FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Survey of Viability of AI-based Task Scheduling for Robotic Fleets using a Predictive Cost Function

Miguel Faria Marmelo



Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. Pedro Costa

October 08, 2026

© Miguel Faria Marmelo, 2026

Survey of Viability of AI-based Task Scheduling for Robotic Fleets using a Predictive Cost Function

Miguel Faria Marmelo

Mestrado em Engenharia Eletrotécnica e de Computadores

Approved in oral examination by the committee:

President: Prof. Maria do Rosário Marques Fernandes Teixeira de Pinho
Referee: Prof. João Carlos Sampaio Maldonado Costa Mendes
Referee: Prof. Pedro Luís Cerqueira Gomes da Costa

October 08, 2026

Resumo

Com a crescente adoção de robots móveis autónomos (AMRs) na indústria 4.0, surge a necessidade de estratégias de coordenação altamente eficientes. Os métodos tradicionais de alocação de tarefas em sistemas multi-robot (MRTA) recorrem frequentemente a heurísticas centradas no curto prazo que privilegiam resultados imediatos, negligenciando a eficiência energética a longo prazo e a degradação da bateria. O agendamento de tarefas para sistemas multi-robot em ambientes industriais dinâmicos exige a gestão simultânea da coordenação entre robots, das restrições de bateria e da prioridade das tarefas. As abordagens existentes tratam tipicamente estas preocupações de forma independente ou tratam as restrições de bateria como uma penalidade suave dentro do sinal de recompensa, em vez de as tratarem como uma garantia estrutural. Esta dissertação propõe uma nova arquitetura de agendamento de tarefas baseada em *reinforcement learning* (RL), integrando uma função de custo concebida para otimizar simultaneamente o tempo de execução das tarefas e a longevidade operacional da frota. Nesta abordagem, a gestão de bateria é integrada de forma nativa na política aprendida, através de uma representação de estado unificada e de uma máscara de ações em camadas. O modelo é alargado para lidar com tarefas prioritárias, que devem ser resolvidas antes das tarefas normais, respeitando as restrições de viabilidade energética. Uma arquitetura encoder decoder baseada em atenção processa cada nó como um vetor de características composto por coordenadas e uma flag de prioridade, permitindo ao modelo aprender estratégias de agendamento coordenadas que equilibram a minimização do makespan, a segurança da bateria e a prioridade das tarefas. O treino é realizado on-policy através de instâncias simuladas, nas quais as tarefas chegam dinamicamente ao longo do tempo. Resultados experimentais ao longo de 1000 episódios mostram que a política de RL proposta iguala a taxa de conclusão de episódios (99,8% vs. 97,0%) do solver exato OR-Tools, e supera substancialmente uma baseline *greedy* de *nearest neighbour* (NN, 74,3%), ao mesmo tempo que requer aproximadamente $9\times$ menos tempo de computação acumulado por episódio.

Abstract

With the increasing adoption of autonomous mobile robots (AMRs) in industry 4.0, the need for highly efficient coordination strategies arises. Traditional multi-robot task allocation (MRTA) methods often rely on short term focused heuristics that prioritise immediate throughput, neglecting long term energy efficiency and battery degradation. Task scheduling for multi-robot systems in dynamic industrial environments requires simultaneous management of robot coordination, battery constraints and task priority. Existing approaches typically address these concerns independently or treat battery constraints as a soft penalty within the reward signal rather than as a structural guarantee. This dissertation proposes a novel reinforcement learning (RL) based task scheduling architecture, integrating a cost function designed to simultaneously optimise task execution time and fleet operational longevity. In this approach, battery management is integrated natively into the learned policy through a unified state representation and layered action masking. The model is extended to handle priority tasks, which must be addressed before non priority work while respecting energy feasibility constraints. An attention based encoder decoder architecture processes each node as a feature vector comprising coordinates and a priority flag, enabling the model to learn coordinated scheduling strategies that balance makespan minimisation, battery safety and task priority. Training is performed on-policy via simulated instances where tasks arrive dynamically over time. Experimental results over 1,000 dynamic episodes show that the proposed RL policy matches the episode completion rate (99.8% vs. 97.0%) of the exact OR-Tools solver, substantially outperforming a greedy NN baseline (74.3%), while requiring approximately $9\times$ less cumulative computation time per episode.

United Nations (UN) Sustainable Development Goals

This dissertation aligns with the UN agenda by improving the scheduling of AMR fleets. By integrating AI for the coordination of these fleets, including native battery safety guarantees, the proposed solution aims to make industrial logistics more autonomous and reliable, reducing reliance on computationally expensive exact solvers or short sighted heuristic dispatch rules for time critical operations.

SDG 7: Affordable and Clean Energy Ensure access to affordable, reliable, sustainable and modern energy for all. By enforcing battery safety as a structural guarantee rather than a soft constraint, this work supports the reliable, continuous operation of electric AMR fleets, a prerequisite for their wider adoption as a lower emission alternative to combustion powered industrial vehicles.

SDG 9: Industry, Innovation, and Infrastructure Build resilient infrastructure, promote inclusive and sustainable industrialisation and foster innovation. This project contributes a learned scheduling approach that matches the reliability of exact optimisation methods at a fraction of their computational cost, supporting more responsive and resilient industrial automation infrastructure.

SDG	Target	Contribution	Performance Indicators and Metrics
7	7.1	By enforcing a structural battery safety guarantee rather than relying on manual intervention or external safety modules, the proposed scheduler enables continuous, unattended operation of electric AMR fleets, supporting their reliability as a lower emission alternative to combustion powered logistics vehicles.	• Episode completion rate under battery constraints. • Battery failure rate (target: zero). • Cumulative decision latency relative to exact optimisation methods.
9	9.1	The implementation of RL for MRTA represents a technological upgrade to standard logistics infrastructure, delivering scheduling decisions at a fraction of the computational cost of exact optimisation, supporting more autonomous industrial operation.	• Cumulative decision latency relative to exact optimisation methods. • Episode completion rate relative to classical heuristic and exact baselines. • Average queueing time at shared charging infrastructure.

Acknowledgements

I would like to begin by thanking my supervisors, Prof. Pedro Costa and Eng. Catarina Rema, for their guidance, constant availability, sharp feedback and for the opportunity to pursue this work in collaboration with INESC TEC.

I would also like to thank the Faculty of Engineering of the University of Porto for the resources and support made available throughout my studies.

To my girlfriend, my family and my friends: thank you for staying close through all of it, thank you for never once making me doubt it and thank you for making this journey feel a little lighter. This work would not have been possible without you.

