# Reinforcement Learning for AMR Charging Decisions: The Impact of Reward and Action Space Design

Janik Bischoff[1][0009−0007−6592−9768], Alexandru Rinciog[2][0000−0003−0330−6737], and Anne Meyer[1][0000−0001−6380−1348]

[1] Karlsruher Institut für Technologie, Zirkel 2, 76131 Karlsruhe
[2] SLAPStack, Joseph-von-Fraunhofer-Straße 2-4, 44227 Dortmund
alexandru.rinciog@slapstack.de
{janik.bischoff,anne.meyer}@kit.edu

**Abstract.** We propose a novel reinforcement learning (RL) design to optimize the charging strategy for autonomous mobile robots in large-scale block stacking warehouses. RL design involves a wide array of choices that can mostly only be evaluated through lengthy experimentation. Our study focuses on how different reward and action space configurations, ranging from flexible setups to more guided, domain-informed design configurations, affect the agent performance. Using heuristic charging strategies as a baseline, we demonstrate the superiority of flexible, RL-based approaches in terms of service times. Furthermore, our findings highlight a trade-off: While more open-ended designs are able to discover well-performing strategies on their own, they may require longer convergence times and are less stable, whereas guided configurations lead to a more stable learning process but display a more limited generalization potential. Our contributions are threefold. First, we extend SLAPStack, an open-source, RL-compatible simulation-framework to accommodate charging strategies. Second, we introduce a novel RL design for tackling the charging strategy problem. Finally, we introduce several novel adaptive baseline heuristics and reproducibly evaluate the design using a Proximal Policy Optimization agent and varying different design configurations, with a focus on reward.

**Keywords:** Autonomous Block Stacking Warehouses · Vehicle Dispatching · Discrete Event Simulation · Battery Management · Reinforcement Learning · AGV

## 1 Introduction

With increasing uncertainty and supply chain disruptions, innovative logistics solutions are becoming vital. The adoption of autonomous mobile robots (AMR) in block storage warehouses (BSW) reflects this trend.

In a BSW, goods are stored directly on the floor or on top of each other. This storage system has several benefits, such as low investments due to the

minimal infrastructure needed and high throughput. While pallets are stored in lanes next to each other for higher storage efficiency, aisles and cross-aisles are used by vehicles to travel through the BSW.

The authors of [17] have presented the autonomous block stacking warehouse problem (ABSWP), which consists of interdependent decision problems in combination with the usage of AMRs in a BSW. Problems include the storage location assignment problem (SLAP), which assigns incoming items to storage locations, the unit load selection problem (ULSP) to determine which items are retrieved to fulfill an outbound order, and the vehicle dispatching problem, in which vehicles are assigned to transport tasks. Due to the combinatorial complexity of these decision problems, simulation studies are used to derive configurations for a given ABSWP instance. The importance of battery management is often overlooked in such studies. Recognizing the crucial role that battery management plays in autonomous systems, as highlighted in [7, 29], we extend the ABWSP model to include it.

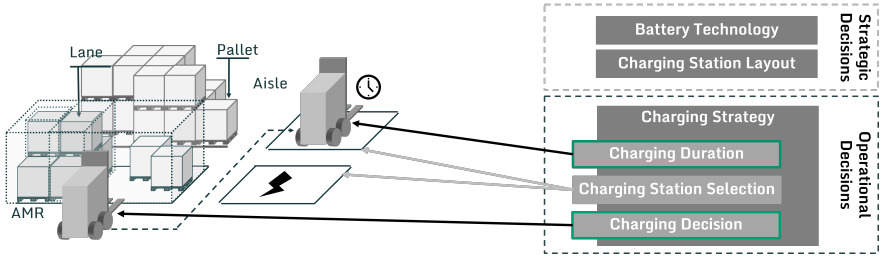



Fig. 1: The problems associated with battery management as per [30].

According to [30], battery management consists of three sub-problems visualized in Figure 1: The selection of battery charging technology (e.g. induction charging, battery swaps), the charging station layout (i.e. the number and position of charging stations), and the battery charging strategy. The former problems are more strategic in nature, while the last problem is operational. The charging strategy, which is the focus of this work, can be further broken down into three sub-problems pertaining to (1) the **charging decision**, i.e. determining when an AMR should go charge, (2) the **charging station selection**, i.e. assigning a specific charging station and the route to that destination to an AMR, and (3) the **charging duration**, i.e. determining how long AMRs should remain at charging stations.

This work employs a reinforcement-learning (RL) meta-heuristic approach to jointly solve the charging decision and charge duration problems. To accomplish this, the agent must decide between discrete charge levels including a no-charge option, at the conclusion of any travel event. We use a fixed heuristic solution for the charging station selection problem and evaluate the resulting charging strategy using a large-scale, real-world ABSWP benchmark — WEPAStacks [18]. Charging strategy solutions can be either exact, heuristic, or meta-heuristic. Exact solutions are intractable for large real-world scenarios such as WEPAStacks.

Simple heuristic solutions are likely not able to adapt to varying AMR demand, resulting in inefficiency. Meta-heuristic solutions provide more adaptability while maintaining tractability. We chose RL as the meta-heuristic paradigm for two reasons. Firstly, in recent years, RL has emerged as a promising approach for solving complex combinatorial optimization problems [11]. Secondly, an RL-compatible ABSWP simulation with mechanisms safeguarding reproducibility was made available in [18], alongside real-world use-case data. This makes the RL training and evaluation effort manageable. Among various RL algorithms, we selected Proximal Policy Optimization (PPO) [23] due to its suitability for practical applications in combinatorial optimization [13].

A key challenge in applying RL lies in the multitude of design choices such as the design of the reward function and action space. We explore different configurations varying the amount of structure imposed on the agent's decision-making process. Our results suggest a tradeoff between learning stability and optimization potential. A more flexible approach to RL-design provides greater optimization potential at the cost of learning stability. Conversely, a more constrained, domain-knowledge informed design, can lead to good results quickly, but may limit the agent's learning potential. Overall, we make the following contributions to the field:

1. Firstly, we **extend SLAPStack**, a fine-grained, rl-compatible, ABSWP simulation framework [18], to account for battery management, and publish the associated code.
2. Secondly, we introduce a **novel Markov decision process (MDP)** for AMR charging.
3. Finally, we train a state-of-the-art RL algorithm on a large-scale ABSWP instance and **reproducibly evaluate several different MDP-configurations** against existing and **novel heuristic strategies**.

