Sustainability-aware Online Task and Charge Allocation for Autonomous Ground Robot Fleets

Syeda Tanjila Atik*, Daniel Grosu†, Marco Brocanelli*
*The Ohio State University, †Wayne State University

***Abstract*—Ensuring low battery degradation of Autonomous Ground Robot (AGR) fleets operating in online scenarios (e.g., delivery) through an efficient task and charge scheduling strategy can significantly enhance their long-term sustainability. Most existing studies are either based on offline methods, which are unsuitable for online scenarios requiring instant decisions, or concentrated on maximizing task allocation, resource usage, and/or revenues without considering the battery health. To overcome these limitations, this paper proposes a family of two joint task allocation and charge scheduling algorithms that activate at specific events to maximize the total revenue while minimizing the battery degradation of the fleet in online scenarios. Utility functions are defined to trade off revenues for battery degradation while deciding, for all the AGRs in the fleet, how to allocate tasks, charging stations, and idle periods. The first algorithm is based on the Kuhn-Munkres approach that makes decisions at each event optimally. The second algorithm utilizes a greedy approach achieving a sub-optimal solution with reduced computational overhead. Our results, obtained through extensive simulations based on a real AGR against several baselines, show that it is possible to achieve up to 20% longer battery lifespan with minimal revenue losses.**

I. Introduction

The presence of autonomous ground robots (AGRs) has increased significantly in various sectors of our society, including food and good deliveries. For example, recently Grubhub has partnered with robot delivery companies such as Starship [1] and Cartken [2] to deploy their full service in many US university campus areas [3]–[6], which exponentially increased the completed deliveries from 500K in 2020, 2M in 2021, to 5M in 2023 [1]. Different from *offline* cases where the tasks to execute are known and fixed, such delivery (or similar) scenarios have an *online* nature, where tasks characteristics (e.g., value, pick-up/drop-off locations) are not known until they arrive and require quick response time from the allocation algorithm. To provide smooth operations, a vital component to take care of for any AGR is its battery. In fact, the maximum capacity of any rechargeable battery (i.e., maximum amount of energy storable) decreases over time, with the battery becoming unusable when its maximum capacity is significantly reduced. Typically, a battery with an 80% remaining capacity is considered at the end of its life owing to the faster degradation after each subsequent use [7]. The battery degradation is heavily influenced by how the AGR's task and recharge schedules are coordinated in the fleet. Without careful usage, the increasing demand of AGRs in various aspects of our society could potentially lead to an unsustainable increase in demand for new batteries. Thus, it is important to design online algorithms that schedule tasks and AGR charging not only to maximize the total revenues, but also to minimize the impact on the battery health of the entire AGR fleet.

The majority of the existing works [8]–[15] mostly focus 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 [16]–[18]. However, they often overlook the long-term impacts of the task scheduling decisions on the battery health. Our previous studies (e.g., [19], [20]) 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 this paper, we present a family of online Battery degradation-aware Task and Charge allocation algorithms, BTC-X, that maximizes revenues and battery lifespan leveraging either an optimal but slower approach (BTC-M) or a greedy but faster one (BTC-G) to make online decisions when an event occurs (e.g., task arrival). In particular, we translate the allocation problem into that of finding a maximum matching between the edges of a bipartite graph, where the edges connect each AGR to the available tasks, charging stations, and idle vertices (in case an AGR is best left waiting for a new task). We define the edge weights to evaluate trade-offs between task revenues, the degradation of the battery incurred to execute each task, or the degradation to schedule a recharge to a specific charging station location. Then, either one of the two variations, BTC-M and BTC-G, finds the optimal or sub-optimal allocation for each AGR leveraging the Kuhn-Munkres algorithm [21] or a greedy approach, respectively. We use our prototype AGR to generate realistic simulation scenarios. Our extensive results make the important contribution of proving the possibility of embedding sustainability into the future operations of AGR fleets with little revenue losses.