Miguel Marmelo

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Problem Statement . . . 2
- 1.3 Objectives . . . 3
- 1.4 Contributions . . . 3
- 1.5 Dissertation Structure . . . 4

2 State of the Art **5**
- 2.1 MRTA Taxonomy . . . 5
- 2.2 Classical and Heuristic Approaches . . . 6
  - 2.2.1 Heuristic Methods . . . 6
  - 2.2.2 Exact Optimisation Methods . . . 7
- 2.3 Metaheuristic and Hybrid Methods . . . 8
  - 2.3.1 Evolutionary and Swarm Intelligence Methods . . . 8
  - 2.3.2 Hybrid Methods . . . 9
- 2.4 RL and Attention based Methods . . . 9
  - 2.4.1 Standard RL for Robot Scheduling . . . 10
  - 2.4.2 Scalable and Distributed Architectures . . . 10
  - 2.4.3 Multi Agent and Hierarchical RL . . . 11
- 2.5 Battery and Energy Aware Fleet Scheduling . . . 12
  - 2.5.1 Energy and Consumption Modelling . . . 12
  - 2.5.2 Battery Degradation and Maintenance . . . 12
  - 2.5.3 Fleet Level Optimisation . . . 12
- 2.6 Positioning of This Work . . . 13

3 Problem Formulation and Proposed Method **16**
- 3.1 Problem Statement . . . 16
- 3.2 Operational Constraints . . . 17
- 3.3 Environment . . . 18
  - 3.3.1 Calibration of Battery Parameters . . . 18
  - 3.3.2 Episode and Task Parameters . . . 19
- 3.4 State Representation . . . 20
- 3.5 Attention based Architecture . . . 20
  - 3.5.1 Encoding the Graph . . . 20
  - 3.5.2 Deciding the Next Action . . . 22
- 3.6 Action Masking . . . 23
  - 3.6.1 Battery Safety . . . 24
  - 3.6.2 Priority Enforcement . . . 24

3.6.3 Premature-return prevention . . . 24
3.6.4 Fallback Rule . . . 25
3.7 Cost Function . . . 25
3.7.1 Timing Penalty . . . 25
3.7.2 Priority Penalty . . . 26

4 Implementation and Development Process **27**
4.1 Development and Training Infrastructure . . . 27
4.2 Correcting the Benchmark Evaluation Harness . . . 27
4.2.1 Stale Action Queues . . . 28
4.2.2 Unbounded Decoder Queries . . . 28
4.2.3 Missing Position Context . . . 28
4.2.4 Graph Size Overflow . . . 29
4.3 Priority Weighting in the Exact Solver . . . 29

5 Experimental Setup and Results **30**
5.1 Training . . . 30
5.2 Experimental Setup . . . 30
5.3 Results . . . 32

6 Discussion and Limitations **34**

7 Conclusion and Future Work **36**
7.1 Contributions . . . 36
7.2 Future Work . . . 37

References **38**

List of Figures

3.1 The simulated environment shortly after episode start, showing the two robots (r_1, r_2) departing their respective bases (b_1, b_2) towards their first assigned tasks, with further pending tasks distributed across the workspace. 19
3.2 Architecture of the proposed attention model for the MRTSP-B with priority tasks. 21

5.1 Qualitative comparison of the three evaluated methods (RL, NN, OR-Tools, left to right) on the same episode instance. 32

List of Tables

2.1 Positioning of all reviewed works against the requirements of this dissertation. Y = yes, N = no, P = partial or qualified support. 13

5.1 Key training parameters. 31
5.2 Benchmark results (1,000 episodes, 100 tasks each). 32

List of Acronyms

AI Artificial Intelligence

AGV Automated Guided Vehicle

ALNS Adaptive Large Neighbourhood Search

AMR Autonomous Mobile Robot

CP Constraint Programming

CTPN Coloured-Timed Petri Net

DL Deep Learning

DP Dynamic Programming

DQN Deep Q-Network

EA Evolutionary Algorithm

ED2PSO Enhanced Dynamic Distributed Particle Swarm Optimisation

FIFO First In, First Out

GA Genetic Algorithm

GNN Graph Neural Network

IoT Internet of Things

JSSP Job Shop Scheduling Problem

LMEO Learning-based Multipopulation Evolutionary Optimisation

MILP Mixed-Integer Linear Programming

MINLP Mixed-Integer Nonlinear Program

ML Machine Learning

MRIG Multi Restart Iterated Greedy

MRTA Multi-Robot Task Allocation

MRTSP-B Battery Constrained Multi-Robot Travelling Salesman Problem

NN Nearest Neighbour

NP-hard Non-deterministic Polynomial-time hard

PSO Particle Swarm Optimisation

RL Reinforcement Learning

ROBOT2026 9th Iberian Robotics Conference

ROS Robot Operating System

SMT Satisfiability Modulo Theories

SSA Sparrow Search Algorithm

ST-SR-TA Single task, Single robot and Time-extended Assignment

TCM Task allocation and charging manager

TS Tabu Search

TSP Travelling Salesman Problem

VRP Vehicle Routing Problem

Chapter 1

Introduction

The growing deployment of autonomous mobile robot (AMR) fleets in industrial settings has made battery aware, priority sensitive task scheduling an increasingly practical concern, not merely a theoretical one. This chapter motivates the problem addressed in this dissertation (Section 1.1), states it formally (Section 1.2), outlines the objectives pursued and the contributions made (Sections 1.3 and 1.4), and closes with an overview of how the remainder of this dissertation is organised (Section 1.5).

1.1 Context and Motivation

The adoption of AMRs in industrial and logistics environments has accelerated substantially under the Industry 4.0 paradigm, as manufacturers and warehouse operators seek to automate the repetitive, spatially distributed transport of materials between workstations, storage areas and assembly lines. Coordinating a fleet of such robots is fundamentally a task scheduling and allocation problem, in which a decision making process must determine, from a stream of work orders arriving over time, which robot executes which task and in what order, so that operations proceed efficiently and without interruption.

In practice, this coordination problem cannot be separated from the physical constraints of the robots themselves. In particular, AMRs operate on finite battery capacity and must periodically interrupt task execution to recharge at dedicated stations. A scheduling policy that ignores this constraint or treats it as an afterthought of an otherwise energy agnostic decision process risks uneven battery usage across the fleet, congestion at shared charging stations and, in the worst case, robots stranding mid route with depleted batteries. Several classical and metaheuristic approaches already account for battery levels or charging constraints when assigning tasks [1]. However, this integration as a structural, first class constraint within a learned policy remains largely absent from the reinforcement learning (RL) literature.

A second requirement that arises naturally in real deployments is the handling of priority tasks: work orders that are more urgent than others. Those tasks should be executed ahead of routine work whenever it is operationally feasible to do so. Combining priority aware dispatch

with battery safe operation compounds the difficulty of the scheduling problem, since urgency and remaining battery capacity can pull a scheduling decision in different directions, a tension that must be resolved without ever compromising safe operation.

Classical approaches to this problem range from simple greedy heuristics, which prioritise immediate throughput but disregard the long term consequences of their choices, to exact optimisation methods such as Mixed-Integer Linear Programming (MILP) and constraint programming (CP), which can guarantee optimal solutions for small, static instances but scale poorly as the number of tasks and robots grows or as tasks arrive dynamically during operation [1]. The scheduling problem underlying multi-robot task allocation (MRTA) maps onto variants of the Job Shop Scheduling Problem (JSSP), which is known to be non deterministic polynomial hard (NP hard) [2]. This theoretical barrier renders exact methods computationally impractical in a real-time, continuously evolving industrial setting, justifying the use of approximate methods like RL.

