTC-M for a representative experiment of 4 AGRs and 150 tasks per day.

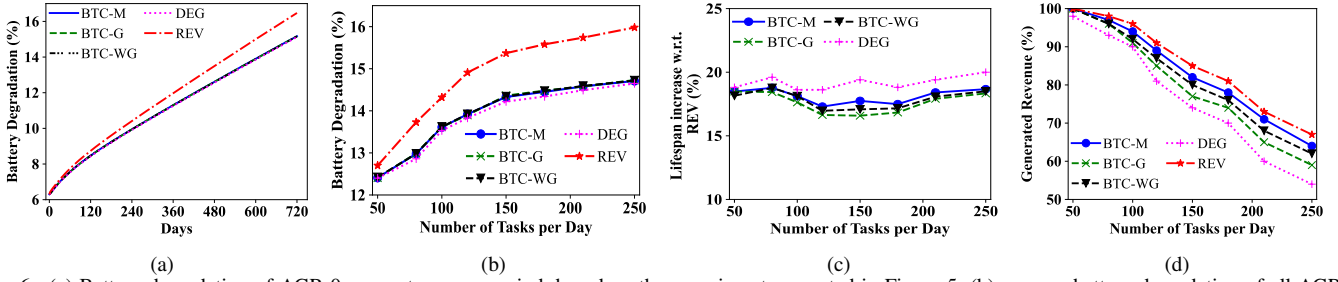


Fig. 6. (a) Battery degradation of AGR 0 over a two-year period, based on the experiment presented in Figure 5; (b) average battery degradation of all AGRs after two years; (c) battery lifespan improvement relative to REV for DEG, BTC-G, BTC-WG, and BTC-M; and (d) revenue generated under increasing task demand.

other hand, DEG (Figure 5b) adopts a battery health-oriented strategy, where task allocation decisions are made to minimize battery degradation, even if it compromises revenue. In this approach, AGRs are recharged whenever their SoC falls below 35%, effectively reducing battery wear. However, this comes at the expense of potentially considerable revenue loss despite achieving a longer battery lifespan. In contrast, the proposed BTC-WG (Figure 5c) and BTC-M (Figure 5d) methods jointly consider both revenue maximization and battery degradation when making task and charge allocation decisions. By balancing these two objectives, the approaches effectively mitigate deep discharge cycles, especially for AGRs with older batteries (e.g., AGR 0 and 3 in Figure 5d), by initiating recharges at relatively higher SoC levels. This strategy enables the system to achieve a favorable trade-off between maintaining lower battery degradation and sustaining higher revenue generation. While BTC-M is designed to determine an optimal allocation at each decision event, BTC-WG provides a sub-optimal yet computationally efficient solution, resulting in minor variations in the allocation patterns, as depicted in the figures.

A comparative analysis of the four approaches reveals notable differences in both revenue generation and battery degradation trends. The REV approach achieves the highest revenue, reaching 84%, while DEG records the lowest at 74%. In comparison, BTC-WG and BTC-M obtain intermediate values of 80% and 82%, respectively, as reported in the subsequent section (150-task scenario, Figure 6d). Furthermore, the progression of battery capacity degradation for AGR0 over a two-year period is presented in Figure 6a for all four methods. The results indicate that, under DEG, AGR0 reaches a battery degradation of 15.12% after 720 days, whereas with REV the degradation is higher, reaching 16.47% over the same duration. The degradation levels obtained with BTC-G, BTC-WG, and BTC-M remain close to what achieved by DEG over the two-year horizon. In other words, we observe that REV

leads AGR 0 to reach the reference battery degradation after only 608 days, while under BTC-G, BTC-WG, and BTC-M, the battery achieves the same degradation level only after 720 days. BTC-M being the optimal leads to higher revenue compared to both BTC-G and BTC-WG. This translates into about 18% longer lifespan compared to REV, illustrating how even modest improvements of 1–2% in battery degradation can result in significantly prolonged battery life.