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

Specifically, this paper makes the following contributions:

- We formulate the online sustainability-aware task and charge allocation problem, and design a family of algorithms called BTC-X to find a solution in polynomial time at each event occurrence.
- We design two algorithms of the family to find a solution to the formulated problem, BTC-M and BTC-G. The former finds an optimal solution but can be slower than the latter, which finds a sub-optimal solution leveraging a greedy algorithm.
- We leverage a real AGR to generate realistic simulated scenarios. Compared to several baselines, our extensive evaluation shows that both BTC-M and BTC-G can achieve up to 20% longer battery lifespan with minimal losses in the generated revenues.

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 and Section V concludes the paper.

II. Related work

Maximizing Task Allocation. Jeon et al. [8] aimed at maximizing the number of tasks completed by assigning each task to the nearest AGR, thus minimizing travel distance. Liu et al. [9] proposed two offline Multi-Agent Pickup-and-Delivery (MAPD) algorithms for task and path planning of AGRs based on the traveling salesman problem. In [10], [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 a market-based task allocation approach [12]–[15], providing decentralized and distributed algorithms. Although they perform well to maximize task allocation, they are better suited for *offline* scenarios where task characteristics are known in advance.

A few other studies focused on online algorithms for maximizing task allocation [16]–[18]. For example, Agarwal and Sarkar [16] 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 [22]–[25]. For example, some solutions [22], [23] proposed algorithms that reduce the discharge rate and leverage voltage scaling to alleviate battery degradation while meeting task deadlines. Kwak et al. [24] proposed a task scheduling approach where the battery temperature is controlled to minimize the degradation rate. He et al. [25] developed a customized charging strategy that adjusts the charging rates of mobile devices based on the availability of the user's time, which is mindful of the device's need for periodic relaxation to prolong battery life. In contrast to our approach, these studies primarily concentrate on extending battery life without addressing the coordination of AGRs for both online task and charge scheduling, which is crucial for optimizing both task performance and battery health.

Joint Task and Charge Scheduling. Focusing on the joint task and charge scheduling problem, Chen and Xie [26] 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. [27] 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 [19], [20] presented offline joint task and charge scheduling algorithms for AGRs. However, they only consider energy thresholds to calculate the cost of battery degradation while recharging or discharging without considering the actual battery capacity degradation of each AGR in the optimization problem. Thus, these approaches cannot capture the actual battery degradation for a new or older robot. In addition, these proposed solutions only consider offline scenarios. *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 scenario involving a fleet of AGRs that carry out delivery tasks within a limited area. These tasks typically involve picking up goods from specified locations and delivering them to designated drop-off locations. The operational area can be indoor or outdoor, with a fixed number of charging locations. A central coordination system processes the delivery task requests and assigns them to the AGRs. The AGRs are considered to be homogeneous (same components) and can navigate autonomously, following optimal routes which is assumed to be calculated by a global navigation system and are provided as inputs to the AGRs. The generation of these efficient routes, however, is beyond the scope of this paper and is extensively covered by other studies [9], [28], [29].

As a real life example of delivery using an AGR fleet, many university campuses have a fleet of AGRs (e.g., 40 AGRs [30]) that deliver food from various locations around campus to different buildings and residence halls. *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).

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:

$$R_i(k) =< l_i^r(k), e_i(k), d_i(k), t_i^c, s_i^r(k) >$$

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:

$$C_j(k) =< l_j^c, s_j^c(k) >$$

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:

$$T_h(k) =< l_h^p, l_h^d, v_h, a_h, f_h, s_h^t(k) >$$

where l_h^p and l_h^d represent the pick-up and drop-off 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 [31]:

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

where δ is a period of time (e.g., to travel from the current location to the task pick-up and then drop-off 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 [32], 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 [32]. On the other hand, in our algorithm we abstract the degradation algorithm proposed in [32] 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 [32] 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 $|\mathcal{A}^{\mathcal{R}}| \times (|\mathcal{A}^{\mathcal{T}}| + |\mathcal{A}^{\mathcal{C}}| + |\mathcal{I}|)$ is initialized to 0 and is then filled for allocation decisions. The $|\mathcal{A}^{\mathcal{R}}|$ rows correspond to the idle AGRs, the first $|\mathcal{A}^{\mathcal{T}}|$ columns correspond to the available task vertices yet to be allocated, the $|\mathcal{A}^{\mathcal{C}}|$ 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. We propose to calculate the utility $U_{ih}(k)$ for allocating task T_h to the AGR R_i at time k as:

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

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. We propose to calculate the utility $W_{ij}(k)$ for scheduling AGR R_i to charge at station C_j at time k as:

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

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 a configurable value 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.