RL has emerged as a promising alternative paradigm for this class of combinatorial problems [3]. Attention based encoder decoder architectures, in particular, have demonstrated that a policy trained offline can produce near optimal routing decisions at inference time, at a small fraction of the computational cost of an exact solver [4, 5]. However, the majority of RL formulations for routing and scheduling problems assume robots with unlimited energy. Battery management, when considered at all, is typically implemented as a soft penalty within the reward function rather than a structural constraint the policy is guaranteed to respect, or is omitted from the formulation entirely [4, 5]. This decoupling between the component that learns to schedule and the component that guarantees energy feasibility introduces an inconsistency between what the policy is trained to optimise and what it is actually allowed to do at deployment time, a mismatch that motivates the central design choice explored in this dissertation: integrating battery awareness natively into the learned policy.

1.2 Problem Statement

The core problem addressed in this dissertation involves dynamically scheduling tasks for a small fleet of AMRs operating continuously, under three simultaneous requirements:

- Tasks arrive stochastically over time rather than being known in advance, requiring the scheduling policy to plan online as new information becomes available;
- Every scheduling decision must respect the battery capacity of the acting robot, guaranteeing that it can always return to a charging base;
- A subset of tasks are marked as priority and must be addressed ahead of routine work whenever they are reachable given the robot's current battery level.

1.3 Objectives

The primary objective is to develop and validate an RL based scheduler that utilises a predictive cost function to optimise fleet performance by assigning and scheduling tasks to these robots in a way that minimises total execution time (makespan) while respecting the battery constraints of the fleet at every decision. Standard optimisation solvers, such as MILP, can find optimal solutions but scale poorly as the fleet size increases. Conversely, standard heuristics fail to account for the long term cost of decisions, such as assigning a distant task to a robot with low battery. Therefore, the research question driving this work is: How can an RL agent be designed to dynamically schedule tasks for a multi-robot fleet, effectively balancing the trade-off between minimising makespan and maintaining safe battery operation?

To achieve this, the following specific tasks are defined:

1. Conduct a literature review of the state-of-the-art to identify gaps in the task scheduling for AMRs problem.
2. Design a unified state representation for a battery constrained variant of the multi-robot travelling salesman problem (MRTSP-B), in which robot positions, battery levels, charging queues and timing penalties are governed by a single state transition function shared between training and deployment.
3. Design and implement a layered action masking mechanism that enforces battery safety and priority preemption directly at the decision level, so that no action leading to battery failure or violating priority urgency can ever be selected by the policy.
4. Train an attention based encoder decoder policy to schedule tasks for a two robot fleet operating under the above constraints in a continuously evolving environment.
5. Benchmark the resulting policy against a greedy nearest neighbour (NN) heuristic and an exact solver (Google OR-Tools), under the same physical and safety conditions, across 1,000 complete episodes.
6. Critically evaluate the resulting system across episode completion rate, travel efficiency, computational cost relative to both baselines, and adherence to the safety and priority guarantees the action mask is designed to provide.

1.4 Contributions

Building on the gaps identified in the existing literature on mobile robot task scheduling, this dissertation makes the following contributions:

- Native battery management as a first class constraint: This work integrates an explicit battery safety mask directly into the decoder, so that battery failure is structurally impossible at inference time, regardless of training progress.

- A scheduling policy for dynamic, continuously evolving environments: Unlike attention based routing models trained on static instances where the full task set is known in advance, the proposed system is trained and evaluated on episodes where tasks arrive stochastically throughout operation, requiring the policy to plan online at every decision step.
- Joint battery and priority aware scheduling: The proposed masking mechanism combines a priority mask, which enforces preemption of routine tasks whenever an urgent task is reachable, with the battery safety mask, so that urgency is respected without ever compromising the guarantee of safe battery operation.
- An empirical evaluation across 1,000 dynamic episodes, comparing the proposed RL policy against a greedy heuristic and an exact Google OR-Tools solver under identical physical and safety rules, reported together with an account of the trade-offs observed between the three methods.
- Another major contribution of this work is a condensed version focused on the proposed method and final experimental results, which has been submitted as a conference paper to the 9th Iberian Robotics Conference (ROBOT 2026).

1.5 Dissertation Structure

The remainder of this dissertation is organised as follows.

Chapter 2 reviews the existing literature on multi-robot task scheduling, covering classical heuristics, metaheuristic and hybrid methods, RL approaches to routing and scheduling, and recent work specifically addressing battery aware fleet scheduling, before positioning the contributions of this dissertation relative to the gaps identified.

Chapter 3 formalises the scheduling problem addressed in this work, situating it within the broader MRTA taxonomy and describing the operational constraints that govern robot behaviour, including battery dynamics and priority handling. It then presents the proposed method in detail: the environment formulation as an instance of the MRTSP-B, the unified state representation, the layered action mask, the attention based encoder decoder architecture and the cost function used for training.

Chapter 4 documents the implementation and development process, including the computational infrastructure used, the iterative debugging of the training and evaluation pipelines, and a series of correctness issues discovered during this work whose resolution was necessary to obtain trustworthy results.

Chapter 5 describes the experimental setup, including the baseline methods used for comparison, and reports the results of the benchmark evaluation across 1,000 dynamic episodes.

Chapter 6 discusses the results in greater depth, examining the trade-offs observed between the evaluated methods and the methodological limitations of the present work.

Chapter 7 concludes the dissertation, summarising its contributions and outlining directions for future work.

Chapter 2

State of the Art

This chapter presents a literature review on task scheduling for fleets of mobile robots, with the aim of identifying the gaps that motivate the work developed in this dissertation.

2.1 MRTA Taxonomy

MRTA is the problem of determining which robot should execute which task so as to optimise a global objective function. The foundational taxonomy of Gerkey and Matarić [6] classifies MRTA problems along three axes: whether each robot can execute a single task at a time or multiple tasks concurrently, whether each task requires exactly one robot or several robots simultaneously and whether task assignment is instantaneous, allocating tasks greedily as they become known, or time-extended, in which a schedule of future tasks is optimised over a planning horizon. Following this taxonomy, the problem addressed in this dissertation is classified as single task, single robot and time-extended assignment (ST-SR-TA):

- Each robot executes one task at a time;
- Each task requires exactly one robot;
- The scheduling policy reasons about a horizon of future tasks rather than committing to the nearest available option as soon as it appears.

It is important to distinguish task scheduling, task allocation and task planning, three closely related but distinct problems that are frequently conflated in the literature. As clarified by Rema et al. [1], task allocation is concerned with deciding which robot performs which task, whereas task scheduling addresses the timing and sequencing of tasks once an allocation has been made, that is, the order in which a robot visits its assigned tasks and how its movements between them are managed. Task planning, in turn, typically refers to the low level motion or path planning required to execute an individual task once scheduled. This dissertation addresses task scheduling in this specific sense, while necessarily also performing task allocation implicitly, since the scheduling policy jointly decides which robot acts and what it does next at every planning step.