C. Effect of Increased Demand

To further investigate the effectiveness of our approach under increased task demand, we have performed additional experiments by keeping the number of AGRs fixed at 4 while varying the number of daily tasks from 50 up to 250. The outcomes of these experiments are presented in Figure 6b, which shows the average battery capacity degradation of the AGRs under different task loads. As the task volume increases, we observe a slight rise in capacity degradation, primarily because the AGRs are not always able to recharge at their ideal SoC levels due to the elevated workload. Despite this, our proposed methods, BTC-M, BTC-G, and BTC-WG, consistently outperform REV by reducing battery capacity degradation by more than 7% across all task loads. As a result, the proposed BTC-M, BTC-G, and BTC-WG approaches achieve an increase in battery lifespan of approximately 17%, 16%, and 17%, respectively, even under elevated task demands, as illustrated in Figure 6c. Finally, we can see that BTC-M, BTC-WG, and BTC-G achieve up to 19%, 19%, and 18% longer lifespan compared to REV.

The degradation observed with our approaches is only 0.7% higher than that of DEG. However, it leads to less than 2% lower lifespan when compared to DEG as seen in Figure 6c. While DEG achieves a marginally higher battery lifespan, its revenue generation remains limited to only 54% when handling 250 tasks per day, with REV achieving 22% higher

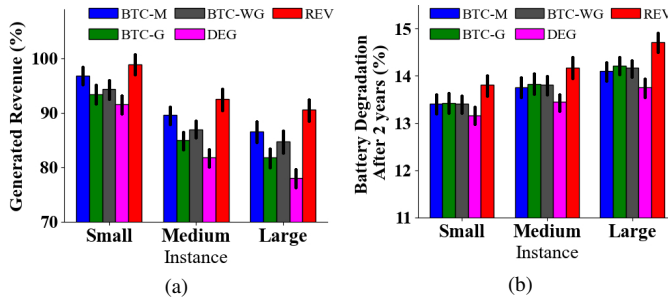


Fig. 7. Comparison of (a) the generated revenue and (b) the average battery capacity degradation of AGRs after two years across small, medium, and large problem instances.

revenues. In comparison, BTC-M, BTC-G, and BTC-WG outperform DEG by achieving approximately 19%, 10%, and 16% more revenues than DEG, respectively, as illustrated in Figure 6d. We can see in terms of revenue generation BTC-WG performs significantly better than BTC-G. This is due to employing a weighted greedy approach that optimizes the task allocation choices considering the heterogeneous ability of each AGR in executing tasks.

Percentage of Allocated Tasks: Table II presents the percentage of allocated tasks for the baselines and the three proposed approaches. As expected, all methods exhibit a sharp decline in task allocation as task load increases, with REV and DEG leading to the best and worst results, respectively. Among our solutions, BTC-M shows the best results, with only 3% lower task allocation than REV at the highest task demand. BTC-WG shows up to 3% higher task allocation than BTC-G and 5% lower than REV. Finally, DEG achieves the lowest task allocation among the others, which is close to that of BTC-G. However, in terms of revenue, we can see that BTC-G generates 10% more revenue than DEG even with the similar number of allocated tasks. This is because DEG only focuses on minimizing the battery degradation while BTC-G focuses on both revenue and degradation.

The results from these experiments confirm that our proposed approaches are capable of significantly extending battery lifespan while incurring only minimal revenue loss, even when faced with increasing task demands.

TABLE II
ALLOCATED TASK (%) WITH VARYING TASK LOADS

Approach	Number of tasks per day				
	50	100	150	200	250
REV	100%	97%	82%	63%	52%
BTC-M	100%	98%	80%	61%	49%
BTC-WG	100%	98%	79%	58%	47%
BTC-G	100%	96%	76%	56%	45%
DEG	99%	96%	74%	55%	42%