Idle Utility. For each AGR R_i, the utility of the idle

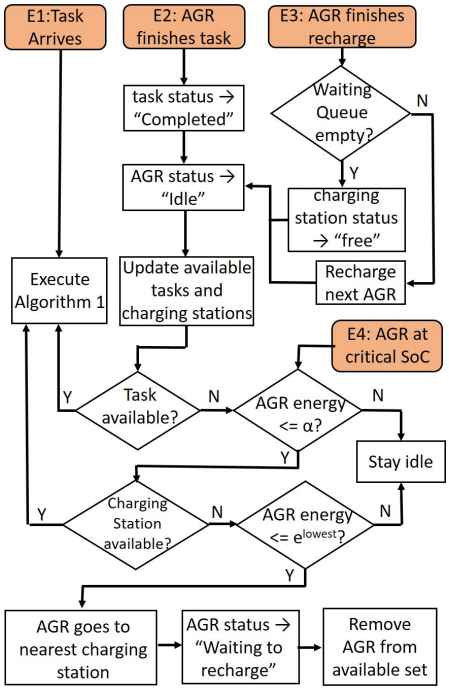


Fig. 1. High level logic of the proposed online BTC-X family of algorithms.

vertex Z_i at time k is simply set to a sufficiently small value λ:

$$Z_i(k) = \lambda \tag{4}$$

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 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 [17], [33].

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., $\leq e^{\text{lowest}}$, which is enough to guarantee safe travel to any charging station and to wait for recharge), reach the nearest charging station to wait in a queue for a recharge when E3 occurs.

F. Algorithm 1: Allocation

As mentioned in the previous section, the allocation Algorithm 1 is executed for several reasons that require evaluating 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). In general, the algorithm first updates the system parameters of the available tasks, AGRs, and charging stations, by querying them through network communications (Lines 2-6 of Algorithm 1). If there are idle AGRs (Line 7), i.e., waiting to be allocated to task or charging station, the algorithm first updates the utility matrix as described in Section III-D (Lines 8-24). 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 $\mathcal{A}^{\mathcal{R}}$, task ID in the set $\mathcal{A}^{\mathcal{T}}$, charging station ID in the set $\mathcal{A}^{\mathcal{C}}$, and idle vertex ID in the set $\mathcal{I}$ (Lines 9-10). The bipartite graph's edge weights, i.e., utility values of the matrix, are then calculated according to Equations 2, 3, and 4 (Lines 11-24).

At this point, the SELECTION() function (described in details in the next paragraph) analyzes the utility values to finalize the allocation choices (Line 25). 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 27-31), 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 $\mathcal{A}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{T}}$, respectively. If an AGR

Algorithm 1 Allocation

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

is allocated to a charging station (Lines 32-37), the algorithm sets the AGR status to `Charging` to the selected charging station, which changes status to `Occupied`, and the sets $\mathcal{A}^{\mathcal{R}}$ and $\mathcal{A}^{\mathcal{C}}$ 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 $t^c_{\sigma(i)}$. If no task arrives before the end of such configurable period t^{max} (Line 33), 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 and 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 of $\mathcal{X}$ tuples 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 two algorithms that define the two algorithms that are members of the BTC-X family, BTC-M and BTC-G. The former is the weighted matching algorithm based on the Kuhn-Munkres approach [21] (called WEIGHTED_MATCHING() here), which finds the optimal allocation for the current event but it might incur larger overhead, particularly in large-scale scenarios. For the latter, corresponding to BTC-G, we propose a new algorithm called GREEDY_SELECTION(), which utilizes a greedy approach to provide a sub-optimal allocation with reduced overhead.