It is similarly important to distinguish mobile robot task scheduling from the classical Vehicle Routing Problem (VRP), despite their structural similarity. A VRP plans routes for a fleet of vehicles that deliver goods to a set of customers from a shared depot, visiting each customer exactly once, typically minimising total travel distance under vehicle capacity and, in some variants, time window constraints. Visit durations are usually negligible, and no precedence relations exist between customers [1, 7].

Mobile robot task scheduling, by contrast, involves tasks with non negligible duration and, frequently, precedence constraints, executed by robots that can perform only one task at a time without interruption once started [7, 8].

Critically for this dissertation, the primary objective in mobile robot task scheduling is to minimise the fleet's makespan (the time at which the last task is completed) rather than total distance travelled, and the problem must additionally account for shared and finite resources such as charging stations, which have no direct analogue in the classical VRP formulation.

2.2 Classical and Heuristic Approaches

Classical approaches to mobile robot task scheduling span a wide spectrum, from computationally inexpensive greedy rules to exact mathematical optimisation. Following the taxonomy of Rema et al. [1], the methods reviewed in this section are organised into two broad groups: heuristic methods, which provide fast approximate solutions, and exact optimisation methods, which provide provable guarantees at greater computational cost.

2.2.1 Heuristic Methods

Heuristic algorithms represent a comparatively small fraction of the literature, reflecting their limited ability to scale to complex, multi constraint scheduling problems, though they remain popular as computationally inexpensive baselines [1]. Typical heuristic strategies assign tasks to robots based on proximity or cost at the moment a robot becomes available, sometimes combined with mechanisms to re-bid or reorder tasks under dynamic conditions [9, 10]. NN assignment, for instance, minimises immediate travel time but is myopic, ignoring the long term cost of each decision.

To mitigate this myopia, more structured heuristic search procedures have been proposed. Liu et al. [11] proposed a Multi Restart Iterated Greedy (MRIG) algorithm for multi AGV (Automated Guided Vehicle) dispatching that repeatedly destroys and reconstructs parts of the schedule to escape local optima, combining an enhanced initialisation strategy with adaptive destruction phases and a simulated annealing acceptance criterion. Yi and Luo [12] model a robotic job shop as a place timed Petri Net and use an artificial potential field to guide an A* search for the minimum makespan firing sequence, exploiting the Petri Net's ability to capture precedence and resource sharing constraints explicitly. Bakshi et al. [13] formulate the single robot task ordering problem as a regularised quadratic programme, enabling gradient based real-time sequencing of prioritised tasks without the combinatorial cost of exhaustive search.

Notably, several heuristic approaches already incorporate battery as an explicit constraint at the point of assignment, for instance by restricting eligible workstations or tasks to those a robot can reach given its current charge [14], or by grouping tasks according to both battery power and time constraints [15].

2.2.2 Exact Optimisation Methods

Exact optimisation approaches, particularly MILP, dynamic programming (DP) and CP, offer the alternative extreme: rather than a fast, approximate rule, these methods formalise the scheduling problem mathematically and search for a provably optimal or bounded sub optimal solution.

Within mobile robot scheduling specifically, recent work has produced progressively tighter formulations. Sahar et al. [16] proposed a tightened MILP formulation for the JSSP with mobile robots that reduces the search space by strengthening the LP relaxation bounds. Yao et al. [17] developed a novel MILP based on a modified disjunctive graph model that integrates mobile robot transport explicitly into job shop scheduling. Justkowiak et al. [18] applied DP to the order scheduling and rack sequencing problem in robotic mobile fulfilment systems, minimising the number of rack visits. Their approach is the first combinatorial solution method for this problem and outperforms monolithic MILP solvers on strong formulations. Riazi and Lennartson [19] address the integrated scheduling and routing problem for AGV fleets using CP and Satisfiability Modulo Theories (SMT) solvers via Benders decomposition, providing a mathematical guarantee that any generated schedule is conflict free with respect to collision avoidance. Addressing the stochastic nature of real warehouse environments, Zhang et al. [20] propose a simulation budget allocation technique that concentrates available computation on the most promising candidate schedules rather than evaluating all options equally, integrating simulation optimisation with deterministic scheduling and routing algorithms.

Beyond mobile robot fleets, exact formulations have also been applied to related multi-robot settings. Lippi and Marino [21] formulate human MRTA as a MILP problem that jointly minimises makespan while incorporating precedence and spatial constraints alongside an online re-allocation mechanism. Behrens et al. [22] employ CP for simultaneous task allocation and motion scheduling in industrial dual arm manipulation, illustrating that exact formulations extend naturally to manipulation settings as well.

All of these exact approaches share the same fundamental limitation that motivates the use of approximate methods in this dissertation: the combinatorial nature of the underlying assignment and sequencing problem causes solve time to grow rapidly with the number of tasks and robots, rendering exact re-optimisation at every planning step impractical in a continuously evolving environment. Lippi and Marino [21] acknowledge this directly, deliberately limiting re-solve frequency to avoid what they describe as “excessive computational burden”. This limitation is precisely why an exact solver (Google OR-Tools [23]) is retained in this dissertation only as a benchmark baseline rather than as the primary scheduling method, as detailed in Chapter 5.

2.3 Metaheuristic and Hybrid Methods

Metaheuristic methods occupy a middle ground between heuristics and exact solvers, using structured search strategies to approach near optimal solutions within acceptable computation time without guaranteeing global optimality. The largest share of the literature reviewed by Rema et al. [1] employs metaheuristic methods, most commonly Genetic Algorithms (GA), Adaptive Large Neighbourhood Search (ALNS) and Particle Swarm Optimisation (PSO).

2.3.1 Evolutionary and Swarm Intelligence Methods

GAs have attracted particular attention because their population based search naturally combines global exploration with local refinement. Li et al. [24] propose a multi strategy GA for the flexible JSSP with AGVs, combining a two layer encoding scheme with multiple heuristic initialisations, double crossover and dual mutation operators to handle the three coupled subproblems of operation sequencing, machine selection and AGV selection simultaneously. Han et al. [25] introduce a dual population collaborative GA that maintains two sub populations with distinct decoding strategies, exchanging individuals through a collaboration operator to expand the search space and avoid premature convergence. Cechinel and De Pieri [26] apply an Island Model GA to multi-robot logistics task allocation, running parallel sub populations with periodic migration of solutions between islands to maintain genetic diversity while coordinating heterogeneous robots with different payload capacities and battery levels.

PSO based approaches have similarly been extended to account for multi-robot coordination. Alzubairi et al. [27] propose an Enhanced Dynamic Distributed PSO (ED^{2}PSO) for automatic task clustering in multi-robot systems, adding local optima detection operators for both personal and global bests to avoid premature stagnation and enable automatic determination of the number of task clusters without prior knowledge. Mousavi et al. [28] combine GA and PSO in a hybrid scheme for multi objective scheduling of AGVs that explicitly accounts for battery charge, using the GA to explore a broad solution space and PSO to refine and converge towards the optimal solution.