D. Performance Comparison with Different Problem Instances

To thoroughly assess the effectiveness of our proposed methods, we have carried out additional experiments covering small, medium, and large problem instances, as described in the experimental setup. The results of these experiments are presented in Figure 7a and Figure 7b, where the red bars represent the standard deviation. As illustrated in Figure 7a, our approaches, namely BTC-M, BTC-G, and BTC-WG,

consistently outperform DEG in terms of revenue generation, achieving average improvements of 5%, 9%, and 11% for small, medium, and large instances, respectively. On the other hand, the REV method produces slightly higher revenue than our approaches, with average gains of 2%, 3%, and 4% for the respective problem sizes, respectively. However, this marginal revenue gain comes at the cost of noticeably higher battery capacity degradation. As illustrated in Figure 7b, the average battery degradation of the AGRs under the REV approach is approximately 4% higher compared to the degradation observed with our proposed methods. It is worth highlighting that even this modest reduction in capacity degradation achieved by our approaches results in a significantly extended battery lifespan, as evidenced by the results presented in Figure 6b and Figure 6c.

Performance and Scalability with different problem instances.

As our proposed approaches and the baselines are implemented for online solutions, we assess the execution time required for each allocation decision, as shown in Figure 8. It is evident that the execution time increases for all methods as the problem size grows. REV consistently achieves the shortest execution times, ranging between 0.006 seconds and 0.05 seconds. This efficiency is mainly attributed to its straightforward decision-making process, which avoids the need to process augmented SoC traces for estimating battery degradation and instead applies simple fixed rules for recharge decisions. On the other hand, DEG incurs slightly longer execution times, from 0.009 to 0.08 seconds, as it incorporates battery degradation estimation by analyzing SoC traces during task allocation. Our proposed methods, namely BTC-M, BTC-G, and BTC-WG, require additional computational effort due to the dual objectives of considering both revenue and battery degradation, along with the integration of task allocation and recharge scheduling. Despite this added complexity, the execution time for each decision remains practical for online applications, with BTC-M taking 0.22 seconds, BTC-G requiring 0.15 seconds, and BTC-WG completing decisions within 0.17 seconds, even when handling large-scale scenarios involving 45 AGRs and 900 tasks per day. BTC-WG has slightly longer execution time compared to BTC-G due to its optimization of utility values prior to allocating tasks. These findings demonstrate the practicality and effectiveness in terms of scalability of our sustainability-aware allocation algorithms when applied to real-world scenarios with reasonable AGR fleet sizes. However, for extremely large fleets, e.g., with more than 1000 AGRs, since the time complexity of our BTC-M is cubic while that of BTC-G and BTC-WG are quadratic, future research may have to investigate how to further reduce the time complexity to achieve better scalability.

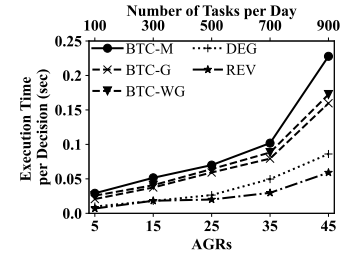


Fig. 8. Execution time comparison.

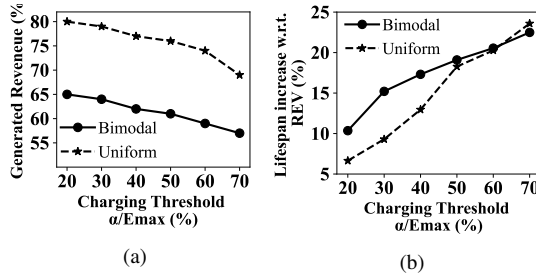


Fig. 9. Comparison of (a) generated revenue and (b) battery lifespan improvement after two years for varying charging threshold values under bimodal and uniform task arrival distributions.