Algorithm 2 GREEDY_SELECTION()

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

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 $O(|\mathcal{A}^{\mathcal{R}}| \cdot (|\mathcal{A}^{\mathcal{T}}| + |\mathcal{A}^{\mathcal{C}}| + |\mathcal{I}|))$. After that the SELECTION() function in Line 25 is executed, which uses the WEIGHTED_MATCHING() algorithm with

TABLE I
SIMULATION PARAMETER VALUES

Parameter	Value	Parameter	Value
CPU Frequency	2.26GHz	SoC Threshold (α)	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 Consumption (p)	38W	Max SoC Recharge	80%

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

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 $O(|\mathcal{A}^{\mathcal{R}}|^2 \cdot (|\mathcal{A}^{\mathcal{T}}| + |\mathcal{A}^{\mathcal{C}}| + |\mathcal{I}|))$. Earlier we, have shown that the running time to execute Algorithm 1 without the SELECTION() function is $O(|\mathcal{A}^{\mathcal{R}}| \cdot (|\mathcal{A}^{\mathcal{T}}| + |\mathcal{A}^{\mathcal{C}}| + |\mathcal{I}|))$. Thus the execution time of BTC-G is $O(|\mathcal{A}^{\mathcal{R}}|^2 \cdot (|\mathcal{A}^{\mathcal{T}}| + |\mathcal{A}^{\mathcal{C}}| + |\mathcal{I}|))$, i.e., lower than that of BTC-M.

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.

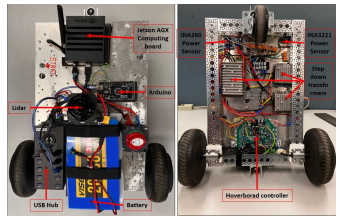


Fig. 2. Our prototype used to profile an AGR power consumption.

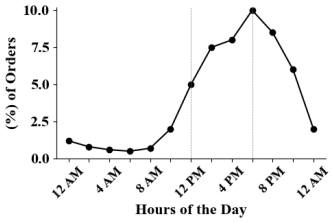


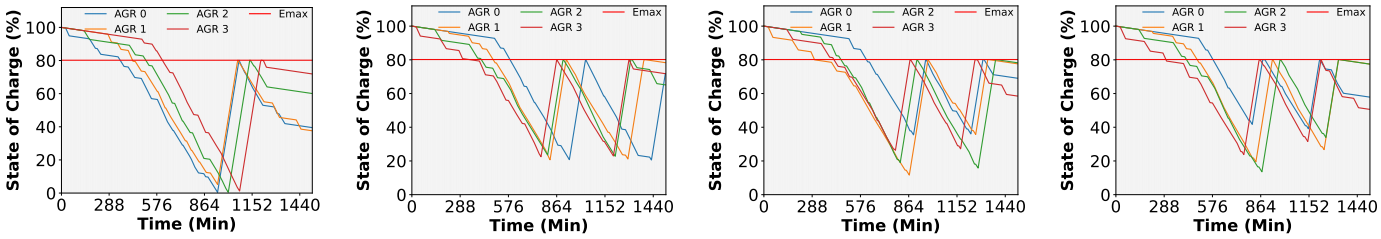
Fig. 3. Hourly distribution of delivery orders for a day [34].

Experiment Scenario. To make the experiments realistic, we have designed a practical scenario simulating the delivery of food orders across a university campus using AGRs. Typically, the operation area of such deliveries is limited. Thus, we have considered an area of 1km^2 divided in squares of $1m^2$. The task arrival, representing food delivery requests, typically has peaks during lunch and dinner hours [34], as Figure 3 shows. 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 (uniform distribution) generate pick-up and drop-off 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 set to 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 [35].*

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 [16]–[18],



(a) (b) (c) (d)

Fig. 4. Comparison of the State-of-charge (SoC) of the AGRs among the baselines (a) REV, (b) DEG, and the proposed (c) BTC-G, and (d) BTC-M considering a sample experiment with four AGRs where AGR 0 and 3 have older and AGR 1 and 2 have almost new batteries.