Beyond GA and PSO, newer swarm intelligence algorithms have been adapted to AGV scheduling. Cui et al. [29] apply a Discrete Jaya Algorithm to the multi AGV scheduling problem with dynamic unloading times, designing two operators without requiring any algorithm specific parameter tuning: a near optimal operator that moves solutions towards better candidates and an away worst operator that repels them from the worst. Sun et al. [30] propose a Chaotic Sparrow Search Algorithm (SSA) for flexible JSSP with AGVs, using Hénon chaotic mapping for population initialisation and adaptive parameter strategies to escape local optima more effectively than standard PSO.

2.3.2 Hybrid Methods

A comparable share of the literature employs hybrid methods that combine two or more paradigms, most frequently pairing metaheuristics with MILP or with Tabu Search (TS) to intensify the search around solutions of already known quality. Zheng et al. [31] exemplify this pairing directly, formulating a MILP model for simultaneous machine and AGV scheduling and then applying a TS heuristic to solve large scale instances where the MILP becomes computationally intractable. Dang et al. [32] apply ALNS specifically to mobile robot scheduling, iteratively destroying and repairing parts of a candidate solution to escape local optima while adapting the destroy and repair operators over the course of the search. Other hybrid schemes surveyed combine metaheuristics with discrete event simulation, using the simulation to estimate the performance of candidate schedules under uncertainty before a search procedure selects among them [1].

More recent hybrid approaches integrate learning components or exploit distributed computing architectures. Pan et al. [33] propose a learning based multipopulation evolutionary optimisation (LMEO) for the flexible JSSP with finite AGV transportation resources, in which an RL component guides mating selection across sub populations, selecting which individuals to combine at each generation, improving both diversity and convergence compared to standard evolutionary search. Lin et al. [34] design a hierarchical cloud edge control platform for large scale AGV systems, in which a parallelised scheduling algorithm runs as a cloud service to handle fleet level task assignment, while conflict free path planning executes at the edge to minimise communication latency and enable real-time collision avoidance. Xu et al. [35] address green scheduling by applying a multi objective evolutionary algorithm to the integrated flexible JSSP and AGV scheduling problem, explicitly trading off makespan against energy consumption and producing Pareto optimal schedules that allow operators to select operating points according to sustainability targets.

Across both metaheuristic and hybrid methods, two limitations recur that are directly relevant to this dissertation. First, the computational cost of these search procedures, while lower than exact optimisation, remains iterative and comparatively expensive, similarly limiting their suitability for frequent online replanning. Second, most of the methods discussed above are evaluated on problem instances where the complete task set is known in advance, rather than on the dynamically evolving, continuously operating environment this dissertation targets, in which tasks arrive stochastically throughout an episode.

2.4 RL and Attention based Methods

Machine learning (ML) methods, and RL in particular, represent a recent and comparatively small but growing thread of the reviewed literature [1]. RL approaches formulate scheduling as a sequential decision making problem, in which an agent learns a policy $\pi(a \mid s)$ through trial and error interaction with a simulated environment, rather than through explicit search or mathematical optimisation [3].

Foundational work in applying deep learning (DL) to combinatorial routing problems predates its application to multi-robot scheduling specifically. Bello et al. [36] first proposed the use of a pointer network to construct tours for the Travelling Salesman Problem (TSP), using RL to train a policy that outputs a probability distribution over unvisited cities at each step. Nazari et al. [5] generalised this paradigm to the VRP, removing the recurrent sequence encoder and introducing an attention mechanism in the decoder capable of handling dynamic elements, such as evolving customer demands, that change over the course of the episode. Kool et al. [4] subsequently introduced a fully attention based encoder decoder architecture, trained end to end with REINFORCE [37] and a rollout baseline, achieving solution quality competitive with classical heuristics at a fraction of the inference time cost of an exact solver. This attention based encoder decoder paradigm forms the architectural basis of the model proposed in this dissertation, detailed in Chapter 3.

2.4.1 Standard RL for Robot Scheduling

Several works have applied standard RL directly to robot scheduling problems. Durst et al. [38] apply deep RL to dynamic AGV scheduling in manufacturing lines, benchmarking the learned policy against first in, first out (FIFO) and other dispatching rules and showing that learning based dispatch produces more efficient routing. Li et al. [39] apply deep RL to the task selection problem in autonomous mobile robot warehousing, demonstrating that agents can learn to prioritise tasks based on spatial locality without explicit heuristic programming. Tejer et al. [40] focus on robustness, proposing a hybrid Q-learning approach for task scheduling in robotics applications that maintains performance under non deterministic environments where execution conditions fluctuate. Cheng et al. [41] introduce a deep RL framework for batch order scheduling in robotic mobile fulfilment systems, with a reward function that explicitly minimises operational cost rather than makespan alone. Shida et al. [42] address infeasible tasks directly, proposing a framework in which each robot learns exclusion levels for tasks whose feasibility is uncertain due to unknown object weights, enabling proactive rejection of untransportable tasks to avoid deadlock.

2.4.2 Scalable and Distributed Architectures

A consistent limitation of standard RL policies is their failure to generalise when fleet size changes at deployment time. Zhang et al. [43] address this via a graph deep RL framework that encodes robots and tasks as nodes in a graph, using graph sample and aggregate to extract relational features. Because the policy operates on graph structure rather than fixed size vectors, it generalises zero shot to fleet sizes unseen during training. Lai et al. [44] extend this to multi-robot reaching and task and motion planning in shared manufacturing workspaces, proposing a Graph Neural Network (GNN) policy trained via RL on procedurally generated environments that generalises across robot configurations and task layouts. Goarin and Loianno [45] propose a decentralised GNN for multi-robot goal assignment, in which each robot computes its own task assignment using local graph information without a central coordinator. Ho et al. [46] introduce federated deep

RL for task scheduling in heterogeneous autonomous robotic systems, allowing robots with different capabilities to learn collaboratively without sharing raw data, using a proximity weighted aggregation of local models.

2.4.3 Multi Agent and Hierarchical RL

For large, heterogeneous fleets, flat single agent policies struggle to capture inter robot dependencies. Gu et al. [47] propose a multi agent manufacturing system based on Internet of Things (IoT) for intelligent workshop scheduling, formulating the dynamic flexible job shop problem as a Markov decision process and training an RL scheduling mechanism that adapts in real-time to production changes. Zhao et al. [48] propose a multi agent attention model, based on an encoder decoder structure, for task scheduling in an AGV system, in which the decoder attends over node embeddings using a context vector that incorporates historical task information to select the next assignment. Geng et al. [49] and Agrawal et al. [50] propose integrated multi agent RL frameworks in which both manufacturing machines and mobile robots are modelled as agents, jointly optimising scheduling and transport decisions rather than treating them as independent subproblems. Bolu and Korcak [51] introduce an adaptive warehouse scheduling system that triggers re-planning only when observed execution deviates significantly from the current plan, reducing unnecessary computation. Dong et al. [52] address the flexible JSSP with a cooperative Deep Q-Network (DQN) for AGVs and machines, in which the reward structure incentivises agents to minimise collective makespan rather than individual performance.