E. Revenue Losses Model Validation

In Section III-G, we introduced a linear model to determine the optimized α that keeps revenue loss within a specified limit. Here, we validate this model by evaluating the impact of varying BTC's charging/idle threshold α in a scenario with 4 AGRs and 150 tasks per day. The threshold is varied from 20% to 70% of the maximum battery capacity E^{max} to examine its effect on performance metrics. To study how different α values influence system behavior under various task arrival patterns, we design two scenarios: one with uniform and the other with bimodal arrivals. Each experiment uses our proposed method, BTC-M. Results in Figure 9 show the revenue and AGR battery lifespan improvement achieved by BTC-M compared to the baseline REV. To validate the model, we compute the slope M from Figure 9a, obtaining 0.16 for the bimodal case and 0.2 for the uniform case. For example, using $\alpha^{\min} = 0.2E^{max}$ and a desired maximum revenue loss Δr of 2% relative to $\alpha^{\min}$, Equation 5 estimates charging thresholds of 32.5% for the bimodal distribution and 30% for the uniform distribution. We then calculate the Mean Absolute Percentage Error (MAPE) [41] by comparing actual and predicted revenue losses. The MAPE values are 1.5% for bimodal and 1.3% for uniform distributions, both very low. These results demonstrate that the model effectively determines optimized α values for any given revenue loss target, ensuring predicted losses closely align with the desired limits defined by the fleet manager.

F. Effect of Idle/Charging Threshold

From Figure 9a, we can see that for different values of α the difference in revenue generation follows a similar slope between bimodal and uniform task arrival distributions. However, this is not true in terms of the battery lifespan of the AGRs. As shown in Figure 9b, using a threshold of 50% to 70% of the battery capacity the lifespan increase of the AGRs is similar between the two distributions. However, for lower values of α we observe a much larger difference. For instance, for α set to 30% of E^{max} , the average AGR lifespan with the bimodal distribution is 5% higher than that of the uniform distribution. To better understand this behavior we conducted a more detailed analysis. Specifically, we performed a total of 100 experimental runs for each task arrival pattern—bimodal and uniform—under two different charge/idle threshold values, 30% and 50% of the capacity E^{max} . The results of these

experiments are shown in Figure 10. We perform our analysis through a series of observations on the number of daily charges and Depth of Discharge (DoD) of the AGRs.

Number of Average Daily AGR recharge cycles. In the case of bimodal distribution, the majority of tasks are concentrated during peak hours (e.g., lunch and dinner times), with significantly fewer tasks during the rest of the day. As a result, AGRs remain mostly idle during these low-demand periods and typically require recharging only after periods of intense task execution during peak hours. In contrast, with a uniform distribution, tasks are spread more evenly throughout the day. This leads to consistent activity for the AGRs, causing them to deplete energy consistently and, consequently, recharge more frequently throughout the day. Moreover, when the charging threshold is reduced to 30%, AGRs tend to delay recharging until their state of charge drops below this lower threshold. This behavior leads to a reduction in the average number of recharge cycles in both distribution cases when compared to the scenario with 50% charging threshold, as illustrated in Figures 10d and Figure 10a. Additionally, for both threshold values (30% and 50%), the average number of recharge cycles for AGRs under the uniform distribution remains consistently higher than that under the bimodal distribution. For instance, AGRs recharge approximately 1.8 times more frequently under the uniform distribution than the bimodal one when the charging threshold is 50% and 1.5 times more when the charging threshold is lowered to 30% of the battery capacity. Thus, the uniform distribution leads AGRs to charge more frequently than the bimodal distribution for all charging threshold values.

Highest DoD of the AGRs. Figures 10b and 10e show the percentile of AGRs charging at each hour of the day for the two cases of charging thresholds set to 50% and 30% of the battery capacity, respectively. For the 50% case, the recharge events under the uniform distribution are observed to be evenly distributed throughout the day, as shown in Figure 10b. In contrast, when the threshold is reduced to 30%, the recharge events occur more in batches, as illustrated in Figure 10e. In the case of bimodal distribution, for both 50% and 30% of the charging thresholds, the AGRs are predominantly recharged after the peak demand periods, when they are idle and have already consumed a substantial portion of their energy during task-intensive hours. This behavior provides further insight into the trends observed in Figure 10c and 10f, which present the average highest Depth of Discharge (DoD) for the AGRs under each experimental setting. Specifically, for 50% charging threshold only 13% of the experimental runs show a highest DoD for the uniform distribution to be higher than that of the bimodal case (see highlighted red dots in figures). However, this trend becomes more prominent when the charging threshold is reduced to 30% of the capacity, with approximately 21% of the experiments showing AGRs reaching higher DoD in the uniform case.