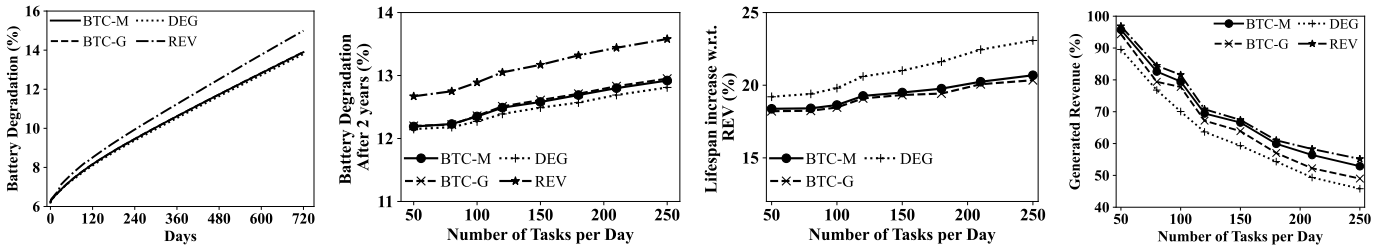
which can be translated into purely maximizing the total revenue generated without considering battery lifespan. To further help maximize the obtained revenues, REV considers as task value the ratio of task revenue over the travel distance. The second baseline, DEG, prioritizes minimizing battery degradation 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 target (DEG). For charging decisions, REV charges only when an AGR energy level reaches the minimum e^{lowest} to maximize task allocation, while DEG limits the depth of discharge by charging AGRs when they reach a minimum threshold of 20% SoC. Decisions are then 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 two variants of our proposed approach, BTC-M, and BTC-G.

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

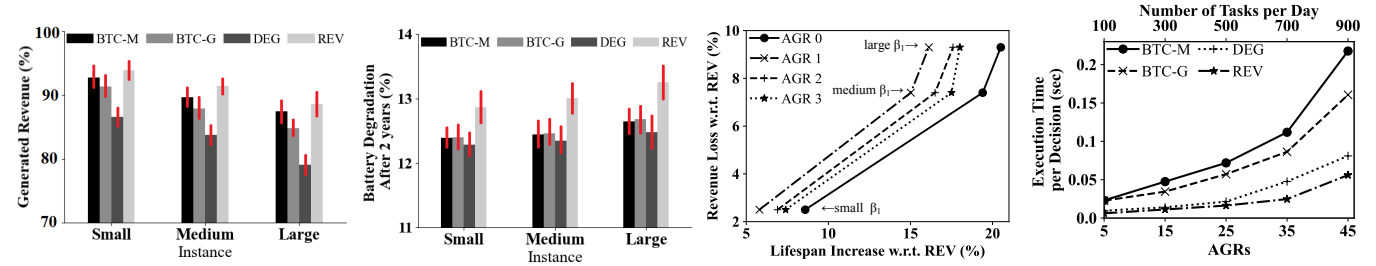
In this section, we provide a detailed description of the operation of the baselines, REV, DEG and our proposed algorithms BTC-M and BTC-G on an example case study. We consider a sample experiment with 4 AGRs, 80 tasks per day, and 2 different charging locations each having two charging stations. Here, we have considered AGRs with almost new (AGR 1 and 2) and older (AGR 0 and 3) batteries, thus they have different degradation levels in the beginning of the simulation. We run this experiment for a duration of two years to be able to better evaluate the evolution of the battery capacity degradation of the AGRs. For enhanced clarity, we show the SoC profiles in a single day of the AGRs with each approach in Figure 4. REV (Figure 4a), since it maximizes task allocation, continues to assign tasks to the AGRs until their energy level is critically low and need to recharge. Although this strategy can generate better revenue by allocating more tasks, it leads to significant battery degradation, particularly when AGRs with older batteries (e.g,. AGR 0) are recharged at a higher depth of discharge. Conversely, DEG (Figure 4b) prioritizes allocating tasks to AGRs solely based on minimal battery degradation, disregarding potential revenue gain. The AGRs are recharged when their SoC drops below 20% (similar to [19], [20]) to prevent further degradation. Despite achieving higher battery lifespan, it may result in substantial revenue losses.