We start with an overview of battery charging strategies in literature in Section 2. We then present the problem setting and benchmark instance in Section 3, including a selection and evaluation of baseline charging strategies. In Section 4, we present the different RL design choices for AMR charging, and detail the configurations we evaluate. Finally, we evaluate the proposed configurations in terms of learning stability and generalization capabilities in Section 5, before drawing our conclusions in Section 6.

## 2 Related Work

While some research exists on battery management for AMRs in intralogistics, no existing work offers suitable charging strategies for real-world ABSWPs. Our review shows that exact methods handle only small problems, heuristics lack flexibility, and RL approaches suffer from non-reproducible simulations.

**Learning Free Approaches**: The lower level decisions comprising battery charging strategies, i.e. deciding when to charge, for how long, and which charging station to use, are often tackled using simple heuristic rules or expert knowledge. A common approach for deciding when AMRs should charge is to use fixed

lower-bound battery capacity thresholds. In [25, 3], for instance, the 20% level is chosen to prevent AMRs from blocking production due to insufficient battery levels. A more flexible approach is opportunity charging. Here, charging is done when an opportunity arises. This may occur when vehicles are idle and near a charging station or during off-peak times. For this strategy, it is important that the placement of the charging station coincides with the regions where AMRs are likely to idle, e.g. on the way to a central parking area or near an in- or outbound dock as described in [5].

While modern AMRs are able to partially recharge via inductive or plug-in charging, battery swapping is still widespread. Herein, the empty battery is replaced with a full one at designated swapping stations. Although battery swapping eliminates idle time for the AMR during charging, it introduces significant disadvantages. It is less safe than charge-based techniques and requires special safety measures such as acid-proof floors at the swapping stations, as noted in [5]. For inductive or plug-in charging systems, a common practice is to recharge to a defined upper level, e.g. 50% / 80% / 100% as in [7, 3, 31, 9]. The charging station selection is also often done in a rule-based fashion, e.g. selecting the closest or the least queued charging station [5, 10].

In addition to these myopic strategies, mixed integer linear programming (MILP) is sometimes applied to determine optimal charging schedules for AMR dispatching as in [27, 14]. In both works, the computational efforts to solve larger instances are mentioned. To overcome this Meyer et al formulate a branch-price-and-cut approach which can solve instances with up to 144 tasks in [14]. They model the problem of assigning transport tasks in intralogistic warehouses to AMRs as an electric vehicle routing problem. Two objectives are proposed to minimize completion times and due dates, respectively. The authors compare battery swapping, partial charging, and full recharging strategies. Besides a MILP program the authors of [27] present a matheuristic approach to solve instances larger than 22 tasks. AMRs can partially recharge after a critical battery threshold is reached. For recharging, the nearest charging station is selected.

Metaheuristic approaches have been proposed in several other works. Mousavi et al. employ a hybrid genetic algorithm-particle swarm algorithm to optimize an AMR scheduling process in a flexible manufacturing system [15]. A constraint ensures that the charge level of an AMR is sufficient to fulfill a given transport order. The objective function minimizes both the makespan and number of AMRs required for the transport tasks. The problem sizes range from 6 to 15 transport jobs. Han et al. use a genetic algorithm to solve an AMR scheduling problem in [6]. A constraint prevents AMRs from accepting transportation orders if their battery level is below 20 %. First, a regular scheduling is carried out, then the schedule is repaired to account for necessary charging tasks. The approach is applied to a case study with 29 tasks.

From the problem sizes used in these works, it is evident that MILP and other exact methods are ill-suited for dynamic, online variants of the ABSWP, which require frequent re-optimization in response to new tasks. This is due to the strong interdependence between battery management and other sub-problems,

meaning that storage assignment, routing, and task precedence must be decided jointly with charging-related variables. Proposed Meta-heuristics other than RL also exist, but these are also typically designed for offline settings with limited task volumes.

**Reinforcement Learning Approaches**: In [12] a feature-based SARSA algorithm is formulated to determine the charging duration for AMRs. AMRs below a certain threshold are required to go to a standby area. If then a free charging station is available, the AMR occupies it. The action is to determine the charging duration of each AMR at the charging station according to the system state and the current energy price. The RL approach is compared to an industry heuristic that uses a safety threshold of 20 % and aims to keep the battery level between 40% and 80%. The observation space consists of information on the AMRs in the standby area and at the charging stations, such as battery charge levels, battery ages, and battery types of the AMRs. 50 AMRs are considered with 0.8 transportation tasks arriving each minute over a course of 30 days. Charging decisions are made every time a task arrives. The authors report improved utilization for their approach.

The authors of [4, 16] present an RL approach for an automated warehouse setting. Twin Delayed Deep Deterministic Policy Gradient is used as the RL algorithm. At each time step, a set of AMRs with the lowest battery in the working area has to go charging, and charging AMRs with a battery level above a certain threshold must return to the working area. The action is to select the number of AMRs to go charging and the working battery threshold. A setting with 640 AMRs and 3,500 to 4,500 orders per day is used to train and evaluate the approach. They simulate 25 days and divide each day into 510 time steps. The number of orders fulfilled per step is used as the reward function. Their results are compared with rule-based charging strategies that are parameterized by a working and charging threshold that determine when the AMRs have to charge and when they have to return to the working area.

In [26] an RL is applied to the dispatching of heterogeneous AMR in an online setting. A Noisy Dueling Double Deep Q-Network which enhances exploration via NoisyNets is employed. Their approach is trained on instances containing up to 12 AMRs and 900 tasks. The approach is compared to the exact and matheuristic methods in [27].

From the reviewed RL approaches we identify a number of gaps. In addition to the missing publication of the simulation and possibilities to reproduce the results, neither of the proposed approaches evaluates different RL design spaces but present single action and reward spaces. The approaches also differ in terms of decision granularity. In [12] decisions are made at the level of individual AMRs, but only the charging duration is optimized. The decision of when to go charging is fixed. In [4, 16] both the number of AMRs to go charging and a target charging threshold are determined, but on an aggregated level without taking the states of the individual AMRs into account.