These results help us explain the difference in battery lifespan for the two distributions at different charging threshold values. In summary, having a higher number of charges is not necessarily detrimental in terms of battery degradation, specially at high charging thresholds higher than 50%, as seen in Figure 9b since each charge is often done at a lower

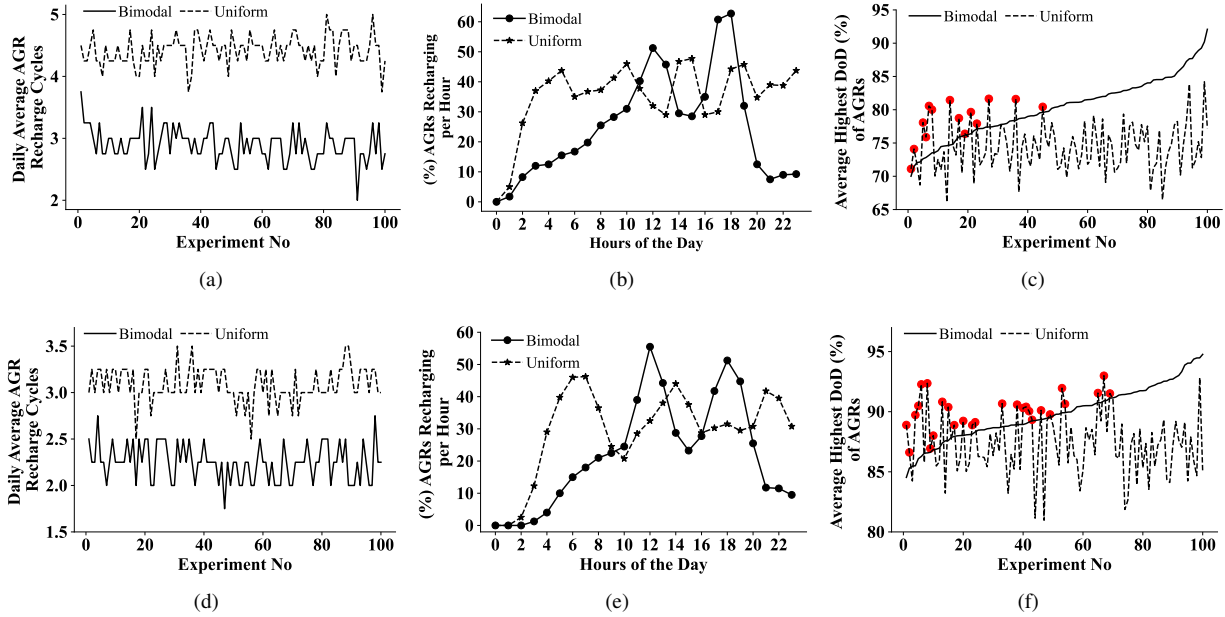


Fig. 10. Comparison of daily average AGR recharge cycles, percentage of AGR recharging per hour, and Average highest DoD (%) of the AGRs between bimodal and uniform task arrival distributions, for charging threshold $\alpha/E^{max} = 50\%$ (a-c) and 30% (d-f) .

DoD with a uniform task arrival. However, decreasing the charging threshold further with a uniform task distribution leads such frequent recharges to be more concentrated in time and at higher DoD levels, which leads to the observed lifespan difference with a bimodal distribution.

V. LIMITATIONS & FUTURE RESEARCH DIRECTIONS

While this work presents an improved version of the online Battery Degradation-Aware Task and Charge Allocation framework, there are several important directions for future research.

First, our framework does not forecast future task arrivals to avoid the complexity of predictive modeling. Incorporating short-term forecasting could enable more proactive decisions, such as coordinating AGR task assignments and charging during peak demand, but would require fast and lightweight predictive models to avoid slowing online decisions.