On the other hand, our proposed approaches BTC-G (Figure 4c) and BTC-M (Figure 4d), incorporate considerations of both revenue generation and battery degradation into their task or charge allocation choices. This balanced approach prevents deep discharge cycles by scheduling recharges for AGRs with older batteries (such as AGR 0 and 3 in Figure 4d) at higher SoC levels and thus, may generate higher revenue while keeping the battery degradation lower. BTC-M aims for an optimal solution at a certain decision event where as BTC-G finds a sub-optimal solution, leading to slightly different allocation choices as seen in the figures. Upon analyzing the performance of these four approaches we have found that while REV achieves the highest generated revenue of 86%, DEG yields the lowest at 77%. On the other hand, BTC-G and BTC-M achieve 80% and 83% revenue ratio, respectively (reported as part of next section, 80 tasks case in Figure 5d). Furthermore, we have shown the trend in battery capacity degradation of AGR 0 over two years for these four approaches in Figure 5a. We can see that with DEG, AGR 0 reaches a reference battery degradation of 13.7% after 720 days (2 years), while with REV it reaches 15% after the same number of days. With both of our approaches, the capacity degradation over two years is similar to that of DEG. In other words, **we observe that REV reaches the reference battery degradation for AGR 0 after only 600 days, compared to the 720 days of BTC-G and BTC-M, with the optimal BTC-M leading to higher revenue than BTC-G at same time. Thus, our approaches lead AGR 0**



(a) (b) (c) (d)

Fig. 5. (a) Battery degradation of AGR 0 from the experiment shown in Figure 4 over 2 years. (b) average battery degradation after 2 years, (c) lifespan increase with respect to REV of AGR batteries across DEG and proposed BTC-G and BTC-M, and (d) generated revenue, for an increasing task demand.



(a) (b) (c) (d)

Fig. 6. Comparison of the (a) generated revenue and (b) average capacity degradation of AGR batteries after 2 years. (c) Revenue loss vs battery lifespan increase for three battery weight values (β_1 in Equation 2) with BTC-M relative to REV. (d) Execution time per allocation decision across the schemes.

to a 20% increase in lifespan w.r.t. REV, which shows how small gains in battery degradation of 1-2% can lead to a substantially longer lifespan.

C. Effect of Increased Demand

We are also interested to see the performance of our approach for increased demand. Thus, we run other experiments where keep the number of AGRs fixed at 4 while increasing the daily tasks from 50 to 250. Figure 5b illustrates the average battery capacity degradation among AGRs under increased task demands. The capacity degradation slightly increases with a larger volume of tasks, likely due to the AGRs not always being recharged at the ideal SoC to meet the higher task execution demand. However, both of our BTC-M and BTC-G approaches manage to achieve a percentage decrease in capacity degradation of 4% compared to REV, which in turn leads to a 20% increase in lifespan (see Figure 5c), even under higher task demands. This degradation is only less than 1% higher than that achieved by DEG leading to only a 3% lower lifespan compared to it as seen in Figure 5c.

Although DEG manages to exhibit a slightly higher lifespan, it generates only 45% revenue at 250 tasks per day while BTC-M and BTC-G generate 20% (similar to REV) and 10% higher revenue than DEG, respectively, as observed in Figure 5d. The slight under-performance of BTC-G compared to BTC-M is due to employing a greedy sub-optimal approach.

These experiments demonstrate our approaches can substantially extend battery lifespan with minimal revenue loss despite increasing task demands.