## 3 Benchmark Extensions and Baseline Evaluation

Meaningful RL experiments require a strong grasp of the experimentation framework and its underlying data. This section outlines the battery management extensions in SLAPStack and WEPAStacks, and compares the heuristic charging strategies used as RL baselines. To keep within the frame of this work, we defer the implementation details to the published repository [1] and elaborate only on the parts needed to understand the RL design discussion that follows.

### 3.1 Benchmark Framework Extension

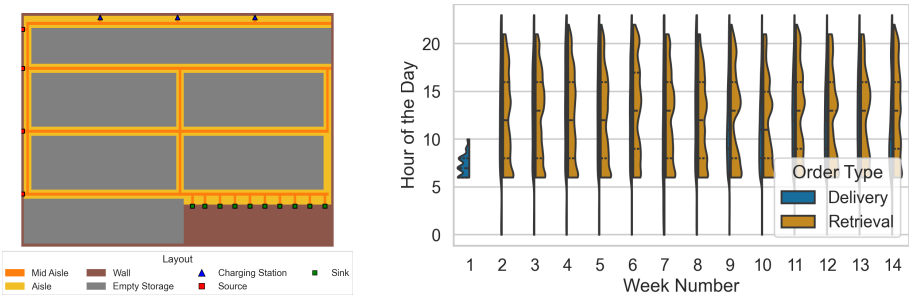
**Augmented SLAPStack**: SLAPStack is a discrete event simulation implementing an intuitive event chain concept detailed in its public GitHub repository [21]. Event chains model the process logic of the block storage and can pause the simulation execution to request inputs from control algorithms. The delivery event chain, for instance, is implemented as follows: When a delivery order arrives, an AMR is sent to the dock to pick up the pallet. When it arrives at the dock, the simulation requests a decision from a SLAPStrategy. The vehicle then moves the pallet to the indicated position, where it is released upon arriving.

Following this pattern, we implemented a charging event chain that requires a decision after every completed transportation task. An external decision entity determines how long an AMR should charge. A value of 0 indicates no charging, while any other value up to 100 specifies both the decision to charge and the desired duration. The vehicle then goes to the charging station and remains there for a corresponding time period, after which the vehicle is released. On every AMR movement, its battery is depleted based on movement time and load. The capacity replenishment occurs on vehicle release at a charging station.

We assume vehicles that are capable of automated plug-in or inductive charging. Following the assumptions of [7] the battery capacity of the vehicles is set to 52 ampere hours (Ah). We also use the same consumption rates of $travel_{\text{loaded}} = 15Ah$ and $travel_{\text{unloaded}} = 10Ah$. Further, a charging duration of half an hour to recharge to 100% is assumed. For recharging the battery, we assume a linear charging behavior as no non-linear movement model was available for this work. Battery degradation effects are not considered in this study.

**Augmented WEPAStacks**: Each SLAPStack use-case is defined by three components: warehouse layout, order set and an initial fill level. The initial fill level maps different stock keeping units (SKU) to the volume present in the warehouse. The order stream defines the times at which pallets with associated SKUs need to be stored or retrieved from the warehouse along with their in-/output point (dock). The layout defines the spatial dimension of all warehouse elements, i.e. storage, aisles and docks. Battery management augments this setup by specifying charging station locations.

WEPAStacks models a finished goods warehouse located at the WEPA GbmH hygiene paper company production site. Figure 2 visualizes the augmented layout and the order stream of the dataset we employ. Three charging stations are placed equidistantly along the north warehouse wall (Figure 2a). The warehouse




(a) Warehouse layout.

(b) Order arrival distributions per week.

Fig. 2: WEPAStacks use-case data visualization.

is 150 by 80 meters, and has 4 input and 10 output docks. The order stream contains 400,000 in- and outbound orders spanning 89 days. Figure 2b displays the order distribution over hours aggregated on a weekly basis. Outbound trucks arrive from 06:00 am to 22:00 pm, while the production outputs are continuous.

To prevent truck and, in particular, production queues in WEPAStacks, the number of pending retrieval and delivery orders is limited to 330 and 240, respectively, as described in [18]. We determine the required number of charging stations using a workflow similar to the one used in [22], where the nr of AMRs is inferred. We first fix simple heuristic solutions for all battery-augmented ABSWP decisions. Then, starting from a single charging station, we run simulations while gradually increasing their numbers until all system constraints are satisfied.

We use the solvers for the ABSW subproblems as reported in [18]. For SLAP we employ the closest open pure lanes strategy. Here, SKUs are stored in the closest lane that contains only that SKU. If no such lane exists, the nearest open location is chosen. The ULSP is solved using a last in first out strategy, in which the SKU that arrived last at the warehouse is retrieved first. For vehicle dispatching, the nearest available vehicle is selected.

### 3.2 Baseline Strategies

**Definition**: We introduce three strategies to address the charging-decision and charging-duration problems.

- *Fixed-threshold*: A fixed-threshold strategy is parameterized by two thresholds: $TH_{lower}$, which determines the battery level at which an AMR has to go charge, and $TH_{upper}$, which specifies the target battery level. We set $TH_{lower}$ to 20 % and vary $TH_{upper}$ from 30 % to 100 % in steps of ten.
- *Opportunity*: The opportunity-charge definition from [5], which was explained in Section 2, is not applicable in our scenario since the charging stations are not placed near the inbound and outbound docks. In our context, opportunity charging refers to charging when stations are available, and no work is pending. $TH_{upper}$ is fixed to 100%.

– *HighLow*: In the high-low approach, $TH_{lower}$ is fixed while $TH_{upper}$ varies depending on the retrieval order queue length. If there are queued retrieval orders, we use $TH_{upper}$; otherwise the battery is fully re-charged. To determine a good value for $TH_{lower}$, we tune it using the same thresholds as in the *Fixed-threshold* strategy.