Within this multi agent paradigm, Zhang et al. [53] extend deep RL to heterogeneous multi-robot systems with dynamic, prioritised task scheduling, proposing an actor critic attention mechanism that generates online task sequences while explicitly accounting for task priority. Notably, their formulation does not incorporate hard battery constraints, addressing dynamic urgency without addressing energy feasibility. Lassoued et al. [54] combine Coloured-Timed Petri Nets (CTPNs) with actor critic RL and dynamic action masking to reduce the decision space for AGV and tool sharing scheduling in flexible manufacturing system. Their masking mechanism, however, enforces structural feasibility with respect to the manufacturing process itself, rather than battery or energy constraints.

Across this body of work, a consistent pattern emerges: the majority of RL formulations for routing and scheduling problems either assume robots with unlimited energy [4, 5] or, when battery is considered at all, treat it as a soft penalty term within the reward function rather than a structural constraint the policy is guaranteed to respect. This is a meaningful gap in a paradigm otherwise well suited to the dynamic, continuously operating environments in which battery aware scheduling matters most, and constitutes the central motivation for the design choices presented in Chapter 3.

2.5 Battery and Energy Aware Fleet Scheduling

While battery aware scheduling within RL specifically remains comparatively underexplored, battery aware fleet scheduling as a broader problem has recently attracted growing attention. The electrification of industrial fleets transforms energy consumption from a fixed overhead into a dynamic variable that can be managed at the software level, but introduces the challenge of managing limited battery resources without compromising throughput.

2.5.1 Energy and Consumption Modelling

Several recent works incorporate explicit energy models into the scheduling formulation rather than treating battery level as a simple constraint. Ito et al. [55] propose a task allocation framework for robots that can switch between multiple locomotion modes, formulating a constrained optimisation problem that determines both the task assigned to each robot and the mode in which it should be executed, based on the energy required for each mode in the current environment. While the application targets heterogeneous multi modal fleets beyond the scope of this dissertation, the framework illustrates how energy cost can be made a first class input to task allocation.

2.5.2 Battery Degradation and Maintenance

A growing thread addresses not only runtime battery constraints but the longer term degradation of battery capacity caused by charging and discharging patterns. Atik et al. [56] demonstrate that integrating preventive maintenance decisions into the task allocation and charging schedule for AMR fleets substantially reduces both task downtime and battery degradation, compared to approaches that treat maintenance as an independent concern.

2.5.3 Fleet Level Optimisation

Chavan and Brocanelli [57] propose a task allocation and charging manager (TCM) for AMR fleets, formulating the problem as a Mixed-Integer Nonlinear Program (MINLP) that jointly minimises task downtime and maximises battery quality of life, and further propose a polynomial time greedy algorithm that closely approximates the exact MINLP solution (within a performance ratio of 1.15 in their experiments) at a fraction of the computation time. Li et al. [58] formulate a fleet level MILP that couples task assignment, sequencing, charging mode selection and shared charger access under explicit battery degradation objectives, solved via a hierarchical matheuristic that decomposes the fleet level problem into independently solvable per robot subproblems.

Taken together, these works confirm that battery aware, fleet level scheduling is an active and practically motivated research area. However, all of the above approaches rely on exact or greedy optimisation formulations solved anew, or approximately re-solved, at each planning step, rather than on a policy trained offline that produces a decision through a single forward pass at inference time. This distinction is not merely architectural: as discussed in Section 2.2, the computational cost of repeatedly solving an exact or near exact formulation online is precisely the property that

limits the applicability of such methods to fast paced, continuously operating industrial settings, a limitation this dissertation's benchmark results quantify directly in Chapter 5.

2.6 Positioning of This Work

Table 2.1 extends this comparison to the individual level, positioning every work cited in this chapter against five criteria directly relevant to this dissertation: whether the method is learned rather than search or optimisation based, whether it natively handles tasks arriving dynamically during operation, as opposed to a fully known, static instance, whether battery or energy is treated as an explicit constraint of the formulation, whether task priority is supported and whether the method is suited to frequent online planning without prohibitive computational cost. Entries marked "P" (partial) reflect cases where the property is present in a related but not fully equivalent form, for instance, a method that adapts to changing agent performance without handling newly arriving tasks, based strictly on what is confirmed in the descriptions given above. No entry is inferred beyond the evidence discussed in this chapter.

Table 2.1: Positioning of all reviewed works against the requirements of this dissertation. Y = yes, N = no, P = partial or qualified support.

Work	Method	Learned	Dynamic tasks	Energy aware	Priority aware	Low cost online
Classical / exact optimisation						
Sahar et al. [16]	MILP (tightened)	N	N	N	N	N
Yao et al. [17]	MILP (disjunctive graph)	N	N	N	N	N
Justkowiak et al. [18]	DP	N	N	N	N	N
Riazi and Lennartson [19]	CP / SMT	N	N	N	N	N
Zhang et al. [20]	Sim. budget allocation	N	P	N	N	P
Lippi and Marino [21]	MILP	N	P	N	N	N
Behrens et al. [22]	CP	N	N	N	N	N
Heuristic						
Avhad et al. [9]	Greedy / re-bid	N	Y	N	N	Y
Dang et al. [10]	Heuristic (batch)	N	Y	N	N	Y
Liu et al. [11]	MRIG	N	N	N	N	Y
Yi and Luo [12]	Petri Net + A*	N	N	N	N	Y
Bakshi et al. [13]	Quadratic prog.	N	N	N	Y	Y
Kim et al. [14]	Heuristic + battery	N	N	Y	N	Y
Chen and Liu [15]	Heuristic + battery	N	Y	Y	N	Y
Metaheuristic						
Li et al. [24]	Multi strategy GA	N	N	N	N	P
Han et al. [25]	Dual population GA	N	N	N	N	P
Cechinel and De Pieri [26]	Island Model GA	N	N	Y	N	P
Alzubairi et al. [27]	ED^{2}PSO	N	N	N	N	P
Mousavi et al. [28]	GA + PSO	N	N	Y	N	P