D. Performance Comparison with Different Problem Instances

To comprehensively evaluate the performance of our approaches, we have conducted further experiments considering small, medium, and large problem instances (outlined in the experimental setup). Figure 6a and 6b show the results of those experiments with the red bars indicating the standard deviation. It is observed from Figure 6a that the revenue generated by our BTC-M and BTC-G methods consistently exceeds that of DEG by 7%, 7.2% and 10% on average, for small, medium, and large instances, respectively. While the revenue from REV is marginally higher by 1%, 2% and 1.5% for these cases, this slight increase in revenue is accompanied by significantly higher battery capacity degradation. As seen from Figure 6b, the average capacity degradation of AGR batteries is increased by around 5% in case of REV compared to our approaches. It is important to note that this seemingly small improvement in capacity degradation leads to a substantially longer lifespan (see Figure 5b and 5c). We further show this by discussing the revenue loss vs increase in lifespan with respect to REV for a small instance of 4 AGRs.

Results with Different Weight Factors. The task utility (Equation 2) of our proposed approaches use a configurable parameter β_1 to weight the maximization of revenues over minimization of battery degradation. Here, we explore how reducing or increasing this parameter value can lead to different trade offs between these two objectives. We use a 2-year experiment with 4 AGRs at different initial degradation levels and three weight values, small (more revenues), medium, and large (longer lifespan). Figure 6c shows the results. We calculate the revenue losses and increased lifespan of BTC-M relative to REV. While a much better lifespan can be achieved for larger weight values, the increasing revenue losses may become unattractive to industry. However, even just accepting small revenue losses can lead to substantial improvements in the sustainability objective. For example, a loss of 2.5% revenues can lead to 6-9% increase in the battery lifespan of the AGRs. Additionally, if the focus is more on sustainability, accepting 9% revenue loss can lead to 16-21% increase in the

battery lifespan. *We hope that these findings can push industry to consider sustainability in the operations of AGR fleets.*

Comparison of Execution Time. As our proposed approaches and the baselines are implemented for online solutions, we evaluate the execution time for each allocation decision across these methods in Figure 6d. Notably, the execution time for all of the approaches increase with the size of the problem instance. REV exhibits the shortest execution times, ranging from 0.006 seconds to 0.05 seconds. This efficiency stems from REV's simpler operational logic, which does not require analyzing augmented SoC traces to estimate battery degradation and primarily relies on fixed rules for recharging. Conversely, DEG incurs a slightly longer time, from 0.009 to 0.08 seconds, due to the fact that it has to analyze the SoC trace to estimate the degradation level while allocating tasks. Our approaches, BTC-M and BTC-G require more time to make allocation decisions because of the two-objective considerations as well as the integration of both task and recharge decisions. However, it only takes 0.21 seconds for BTC-M and 0.16 seconds for BTC-G to provide each allocation decision even in scenarios involving 45 AGRs and 900 tasks per day, which is reasonable for online scenarios.

These results prove the effectiveness of our sustainability-aware allocation algorithms in real-world scenarios.

V. Conclusion and Future Work

In this paper, we proposed a family of two online joint task allocation and charge scheduling algorithms that allocate tasks across AGRs in a fleet for maximized revenue and minimized battery degradation. While one of the proposed algorithms finds optimal allocations, the other algorithm uses a greedy approach to find sub-optimal solutions with shorter execution times. We leveraged a real-world inspired simulation setup and conducted extensive simulations based on a real AGR. Our simulation results demonstrated the possibility of achieving up to 20% longer battery lifespan with minimal revenue losses compared to the state-of-the-art inspired baselines. In future, we plan to extend our work to include statistical prediction of task arrivals while making allocation decisions along with the consideration of heterogeneous AGR components.

References

[1] Starship, "Home," https://www.starship.xyz/company/, 2022.

[2] Cartken, "Cartken Home Page," https://www.cartken.com, 2023.

[3] 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.

[4] Best Colleges, "Robots Invade College Campuses," https://www.bestcolleges.com/news/robots-invade-college-campuses/, 2023.

[5] 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.

[6] Joe Guszkowski - 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.