The authors of [3] highlight that non-adaptive charging strategies often result in too few AMRs being available to handle tasks. To alleviate this, we propose the following improvement:

– *Interrupt*: If retrieval orders are pending and no AMR is available, the interrupt strategy halts charging for any AMR whose battery level exceeds the upper threshold $TH_{interrupt}$. In this work, we set $TH_{interrupt}$ to 50 %

For all strategies, we fix the selection of the charging station: the nearest available charging station is prioritized; if none is free, the station with the shortest queue is chosen.

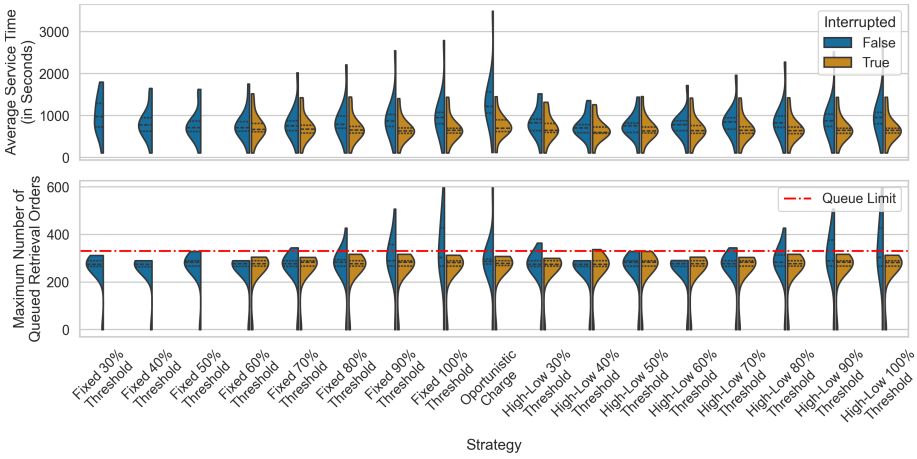



Fig. 3: Week and intercept dependent distribution of service times and buffer sizes.

**Evaluation**: In Figure 3, we show the results of the proposed baseline strategies for all weeks as violin plots. We compare the strategies in terms of average service time and maximum number of queued retrieval orders, both with and without the interrupt scheme. The figure suggests that charging strategies can help maintain AMR availability, either through short charging bursts, as *Fixed-threshold* with $TH_{upper} = 40\%$, or by interrupting the charging process. *Fixed-threshold* strategies with target thresholds of 80% to 100% violate queue constraints and result in long service times. This is in line with the results of [14], where the poor performance of full recharge strategies was highlighted. Implementing the interrupt scheme consistently improves all metrics, with the most significant improvements observed in high-threshold configurations. For example, applying

interruption to the *Fixed-threshold* strategy with $TH_{upper} = 100\%$ shows a 33% reduction in service time and 47% reduction in maximum queue size. Notably, the *HighLow* strategy with $TH_{upper} = 40\%$ achieves the best overall performance among non-interrupted strategies, with an average service time of 741 seconds and maximum queue size of 290 orders. This suggests that adapting charging durations based on the system load helps to maintain AMR availability. With interruption enabled, the benefits of the *HighLow* strategy are reduced. In this case, *HighLow* with $TH_{upper} = 90\%$ performs best, closely matching the corresponding *Fixed-threshold* strategy.

The *Opportunity* strategy on the other hand is not able to satisfy the constraints. In addition, it leads to the highest average service times. This indicates flaws in the design of the heuristic.

## 4 A Reinforcement Learning Charging Strategy

In this section, we outline the proposed approach to AMR charging using RL. We present the core components of the MDP, and motivate the use of PPO. Following the extension described in Section 3.1, SLAPStack now supports training and deploying RL agents for charging decisions in addition to SLAP, and ULSP. Since this work focuses on charging decisions, we adopt the best-performing heuristic solvers from [18] to handle SLAP and ULSP in the background. The simulation only pauses to request a charging decision from the RL agent. A charging decision is triggered after any completed transport task.

**State Representation and Feature Design**: Using the full system state of the ABSWP — encompassing all vehicle movements and storage locations — as the agent input would be computationally prohibitive. In line with [16] and [4], we adopt a feature-based state representation.

This includes AMR-related and operational features, such as battery level, distance to the charging stations, and the number of pending charging events and orders. We further incorporate ABSWP-specific features, most notably the average lane-wise entropy introduced in [18], which takes a value in $[0, \infty)$, with lower values indicating more ordered lanes.

As shown in [20], normalizing the observation space is essential for on-policy deep actor-critic algorithms such as PPO. We therefore scale all features to the $[0, 1]$ range. Table 2 provides the formal definitions of these features using the notation in Table 1.

**Reward and Action Space Design**: We propose several reward and action space configurations that progressively transition from a broad, flexible design to a more structured one. The underlying intuition is that a wider action space and sparse rewards offer more room for the agent to discover novel strategies, while a narrower, reward-engineered approach — augmented by domain knowledge — may lead to faster convergence, but could limit the discovery of high-performing strategies.

We vary configurations in terms of the reward function definition, action space, and use of the interrupt heuristic. We utilize a discrete action space $A$. At

Table 1: Feature space variables.

| Symbol | Description |
|---|---|
| $\mathcal{V}, \mathcal{C}$ | Set of AMRs ($\mathcal{V}$) and charging station ($\mathcal{C}$) indices |
| $\mathcal{L}, \mathcal{O}$ | Set of lane ($\mathcal{L}$), and finished order ($\mathcal{O}$) indices |
| $B$ | Maximum battery capacity |
| $B_n^{\text{cs}}$ | Battery level of AMR at charging station $n$ |
| $V_j^{\text{bl}}$ | Battery level of AMR $j$ |
| $V_j^{\text{depleted}}$ | 1 if AMR $j$ is depleted; 0 otherwise |
| $V_j^{\text{busy}}$ | 1 if AMR $j$ is busy; 0 otherwise |
| $V_j^{\text{free}}$ | 1 if AMR $j$ is free; 0 otherwise |
| $q_n$ | Queue length at charging station $n$ |
| $q_r, q_d$ | Queue lengths of retrieval and delivery orders |
| $q_r^{\max}, q_d^{\max}$ | Maximum queue length for retrieval and delivery orders |
| $T^{\text{ol}}, T^{\text{nl}}$ | Number of occupied and total storage locations |
| $h, d$ | Current hour and current day |
| $TR_i, DR_i$ | Travel time ($TR_i$), and distance ($DR_i$) for retrieval order $i$ |
| $TD_i, DD_i$ | Travel time ($TD_i$), and distance ($DD_i$) for delivery order $i$ |
| $L_k$ | Warehouse Lane $k$ |
| $p_{\text{SKU}}$ | Relative SKU amount in a Lane |

each time step $t$, given agent state $S_t$ the agent can select a non-zero charging time threshold, or decide not to charge by selecting the 0 action. This allows us to address both charging decision and charging duration sub-problems. The number of target thresholds is a design choice. In this work, we employ two alternatives. $A_{full} := [0, 30, 40, 50, 60, 70, 80, 90, 100]$ contains all the thresholds used in our baseline evaluation. $A_{binary} := [0, 100]$ is a binary action space, triggering full charge or no charge at all.