Second, while the proposed greedy and weighted greedy algorithms (BTC-G and BTC-WG) perform well, a key direction is formally analyzing their approximation bounds to provide theoretical guarantees and deeper insights. Another is incorporating heterogeneous AGR capabilities and battery chemistries, since real fleets may include vehicles with varying payloads, speeds, and other features. Accounting for these differences could improve both effectiveness and equity in task and charge allocation.

Finally, future work should examine adapting this framework to larger, distributed systems. For example, edge or cloud-based deployments with multiple agents may introduce communication delays, making it important to analyze performance under such conditions to support scalability and practical implementation.

VI. CONCLUSION

In this paper, we presented an improved framework for online, sustainability-aware task and charge allocation in autonomous ground robot (AGR) fleets. Unlike our prior work [19], the new framework incorporates a model that helps maximize the fleet sustainability and revenues while ensuring that the relative revenue losses remain within the user-defined acceptable limit. In addition, we proposed a weighted greedy algorithm, BTC-WG, which improves task allocation efficiency over the previously introduced greedy approach, BTC-G, by accounting for the variation in the number of executable tasks available to each AGR. We also conducted a comprehensive evaluation of the BTC-X framework under multiple task arrival distributions and varying relative revenue loss bounds to analyze their effects on the overall battery lifespan of the fleet. Simulation results based on a real AGR model, compared against multiple baselines, demonstrated that our framework can extend battery lifespan by up to 17% and 19% in case of uniform and bimodal task arrival distributions, respectively, with minimal revenue loss of approximately 2-4%.

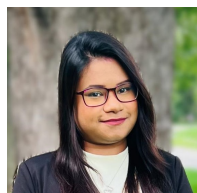
REFERENCES

- [1] Amazon, "10 years of amazon robotics: How robots help sort packages, move product, and improve safety," <https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety>, 2022.
- [2] Starship, "Home," <https://www.starship.xyz/company/>, 2022.
- [3] Cartken, "Cartken Home Page," <https://www.cartken.com>, 2023.
- [4] Grubhub, "Grubhub and Starship Technologies Partner to Bring Robot Delivery Services to College Campuses," <https://about.grubhub.com/news/grubhub-and-starship-technologies-partner-to-bring-robot-delivery-services-to-college-campuses/>, 2023.
- [5] Best Colleges, "Robots Invade College Campuses," <https://www.bestcolleges.com/news/robots-invade-college-campuses/>, 2023.
- [6] Today at Wayne, "Starship, on-campus robotic food delivery service, set to launch this fall," <https://today.wayne.edu/news/2022/07/08/starship-on-campus-robotic-food-delivery-service-set-to-launch-this-fall-48889>, 2022.