[7] Nadim Maluf, "What happens after 80https://www.qnovo.com/blogs/what-happens-after-80-percent, 2014.

[8] S. Jeon, J. Lee, and J. Kim, "Multi-robot task allocation for real-time hospital logistics," in *IEEE SMC*, 2017.

[9] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and path planning for multi-agent pickup and delivery," in *AAMAS*, 2019.

[10] L. Luo, N. Chakraborty, and K. Sycara, "Distributed algorithm design for multi-robot task assignment with deadlines for tasks," in *IEEE ICRA*, 2013.

[11] C. Sarkar, H. S. Paul, and A. Pal, "A scalable multi-robot task allocation algorithm," in *IEEE ICRA*, 2018.

[12] D.-H. Lee, "Resource-based task allocation for multi-robot systems," *Robotics and Autonomous Systems*, vol. 103, pp. 151–161, 2018.

[13] 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.

[14] Z. Zhu, B. Tang, and J. Yuan, "Multirobot task allocation based on an improved particle swarm optimization approach," *International Journal of Advanced robotic systems*, vol. 14, no. 3, p. 1729881417710312, 2017.

[15] G. Ferri, J. Bates, P. Stinco, A. Tesei, and K. LePage, "Autonomous underwater surveillance networks: A task allocation framework to manage cooperation," in *IEEE OTO*, 2018.

[16] M. Agarwal and C. Sarkar, "Cannot avoid penalty for fluctuating order arrival rate? let's minimize," in *IEEE/RSJ IROS*, 2019.

[17] N. Thibault and C. Laforest, "Online time-constrained scheduling problem for the size on\kappa machines," in *IEEE ISPAN*, 2005.

[18] P. Visalakshi and S. Sivanandam, "Dynamic task scheduling with load balancing using hybrid particle swarm optimization," *Int. J. Open Problems Compt. Math*, vol. 2, no. 3, pp. 475–488, 2009.

[19] A. S. Chavan and M. Brocanelli, "Towards high-quality battery life for autonomous mobile robot fleets," in *IEEE ACSOS*, 2022, pp. 61–70.

[20] S. T. Atik, A. S. Chavan, D. Grosu, and M. Brocanelli, "A maintenance-aware approach for sustainable autonomous mobile robot fleet management," *IEEE Transactions on Mobile Computing*, vol. 23, no. 6, pp. 7394–7407, 2024.

[21] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[22] J. Luo and N. K. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," in *ACM DAC*, 2001.

[23] P. Chowdhury and C. Chakrabarti, "Static task-scheduling algorithms for battery-powered dvs systems," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 13, no. 2, pp. 226–237, 2005.

[24] J. Kwak, K. Lee, T. Kim, J. Lee, and I. Shin, "Battery aging deceleration for power-consuming real-time systems," in *IEEE RTSS*, 2019.

[25] L. He, Y.-C. Tung, and K. G. Shin, "iCharge: User-interactive charging of mobile devices," in *ACM MobiSys*, 2017.

[26] J. Chen and J. Xie, "Joint task scheduling, routing, and charging for multi-uav based mobile edge computing," in *IEEE ICC*, 2022.

[27] J. Shi, T. Zeng, and S. Moura, "Electric fleet charging management considering battery degradation and nonlinear charging profile," *Energy*, vol. 283, p. 129094, 2023.

[28] 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.

[29] 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.

[30] 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.

[31] M. Wei and V. Isler, "Air to ground collaboration for energy-efficient path planning for ground robots," in *IEEE/RSJ IROS*, 2019.

[32] B. Xu, A. Oudalov, A. Ulbig, G. Andersson, and D. S. Kirschen, "Modeling of lithium-ion battery degradation for cell life assessment," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 1131–1140, 2016.

[33] 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.

[34] 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 Systems*, vol. 243, p. 108489, 2022.

[35] S. T. Atik, D. Grosu, and M. Brocanelli, "BTC-X repository," https://github.com/tanjila2020/BTC_repository_ACSOS_2024, 2024.