Using discrete charging thresholds instead of continuous durations, we enable action masking, which prevents agents from taking illegal actions. Action masking applies a binary mask over the action space at time step $t$ to mask out invalid actions by setting their sample probability to negative infinity [8]. Action masking is crucial for our application to prevent infeasible battery states, e.g. a target battery level lower than the current one. We apply the following action mask to determine feasible actions for an AMR at time step $t$: Action $A_i \in A$ is feasible if $A_i > V_t^{bl}$, where $V_t^{bl}$ denotes the battery level of the AMR being considered at time step $t$. This allows us to tackle both charging decisions and charging duration sub-problems. We use 20% as the final battery level where recharging is required to prevent AMRs from stranding during operation. Therefore, if $V_t^{\text{bl}} <= 20$, action 0 becomes invalid and charging becomes mandatory.

To the best of our knowledge, there is no universal approach for constructing a reward function. The design of an effective reward function involves several challenges, such as managing competing objectives. The so-called credit assignment problem [28] is another major challenge: Due to the delayed nature of rewards, it is difficult to identify the actions that led to a particular outcome. For these reasons, finding a suitable reward is often a trial-and-error process [2]. To overcome the credit assignment problem, reward shaping is often used, where domain and expert knowledge are incorporated to guide the agent's learning process, thus providing more direct feedback [28].

In the context of the ABSWP, our main objective is to minimize the average service time $ST_{avg}$, defined as the sum of all service times divided by the number

Table 2: Features used during RL-training.

| $\mathbf{F_i}$ | **Feature Name** | **Definition** |
|---|---|---|
| 1 | Mean battery level for all AMRs | $\frac{1}{\lvert\mathcal{V}\rvert}\sum_{j\in\mathcal{V}}\frac{V_j^{\mathrm{bl}}}{B}$ |
| 2 | Mean battery level for all busy AMRs | $\frac{1}{\sum_{j\in\mathcal{V}}V_j^{\mathrm{busy}}}\sum_{j\in\mathcal{V}}\frac{V_j^{\mathrm{bl}}\cdot V_j^{\mathrm{busy}}}{B}$ |
| 3 | Battery level of charging AMRs | $B_n^{\mathrm{cs}}, n\in\mathcal{C}$ |
| 4 | Battery level of AMRs | $\frac{1}{\lvert\mathcal{V}\rvert}V_j^{\mathrm{bl}}, j\in\mathcal{V}$ |
| 5, 6, 7 | Number of currently depleted/free/busy AMRs | $\frac{1}{\lvert\mathcal{V}\rvert}\sum_{j\in\mathcal{V}}V_j^{\mathrm{depleted\mid free\mid busy}}$ |
| 8 | Overall fleet utilization | $\frac{1}{\lvert\mathcal{V}\rvert}\sum_{j\in\mathcal{V}}V_j^{\mathrm{busy}}$ |
| 9 | Warehouse fill level | $1-\frac{T^{\mathrm{ol}}}{T^{\mathrm{nl}}}$ |
| 10 | Number of queued AMRs at charging stations | $q_n, n\in\mathcal{C}$ |
| 11 | Number of queued retrieval, and delivery orders | $\frac{q_r}{q_r^{\max}}$, $\frac{q_d}{q_d^{\max}}$ |
| 12 | Hour (sinusoidal encoding) | $\sin(2\pi\cdot h\cdot 24^{-1})$, $\cos(2\pi\cdot h\cdot 24^{-1})$ |
| 13 | Day of the week | $d\in\{1,\ldots,7\}$ |
| 14 | Number of currently free charging stations | $\lvert\{n\in\mathcal{C}\mid q_n=0\}\rvert$ |
| 15 | Average lane-wise entropy | $-\frac{1}{\lvert\mathcal{L}\rvert}\sum_{k\in\mathcal{L}}\sum_{\mathrm{SKU}\in L_k}p_{\mathrm{SKU}}\log(p_{\mathrm{SKU}})$ |
| 16, 17 | Average travel time retrieval \| delivery | $\frac{1}{\lvert\mathcal{O}\rvert}\sum_{i\in\mathcal{O}}TR_i\lvert TD_i$ |
| 18, 19 | Average distance retrieval \| delivery | $\frac{1}{\lvert\mathcal{O}\rvert}\sum_{i\in\mathcal{O}}DR_i\lvert DD_i$ |

of completed orders. The service time $ST_i$ for an order $i\in\mathcal{O}$ is the difference between its completion time and arrival time. In addition, we want to incentivize the agent to stay within the order queue limits. To achieve these goals, we propose four reward functions that progressively incorporate higher levels of guidance:

- **Service time-based reward:** The first reward function directly penalizes the agent based on the negative average service time observed at the current decision step $-ST_{avg}$, thus aiming to directly optimize for the main objective.
- **Queue-based reward:** The second reward function is $-q_r+q_d$, representing the negative sum of all queued retrieval and delivery orders in the current step. Although it does not explicitly optimize service time, our baseline analysis shows that high queue levels can lead to increased service times.
- **Composite reward + free AMR component:** This reward function combines the previous two components, while introducing an additional incentive to keep AMRs available during busy system states: when orders are pending, the agent receives a bonus for having free AMRs available, as this increases flexibility in fulfilling orders.
- **Shaped reward:** This reward extends the queue-based approach and introduces explicit penalties and incentives for charging decisions. The agent is penalized for initiating charging when no stations are available and for charging actions that result in situations where no AMRs remain free while there are queued orders. Conversely, rewards are given for charging actions taken when there are no queued orders and when a free charging station is available.