- [7] Joe Guskowski - FSD, "Grubhub rolling out new robots at Ohio State, with more campuses to come," <https://www.foodservicedirector.com/technology-equipment/grubhub-rolling-out-new-robots-ohio-state-more-campuses-come>, 2022.
- [8] Nadim Maluf, "What happens after 80%?" <https://www.qnov.com/blogs/what-happens-after-80-percent>, 2014.
- [9] S. Jeon, J. Lee, and J. Kim, "Multi-robot task allocation for real-time hospital logistics," in *2017 IEEE Int. conf. on syst., man, and cybernetics (SMC)*, 2017.
- [10] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *Proc. Int. Joint Conf. on Autonomous Agents and Multiagent Syst. (AAMAS)*, 2019.
- [11] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2018.
- [12] D.-H. Lee, "Resource-based task allocation for multi-robot systems," *Robotics and Autonomous Syst.*, vol. 103, pp. 151–161, 2018.
- [13] Z. Zhu, B. Tang, and J. Yuan, "Multirobot task allocation based on an improved particle swarm optimization approach," *Int. Journal of Advanced robotic syst.*, vol. 14, no. 3, p. 1729881417710312, 2017.
- [14] G. Ferri, J. Bates, P. Stinco, A. Tesei, and K. LePage, "Autonomous underwater surveillance networks: A task allocation framework to manage cooperation," in *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*. IEEE, 2018, pp. 1–10.
- [15] M. Agarwal and C. Sarkar, "Cannot avoid penalty for fluctuating order arrival rate? let's minimize," in *2019 IEEE/RSJ Int. Conf. on Intelligent Robots and Syst. (IROS)*. IEEE, 2019, pp. 7386–7393.
- [16] P. Visalakshi and S. Sivanandam, "Dynamic task scheduling with load balancing using hybrid particle swarm optimization," *Int. J. Open Problems Comput. Math.*, vol. 2, no. 3, pp. 475–488, 2009.
- [17] A. S. Chavan and M. Brocanelli, "Towards high-quality battery life for autonomous mobile robot fleets," in *IEEE Int. Conf. on Autonomic Comput. and Self-Organizing Syst. (ACSOS)*, 2022, pp. 61–70.
- [18] S. T. Atik, A. S. Chavan, D. Grosu, and M. Brocanelli, "A maintenance-aware approach for sustainable autonomous mobile robot fleet management," *IEEE Trans. Mobile Comp.*, vol. 23, no. 6, pp. 7394–7407, 2024.
- [19] S. T. Atik, D. Grosu, and M. Brocanelli, "Sustainability-aware online task and charge allocation for autonomous ground robot fleets," in *IEEE Int. Conf. on Autonomic Comp. and Self-Organizing Syst. (ACSOS)*, 2024, pp. 11–20.
- [20] A. Botros, B. Gilhuly, N. Wilde, A. Sadeghi, J. Alonso-Mora, and S. L. Smith, "Optimizing task waiting times in dynamic vehicle routing," *IEEE Robotics & Automation Lett.*, vol. 8, no. 9, pp. 5520–5527, 2023.
- [21] L. Zhang, J. Zhao, E. Lamon, Y. Wang, and X. Hong, "Energy efficient multi-robot task allocation constrained by time window and precedence," *IEEE Trans. Automation Sci. and Eng.*, pp. 1–12, 2023.
- [22] M. Braquet and E. Bakolas, "Greedy decentralized auction-based task allocation for multi-agent systems," *IFAC-PapersOnLine*, vol. 54, no. 20, pp. 675–680, 2021.
- [23] Y. Shen, B. McClosky, J. W. Durham, and M. M. Zavlanos, "Multi-agent reinforcement learning for resource allocation in large-scale robotic warehouse sortation centers," in *2023 62nd IEEE Conf. on Decision and Control (CDC)*, 2023, pp. 7137–7143.
- [24] P. Chowdhury and C. Chakrabarti, "Static task-scheduling algorithms for battery-powered dvs systems," *IEEE trans. on very large scale integration (VLSI) syst.*, vol. 13, no. 2, pp. 226–237, 2005.
- [25] J. Kwak, K. Lee, T. Kim, J. Lee, and I. Shin, "Battery aging deceleration for power-consuming real-time systems," in *2019 IEEE Real-Time Syst. Symposium (RTSS)*. IEEE, 2019, pp. 353–365.
- [26] L. He, Y.-C. Tung, and K. G. Shin, "icharge: User-interactive charging of mobile devices," in *Proc. 15th Annual Int. Conf. on Mobile Syst., Applications, and Services*, 2017, pp. 413–426.
- [27] J. Chen and J. Xie, "Joint task scheduling, routing, and charging for multi-uav based mobile edge computing," in *ICC 2022-IEEE Int. Conf. on Communications*. IEEE, 2022, pp. 1–6.
- [28] J. Shi, T. Zeng, and S. Moura, "Electric fleet charging management considering battery degradation and nonlinear charging profile," *Energy*, vol. 283, p. 129094, 2023.
- [29] J. Palacín, E. Rubies, R. Bitriá, and E. Clotet, "Path planning of a mobile delivery robot operating in a multi-story building based on a predefined navigation tree," *Sensors*, vol. 23, no. 21, p. 8795, 2023.
- [30] Z. Zhang, L. Wu, B. Zhang, S. Jia, W. Liu, and T. Peng, "Energy-efficient path planning for a multi-load automated guided vehicle executing multiple transport tasks in a manufacturing workshop environment," *Environmental Science and Pollution Research*, pp. 1–21, 2024.
- [31] Madison Kinner, "Food delivery robots to return to campus this fall," <https://www.thelantern.com/2022/06/food-delivery-robots-to-return-to-campus-this-fall/>, 2022.
- [32] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," *AI magazine*, vol. 29, no. 1, pp. 9–9, 2008.
- [33] Amazon, "How amazon robots navigate congestion," <https://www.amazon.science/latest-news/how-amazon-robots-navigatecongestion>, 2022.
- [34] M. Wei and V. Isler, "Air to ground collaboration for energy-efficient path planning for ground robots," in *IEEE/RSJ IROS*, 2019.
- [35] B. Xu, A. Oudalov, A. Ulbig, G. Andersson, and D. S. Kirschen, "Modeling of lithium-ion battery degradation for cell life assessment," *IEEE Trans. on Smart Grid*, vol. 9, no. 2, pp. 1131–1140, 2016.
- [36] R. Li, X. Cheng, and Y. Zhou, "Online scheduling for jobs with nondecreasing release times and similar lengths on parallel machines," *Optimization*, vol. 63, no. 6, pp. 867–882, 2014.
- [37] J. Munkres, "Algorithms for the assignment and transportation problems," *J. of the SIAM*, vol. 5, no. 1, pp. 32–38, 1957.
- [38] D. S. Moore, G. P. McCabe, and B. A. Craig, *Introduction to the Practice of Statistics*, 9th ed. New York: W.H. Freeman and Company, 2017.
- [39] H. Jahanshahi, A. Bozanta, M. Cevik, E. M. Kavuk, A. Tosun, S. B. Sonuc, B. Kosucu, and A. Başar, "A deep reinforcement learning approach for the meal delivery problem," *Knowledge-Based Syst.*, vol. 243, p. 108489, 2022.
- [40] S. T. Atik, D. Grosu, and M. Brocanelli, "BTC-X repository," https://github.com/tanjila2020/TSUSC2025_Enhanced_BTC_Repo, 2025.
- [41] I. D. Dinov, *Data Science and Predictive Analytics: Biomedical and Health Applications Using R*. Springer, 2018. [Online]. Available: <https://link.springer.com/book/10.1007/978-3-319-72347-1>