Continued on next page

Table 2.1 - continued

Work	**Method**	**Learned**	**Dynamic tasks**	**Energy aware**	**Priority aware**	**Low cost online**
Cui et al. [29]	Discrete Jaya	N	N	N	N	P
Sun et al. [30]	Chaotic SSA	N	N	N	N	P
Hybrid / learning assisted						
Zheng et al. [31]	MILP + TS	N	N	N	N	P
Dang et al. [32]	ALNS	N	N	N	N	P
Pan et al. [33]	Evolutionary Algorithm (EA) + RL selection	P	N	N	N	P
Lin et al. [34]	Cloud-edge scheduling	N	P	N	N	P
Xu et al. [35]	Multi objective EA	N	N	Y	N	P
RL - foundational routing						
Bello et al. [36]	Pointer Net + RL	Y	N	N	N	Y
Nazari et al. [5]	Attention + RL	Y	Y	N	N	Y
Kool et al. [4]	Attention + RL	Y	N	N	N	Y
RL - robot scheduling						
Durst et al. [38]	Deep RL (dispatching)	Y	Y	N	N	Y
Li et al. [39]	Deep RL (task selection)	Y	Y	N	N	Y
Tejer et al. [40]	Q-learning (robust)	Y	Y	N	N	Y
Cheng et al. [41]	Deep RL (cost minimisation)	Y	Y	N	N	Y
Shida et al. [42]	RL (exclusion levels)	Y	Y	N	N	Y
RL - scalable / distributed						
Zhang et al. [43]	Graph deep RL	Y	N	N	N	Y
Lai et al. [44]	GNN + RL	Y	N	N	N	Y
Goarin and Loianno [45]	Decentralised GNN	Y	N	N	N	Y
Ho et al. [46]	Federated deep RL	Y	Y	N	N	Y
RL - multi agent						
Gu et al. [47]	Multi agent RL	Y	Y	N	N	Y
Zhao et al. [48]	Multi agent attention	Y	N	N	N	Y
Geng et al. [49]	Multi agent RL (GNN)	Y	Y	N	N	Y
Agrawal et al. [50]	Multi agent RL (machines + robots)	Y	Y	N	N	Y
Bolu and Korcak [51]	Adaptive replanning	N	Y	N	N	Y
Dong et al. [52]	Cooperative DQN	Y	N	N	N	Y
Zhang et al. [53]	Actor critic attention	Y	Y	N	Y	Y
Lassoued et al. [54]	CTPN + actor critic RL	Y	P	N	N	Y
Battery and energy aware scheduling						
Ito et al. [55]	MRTA multi mode	N	N	Y	N	N
Atik et al. [56]	Scheduling + maintenance	N	P	Y	N	P
Chavan and Brocanelli [57]	MINLP + greedy	N	P	Y	N	P
Li et al. [58]	MILP + matheuristic	N	N	Y	N	P
This dissertation	**Attention + RL + masking**	**Y**	**Y**	**Y**	**Y**	**Y**

This comparison makes explicit that no single reviewed work combines all five properties simultaneously. Zhang et al. [53] comes close, satisfying four of the five criteria, but, as detailed

in Section 2.4, lacks battery awareness entirely.

Three gaps emerge directly from this pattern:

1. RL methods capable of dynamic, low cost online scheduling (Section 2.4) consistently lack structural battery guarantees;
2. Battery aware fleet scheduling methods that do provide such guarantees (Section 2.5) rely on comparatively expensive exact or greedy optimisation rather than a learned policy;
3. No reviewed work combines battery safety with priority aware dispatch.

This dissertation addresses all three jointly, by integrating a layered battery safety and priority mask directly into the decoder of an attention based policy trained on dynamically arriving tasks, as detailed in Chapter 3.

Chapter 3

Problem Formulation and Proposed Method

This chapter formalises the scheduling problem addressed in this dissertation and presents the method proposed to solve it. Section 3.1 restates the problem within the taxonomy introduced in Chapter 2 and formally defines the MRTSP-B. Section 3.2 specifies the operational constraints governing robot behaviour, covering movement, battery management and priority handling. Section 3.3 instantiates the problem with the concrete numerical parameters used throughout this work. Section 3.4 describes the unified state representation that governs the environment's dynamics. Section 3.5 presents the attention based encoder decoder architecture used to parameterise the scheduling policy. Section 3.6 details the layered action masking mechanism that enforces battery safety and priority preemption structurally, independently of the policy's training progress. Section 3.7 presents the cost function used to train the policy and the reasoning behind its design.

3.1 Problem Statement

As established in Section 2.1, this dissertation addresses a task scheduling problem classified as ST-SR-TA under the taxonomy of Gerkey and Matarić [6]: each robot executes one task at a time, each task requires exactly one robot and the scheduling policy reasons over a horizon of future tasks rather than committing greedily to the nearest available option. Building on this classification, and on the distinction between mobile robot task scheduling and the classical VRP established in Section 2.1, this dissertation defines the MRTSP-B as the specific problem instance solved throughout this work.

At each planning step, given the current positions and battery levels of both robots and the set of currently visible pending tasks, the objective of the MRTSP-B is to determine a joint route assignment $\pi = (\pi^{r_1}, \pi^{r_2})$ that minimises the fleet's makespan, i.e., the time at which the last task in the episode is completed, while satisfying three operational requirements simultaneously: every selected action must keep the acting robot able to safely return to a charging base, tasks may arrive dynamically at any point during the episode, requiring the policy to plan online and tasks marked

as priority must be addressed ahead of routine work whenever they are reachable, without ever compromising the first requirement (Section 3.2).

An episode is understood as one complete execution of the environment from its initial state until a terminal condition is reached: either the completion of all tasks or the exhaustion of the time budget detailed in Section 3.2. Concretely, an episode in this dissertation consists of 100 tasks in total, revealed dynamically over time such that at most 20 are visible to the scheduling policy simultaneously. The full set of episode parameters is detailed in Section 3.2.

Unlike a purely combinatorial formulation solved once for a fixed set of inputs, the MRTSP-B is solved repeatedly over the course of an episode: every time a robot becomes idle or a new batch of tasks becomes visible, the problem is instantiated again with the fleet's current state and a single next action is selected. Newly arrived tasks never interrupt a robot already executing an action, they only become available as candidates the next time that robot is due to make a decision. This planning centric formulation is what necessitates a policy capable of producing a decision quickly and reliably at every such step, rather than a method that recomputes a full schedule from scratch, motivating the RL approach detailed in Section 3.5.

3.2 Operational Constraints

The environment models a continuous, 24/7 operation, in which work orders arrive dynamically throughout an episode. Two robots operate within this environment, returning periodically to one of two charging bases to recharge. Robot behaviour is governed by the following operational constraints, covering both movement and energy management.

- Each robot moves at a constant speed, with inter location distances computed using the Euclidean metric.
- Battery charge depletes linearly with distance travelled and must remain above zero at all times. If a robot's battery is fully depleted mid route, the robot is immobilised and the episode is considered failed.
- Robots must additionally maintain their battery above a critical threshold of 20% of maximum capacity at all times. A task is only undertaken if the robot can complete it and still return to a charging base without the battery dropping below this threshold.
- Recharging proceeds until a target level 80% of maximum capacity is reached, after which the robot is released for new task assignments. If no tasks are pending at that moment, charging continues until the battery reaches 100%.
- When no pending tasks are visible in the environment, robots are directed to return to a charging base to await new assignments, so that idle time is used productively to restore battery in preparation for the next task, rather than remaining stationary away from a base with no means of recharging.

- The two robots are homogeneous, sharing identical speed, battery capacity and capabilities, and either robot may charge at either base, this means that a robot is never restricted to the base nearest its starting position.
- Each base can only accommodate one robot at a time: if a robot arrives at a base already occupied by the other, it must queue and wait.
- Regarding task priority, when one or more priority tasks are reachable given the robot's current battery level, the robot is directed exclusively towards them and may not select standard tasks. If a robot is currently recharging and a priority task becomes reachable with the available charge, the robot may depart before reaching the target level, provided it can complete the task and return to a base safely.
- Priority does not interrupt tasks already in progress: once a robot has committed to and begun executing a task, it completes that task before reconsidering its next assignment.