Where possible, we aim to normalize the reward components to the range $[0,1]$. To that end, we divide the service time and queued retrieval orders by de-

fined upper limits. The number of free AMRs is normalized by the total number of AMRs.

The final two configuration aspects we investigate pertain to the use of the previously introduced *Interrupt* heuristic. We consider both training and evaluation with and without *Interrupt* enabled. We compare the four different configurations shown in Table 3. The first three configurations differ in their reward formulations. *Basic1* and *Basic2* use the service-time and queue-based reward, respectively, incorporating minimal domain knowledge. *LightShaped* uses the composite reward, including the free AMR component. All three configurations use $A_{full}$ and exclude *Interrupt* during training. The fourth configuration, *FullyShaped*, combines the shaped reward with the reduced action space $A_{binary}$. Additionally *Interrupt* is enabled during training. During evaluation, we assess configurations both with and without *Interrupt*.

Table 3: Overview of the different RL-configurations.

| | **Reward** | **Action Space** | **Interrupted Training** | **Interrupted Evaluation** |
|---|---|---|---|---|
| *Basic1* | Service time-based | $A_{full}$ | False | True, False |
| *Basic2* | Queue-based | $A_{full}$ | False | True, False |
| *LightShaped* | Composite reward + Free AMRs | $A_{full}$ | False | True, False |
| *FullyShaped* | Shaped reward | $A_{binary}$ | True | True, False |

**Model**: We use PPO, an on-policy deep actor-critic RL algorithm [24], primarily chosen for its stable action masking implementation in [19]. Another advantage of PPO is its use of a clipped objective function, which constrains policy updates and mitigates the instability typical of policy gradient methods, thereby promoting stable and reliable training.

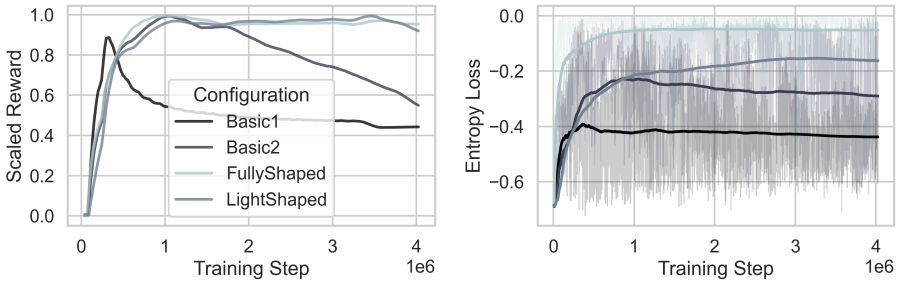
## 5 Experiments

In this section, we first present our setup, followed by an analysis of the training outcomes for the proposed configurations. This is followed by a comprehensive evaluation of the best models in terms of generalization capabilities and operational performance metrics.

**Setup**: All experiments were carried out on an Intel(R) Xeon(R) w5-2445 CPU. We use the open-source implementation of the masked PPO algorithm from [19] applying the default parameters. A rollout buffer size of 2048 and a minibatch size of 64 were used. Both the policy and value networks use a multi-layer perceptron, each consisting of two layers with 64 neurons. The order dataset, containing 400,000 orders, was divided on a weekly basis.

Four randomly sampled weeks were used for training. Each training episode terminates once all orders within the selected week have been simulated. The sampling resulted in weeks 3, 4, 6 and 13 being selected, containing 34,610, 37,134, 36,668, and 18,833 orders, respectively. Training was limited to 4 million

steps, with periodic evaluation every 200,000 steps to retain the best performing model. This periodic evaluation was carried out over all training data sets and averaged. The same simulation setting as in the baseline comparison was used, involving 3 charging stations and 40 AMRs.




(a) Mean episode reward. (b) Entropy loss.

Fig. 4: Learning metric evolution during training for the proposed configurations.

**Training**: The average episode reward for the different configurations is displayed in Figure 4a. For a better visualization, we normalized the rewards to the range $[0, 1]$. As described by Schulman et al. in [23], the policy entropy can be used to encourage exploration during training by incorporating an entropy bonus to the loss function. This additional loss is the negative mean entropy of the policy's action distribution across a batch of observations and is visualized for the different configurations in Figure 4b. Note that our training runs disregard the entropy term during policy update as we set the entropy coefficient to zero. Nevertheless, the value is still useful as an indicator of stochasticity and exploratory behavior of the policy during training.

In terms of reward the *Basic1* configuration initially performs well but declines after around 200,000 steps and never recovers. The reward curve for *Basic2* improves over a longer duration begins to decline after one million steps. *LightShaped* maintains strong performance for more than three million steps, with a decline observed after approximately 3.5 million steps. The *FullyShaped* configuration yields the most stable reward curve, with an indication of convergence toward the end of training. The entropy loss curves provide further insight into these dynamics. *Basic1* and *Basic2* exhibit poor convergence behavior, with persistently unstable entropy indicating ongoing exploration and a failure to form stable policies. This is consistent with their subpar reward performance. This suggests that the service-time- and queue-based rewards are not suited to effectively guide the agent toward reliable long-term strategies. *FullyShaped*, by contrast, converges rapidly to a low-entropy policy, with the entropy loss approaching and remaining near zero. This indicates limited exploration. *LightShaped* strikes a balance: entropy remains relatively high, supporting continued exploration, while gradually decreasing, reflecting policy stabilization. This controlled exploration aligns with its strong and sustained reward performance.

While *Basic1* and *Basic2* struggle with instability and performance degradation, the configurations with shaped reward functions *LightShaped* and *FullyShaped* demonstrate more robust learning dynamics. This suggests that incorporating more structured feedback into the reward function and reducing the action space enhances the agent's learning stability but may limit the exploration of the action space and encourage early deterministic policies, as can be seen from the entropy loss curve of *FullyShaped*.