BIOGRAPHIES

Syeda Tanjila Atik is a Ph.D. candidate in the Department of Electrical and Computer Engineering at the Ohio State University and a student member of the Energy-aware Autonomous Systems Lab (EAS-Lab). She earned her B.Sc. and M.Sc. in Information Technology from Jahangirnagar University, Bangladesh in 2015 and 2017. Her research interests are in the area of edge computing, Internet of Things, autonomous mobile robots, and machine learning.



Daniel Grosu (Senior Member, IEEE) received the Diploma in engineering from the Technical University of Iași, Romania, in 1994 and the MSc and PhD degrees in computer science from the University of Texas at San Antonio in 2002 and 2003, respectively. Currently, he is a professor in the Department of Computer Science, Wayne State University, Detroit. He is an *IEEE Computer Society Distinguished Contributor*. He serves as a senior associate editor for *ACM Computing Surveys* and as an associate editor for *IEEE Trans. on Parallel and Distributed Systems*, and *IEEE Trans. on Cloud Computing*. He is a Distinguished Member of the ACM, a senior member of the IEEE, and the IEEE Computer Society.



Marco Brocanelli (Senior Member, IEEE) is an Assistant Professor in the Department of Electrical and Computer Engineering at The Ohio State University and the director of the Energy-aware Autonomous Systems Lab (EAS-Lab). He received the B.E. and M.E. degrees in Control Systems from the University of Rome Tor Vergata (Italy). In August 2018, he received the Ph.D. degree in the Electrical and Computer Engineering program of The Ohio State University. His research interests are in the area of cyber-physical systems, energy-aware systems, Internet of Things (IoT), edge computing, embedded and real-time systems. In 2022 he received the CAREER award from NSF and he currently serves as an associate editor for *ACM Computing Surveys*.