**Evaluation**: To assess the model generalization capabilities, we report the performance of the best models found during training, averaged over all non-training weeks (Table 4). We compare the average service time, the maximum and average number of queued retrieval orders, the average battery level, and the average distance traveled per AMR. The results are grouped based on the use of the *Interrupt* heuristic, and we further include the top-performing heuristics from the baselines in Section 3.

Table 4: Comparison of the best RL-models found during training. Results are grouped by use of *Interrupt* and sorted by **Avg. Service Time (s)**. Best values are highlighted.

| Strategy | Avg Service Time (s) | Max Retrieval Queue | Mean Retrieval Queue | Mean Battery Level | Mean Travel Distance (km/AMR) |
|---|---|---|---|---|---|
| **Interrupted False** | | | | | |
| *LightShaped* | 588.84 | **290** | 19.94 | 61.99 | 139.77 |
| *Basic2* | 675.17 | **290** | 22.06 | 66.78 | 137.35 |
| *HighLow 40% Th* | 702.12 | **290** | 22.66 | 52.34 | 131.56 |
| *Fixed 60% Th* | 714.87 | **290** | 22.20 | 45.42 | 131.66 |
| *FullyShaped* | 719.70 | 346 | 24.10 | 79.79 | 145.38 |
| *Opportunity* | 1223.09 | 367 | 37.64 | 78.28 | 142.24 |
| *Basic1* | 4897.74 | 950 | 149.99 | 57.62 | 163.82 |
| **Interrupted True** | | | | | |
| *FullyShaped* | **581.85** | **290** | 19.73 | 64.27 | 144.83 |
| *LightShaped* | 582.33 | **290** | **19.52** | 56.14 | 139.79 |
| *Basic2* | 601.67 | **290** | 20.23 | 60.74 | 137.97 |
| *Fixed 90% Th* | 628.33 | 296 | 20.84 | 49.42 | 131.37 |
| *HighLow 90% Th* | 685.00 | 317 | 22.39 | 48.87 | 136.01 |
| *Opportunity* | 709.55 | **290** | 23.85 | 64.11 | 144.42 |
| *Basic1* | 5027.13 | 935 | 156.72 | 51.97 | 148.87 |

Table 4 illustrates how reward structure, action space design, and the *Interrupt* heuristic influence AMR charging performance. We can note that with and without the use of *Interrupt* we can find RL-based charging strategies that outperform the best baseline in terms of average service times and number of queued retrieval orders. Compared to *HighLow*, RL strategies show increased average travel distance per AMR, likely due to shorter charging cycles that necessitate more frequent trips to charging stations. The *FullyShaped* configuration, which combines a shaped reward with a reduced action space and domain knowledge, achieves the lowest average service time (581.85s) when interruption is enabled. However, *LightShaped* performs best in the non-interrupted case,

achieving an average service time of 588.84s. Furthermore, when interruption is enabled, *LightShaped* results in lower average retrieval queues (19.52) compared to *FullyShaped* (19.73). Both *Basic1* and *Basic2*, which use service-time and queue-based rewards, perform notably worse. Most notably, *Basic1*, which directly optimizes for service time, performs worst overall, with average service times of 5027.13s (with interruption) and 4897.74s (without). *Basic2* that only indirectly affects service times performs significantly better with average service times of 675.17s when not interrupted. Both *Basic2* and *LightShaped* can be considerably enhanced with *Interrupt*, reducing both service times and queued retrieval orders. The *FullyShaped* model, trained with interruption enabled, performs significantly worse when evaluated without it — service times rise to 719.70s, and the queued retrieval orders peak at 346. This suggests that *FullyShaped* fails to generalize as well as the more adaptable *LightShaped* model.

## 6 Conclusion

In this work, we extended an open-source simulation framework to support experiments on different AMR charging strategies in an ABSW setting. We also presented an MDP to *jointly* handles both the decision to charge and the duration of charging. Finally, we evaluated different MDP configurations and showed that RL-based charging strategies can outperform common heuristics. Our study highlights that the choice of reward function, action space, and use of heuristics can have a significant impact on the learning stability and performance of RL agents. Using a reward function without domain knowledge, such as service times, in combination with a broad action space yielded the worst results. The best performance was achieved using a shaped reward, a reduced action space, and a simple heuristic rule to interrupt charging. The inability of the model to perform well without this mechanism underscores the dangers of overfitting to specific problem settings. The most balanced setup used a multi-objective reward function with domain knowledge.

Given our findings, future research should explore more systematic approaches to reward shaping and validation. To improve realism, future work should incorporate more detailed battery models that account for degradation and non-linear charging behavior. Additionally, further work is needed to better isolate scenarios where battery management is the primary performance bottleneck, as opposed to other system aspects such as layout design or order sequence. A more thorough and systematic evaluation of algorithms and parameters, e.g. learning rate, is also essential to fully understand the potential of the promising RL designs proposed in this work.

**Acknowledgments.** This research was funded by the European Union - NextGenerationEU - funding code 13IK032I

## References


1. Bischoff, J., Rinciog, A., Pfrommer, J., Anne, M.: Slapstackcharged: An rl-compatible battery management extention for autonomous block stacking warehouse problems. https://github.com/j4n1k/slapstack-battery-management (2025)
2. Booth, S., Knox, W.B., Shah, J., Niekum, S., Stone, P., Allievi, A.: The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications. Proceedings of the AAAI Conference on Artificial Intelligence **37**(5), 5920–5929 (Jun 2023), http://dx.doi.org/10.1609/aaai.v37i5.25733
3. Colling, D., Oehler, J., Furmans, K.: Battery charging strategies for agv systems. Volume 2019 p. Issue 12 (2019), https://www.logistics-journal.de/proceedings/2019/4985
4. Deng, Y., An, B., Qiu, Z., Li, L., Wang, Y., Xu, Y.: Battery Management for Automated Warehouses via Deep Reinforcement Learning, p. 126–139. Springer International Publishing (2020), http://dx.doi.org/10.1007/978-3-030-64096-5_9
5. Ebben, M.: Logistic control in automated transportation networks. Ph.D. thesis, University of Twente, Netherlands (Jun 2001)
6. Han, W., Xu, J., Sun, Z., Liu, B., Zhang, K., Zhang, Z., Mei, X.: Digital twin-based automated guided vehicle scheduling: A solution for its charging problems. Applied Sciences **12**(7), 3354 (Mar 2022), http://dx.doi.org/10.3390/app12073354
7. Haney, R.: Modelling battery constraints in discrete event automated guided vehicle simulations. International Journal of Production Research **33**(11), 3023–3040 (Nov 1995), https://doi.org/10.1080/00207549508904859
8. Huang, S., Ontañón, S.: A closer look at invalid action masking in policy gradient algorithms. The International FLAIRS Conference Proceedings **35** (May 2022), http://dx.doi.org/10.32473/flairs.v35i.130584
9. Kabir, Q.S., Suzuki, Y.: Increasing manufacturing flexibility through battery management of automated guided vehicles. Computers & Industrial Engineering **117**, 225–236 (Mar 2018), http://dx.doi.org/10.1016/j.cie.2018.01.026
10. de Koster, R., Le-Duc, T., Roodbergen, K.J.: Design and control of warehouse order picking: A literature review. European Journal of Operational Research **182**(2), 481–501 (Oct 2007), https://doi.org/10.1016/j.ejor.2006.07.009
11. Kuhnle, A., Schäfer, L., Stricker, N., Lanza, G.: Design, implementation and evaluation of reinforcement learning for an adaptive order dispatching in job shop manufacturing systems. Procedia CIRP **81**, 234–239 (2019), http://dx.doi.org/10.1016/j.procir.2019.03.041
12. Lin, C.C., Chen, K.Y., Hsieh, L.T.: Real-time charging scheduling of automated guided vehicles in cyber-physical smart factories using feature-based reinforcement learning. IEEE Transactions on Intelligent Transportation Systems **24**(4), 4016–4026 (Apr 2023), http://dx.doi.org/10.1109/TITS.2023.3234010
13. Mazyavkina, N., Sviridov, S., Ivanov, S., Burnaev, E.: Reinforcement learning for combinatorial optimization: A survey. Computers & Operations Research **134**, 105400 (2021)
14. Meyer, A., Gschwind, T., Amberg, B., Colling, D.: Exact algorithms for routing electric autonomous mobile robots in intralogistics. European Journal of Operational Research (Dec 2024), http://dx.doi.org/10.1016/j.ejor.2024.12.041
15. Mousavi, M., Yap, H.J., Musa, S.N., Tahriri, F., Md Dawal, S.Z.: Multi-objective agv scheduling in an fms using a hybrid of genetic algorithm and particle swarm optimization. PLOS ONE **12**(3), e0169817 (Mar 2017), http://dx.doi.org/10.1371/journal.pone.0169817

16. Mu, Y., Li, Y., Lin, K., Deng, K., Liu, Q.: Battery management for warehouse robots via average-reward reinforcement learning. In: 2022 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE (Dec 2022), http://dx.doi.org/10.1109/ROBIO55434.2022.10011784
17. Pfrommer, J., Meyer, A.: Autonomously organized block stacking warehouses: A review of decision problems and major challenges. Logistics Journal : Proceedings **2020**(12) (Dec 2020), https://www.logistics-journal.de/proceedings/2020/5158
18. Pfrommer, J., Rinciog, A., Zahid, S., Morrissey, M., Meyer, A.: SLAPStack: A Simulation Framework and a Large-Scale Benchmark Use Case for Autonomous Block Stacking Warehouses, p. 291–305. Springer International Publishing (2022), http://dx.doi.org/10.1007/978-3-031-16579-5_20
19. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021), http://jmlr.org/papers/v22/20-1364.html
20. Raichuk, A., Stańczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S.: What matters for on-policy deep actor-critic methods? a large-scale study. In: International Conference on Learning Representations (2021), https://api.semanticscholar.org/CorpusID:233340556
21. Rinciog, A., Pfrommer, J., Morrissey, M., Sohaib, Z., Vasileva, A., Ogorelysheva, N., Rathod, H., Meyer, A.: Slapstack. https://github.com/malerinc/slapstack.git (2023)
22. Rinciog, A., Pfrommer, J., Rathod, H., Meyer, A., Ogorelysheva, N., Vasileva, A.: Crossstacks: A dataset and a simulative study of storage allocation strategies for cross-docking block-stacking warehouses. In: 2023 Winter Simulation Conference (WSC). IEEE (Dec 2023), http://dx.doi.org/10.1109/WSC60868.2023.10408362
23. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. CoRR **abs/1707.06347** (2017), http://arxiv.org/abs/1707.06347
24. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017), https://arxiv.org/abs/1707.06347
25. Shaukat, H.R., Shaukat, M.A.: Computation of number of charges required for automated guided vehicles with multiple time constraints. In: 2022 IEEE 8th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA). pp. 64–69 (2022)
26. Singh, N., Akcay, A., Dang, Q.V., Martagan, T., Adan, I.: Dispatching agvs with battery constraints using deep reinforcement learning. Computers & Industrial Engineering **187**, 109678 (Jan 2024), http://dx.doi.org/10.1016/j.cie.2023.109678
27. Singh, N., Dang, Q.V., Akcay, A., Adan, I., Martagan, T.: A matheuristic for agv scheduling with battery constraints. European Journal of Operational Research **298**(3), 855–873 (May 2022), http://dx.doi.org/10.1016/j.ejor.2021.08.008
28. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press, second edn. (2018), http://incompleteideas.net/book/the-book-2nd.html
29. Vis, I.F.: Survey of research in the design and control of automated guided vehicle systems. European Journal of Operational Research **170**(3), 677–709 (May 2006), http://dx.doi.org/10.1016/j.ejor.2004.09.020
30. Wang, Y.H., Zhang, H.M., Peh, L.S.: Research on power consumption of lithium ion battery protection circuit. Advanced Materials Research **1049–1050**, 582–585 (Oct 2014), http://dx.doi.org/10.4028/www.scientific.net/AMR.1049-1050.582
31. Zhan, X., Xu, L., Zhang, J., Li, A.: Study on agvs battery charging strategy for improving utilization. Procedia CIRP **81**, 558–563 (2019), http://dx.doi.org/10.1016/j.procir.2019.03.155