Together, these constraints define the complete MRTSP-B instance addressed throughout the remainder of this dissertation. Section 3.3 instantiates this problem with concrete numerical parameters and presents the proposed method for solving it: a unified state representation that encodes these constraints directly into the transition dynamics of the environment, and a layered action mask that enforces them structurally at every decision made by the learned policy.

3.3 Environment

The environment introduced conceptually in Section 3.1 is instantiated with the following concrete parameters, illustrated in Figure 3.1. Robots carry a battery of capacity $B_{\max} = 3.0$ (100%), with critical threshold $B_{\text{crit}} = 0.6$ (20%) and target recharge level $B_{\text{target}} = 2.4$ (80%), following the percentages introduced in Section 3.2. Docking at a base takes a fixed 2.0 time units, followed by linear recharging at a rate of 0.1 time units per percentage point of battery restored. Each robot moves at a constant speed $v = 0.5$ units per second, with the distance between any two locations computed using the Euclidean metric. Because both bases are always reachable in principle, choosing which one to visit is itself part of the scheduling decision: the policy must weigh the distance to each base against the risk of arriving to find it occupied and having to wait, rather than defaulting to the nearest option regardless of the fleet's current state.

3.3.1 Calibration of Battery Parameters

The value of $B_{\max} = 3.0$ was chosen relative to the scale of the workspace $W = [0,1]^2$. Although $B_{\max}$ represents 100% of battery capacity, a robot only charges beyond the target level $B_{\text{target}} = 2.4$ (80%) when idle and no tasks are pending, so task execution is planned starting from B_{target}, not from $B_{\max}$, and every plan must additionally ensure the robot can return to a base while retaining at least the critical threshold $B_{\text{crit}} = 0.6$ (20%). This leaves a usable margin for task execution of:

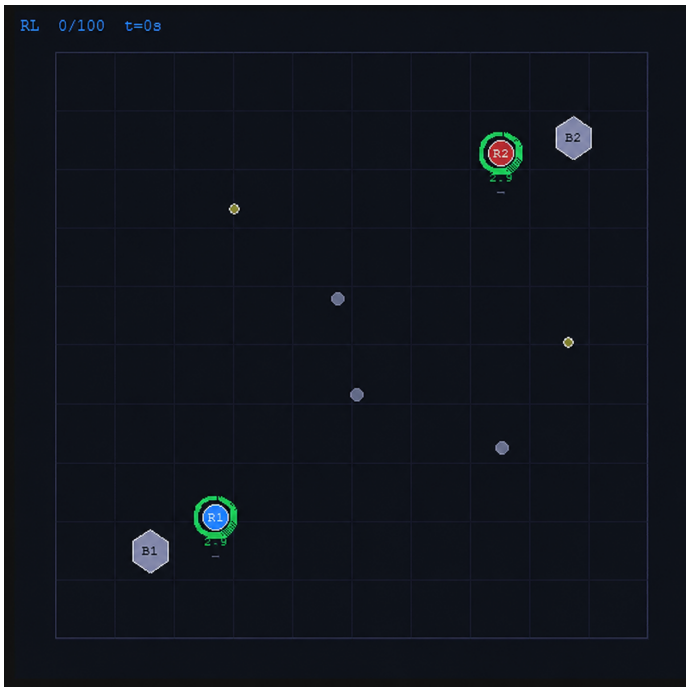


Figure 3.1: The simulated environment shortly after episode start, showing the two robots (r_1, r_2) departing their respective bases (b_1, b_2) towards their first assigned tasks, with further pending tasks distributed across the workspace.

$$B_{\text{target}} - B_{\text{crit}} = 2.4 - 0.6 = 1.8. \tag{3.1}$$

This margin is deliberately chosen to be slightly above the maximum possible distance between any two points in the workspace ($\sqrt{2} \approx 1.41$), which means a robot can, in principle, visit more than one task per charge cycle. At the same time, 1.8 is not so large that the robot can chain many tasks together indefinitely. Depending on their locations, the remaining battery will force a return to a base. The robot must constantly decide which task to visit next, whether to return to base first and which base to use, taking into account both its own battery and the partner robot's state. This is what makes active battery management an integral part of the scheduling problem rather than a background concern: the robot cannot simply pick the nearest task greedily, because doing so may leave it without enough battery to complete the trip safely or to coordinate effectively with the other robot.

3.3.2 Episode and Task Parameters

An episode consists of 100 tasks in total, chosen to simulate a sufficiently long operation that requires multiple recharging cycles and planning events, generated at random locations uniformly distributed within the workspace and revealed dynamically over the course of the episode, in batches of 1 to 3 tasks arriving at intervals of 2 to 6 simulated seconds. At any point in time, at most 20 tasks are visible simultaneously. This simplification keeps the size of the decision

space bounded and tractable for the attention based architecture presented in Section 3.5, while preserving the fundamentally dynamic, partially observable nature of the environment. Among the tasks visible at any given planning step, between 2 and 4 are designated as priority.

3.4 State Representation

The state transition function $F(s,a) \rightarrow s'$ is the single source for all physical dynamics in the environment: battery consumption, charging duration (including docking time and any queuing at a shared base), timing penalties, and failure detection are all computed within this one function, shared identically between training and deployment. This unified design is a deliberate departure from architectures in which battery management is handled by a module external to the learned policy. It guarantees that the cost function observed during training is exactly identical to the cost observed at deployment, eliminating a source of train deployment mismatch that would otherwise undermine the reliability of the learned policy.

Scheduling decisions are event driven. A planning step is triggered whenever a robot becomes idle with pending tasks visible, or when new tasks appear in the environment. At each such step, the robot with the lower accumulated episode time acts first, selecting the next node to visit from the set of currently feasible actions, as determined by the action mask presented in Section 3.6.

3.5 Attention based Architecture

The scheduling policy is parameterised by an attention model adapted from the routing architectures reviewed in Section 2.4, illustrated in Figure 3.2. The model has two jobs: first, it must build a rich internal representation of every node (every task and every base); second, at each decision point, it must use that representation, together with information about where both robots currently are, to decide which node the active robot should visit next. The remainder of this section describes how each of these two jobs is carried out.

3.5.1 Encoding the Graph

Each node i starts out as a simple three value description: its coordinates and its priority flag, (x_i, y_i, p_i), with $p_i \in \{0,1\}$ as introduced in Section 3.1. This raw description is minimal and by itself is not enough for the model to reason about which node is a good choice. The purpose of the encoder is to transform this raw description into an embedding $\mathbf{h}_i \in \mathbb{R}^{d_h}$ that does capture this relational information.

Concretely, a shared linear layer first projects each node's 3-dimensional input into a $d_h =$ 128-dimensional vector. The additional dimensions give the model the capacity to encode richer, relational information about each node. After passing through the attention layers described below, each node's 128-dimensional vector, referred to as its embedding $\mathbf{h}_i$, will have been updated to reflect context that the raw coordinates alone cannot capture, such as whether the node is near

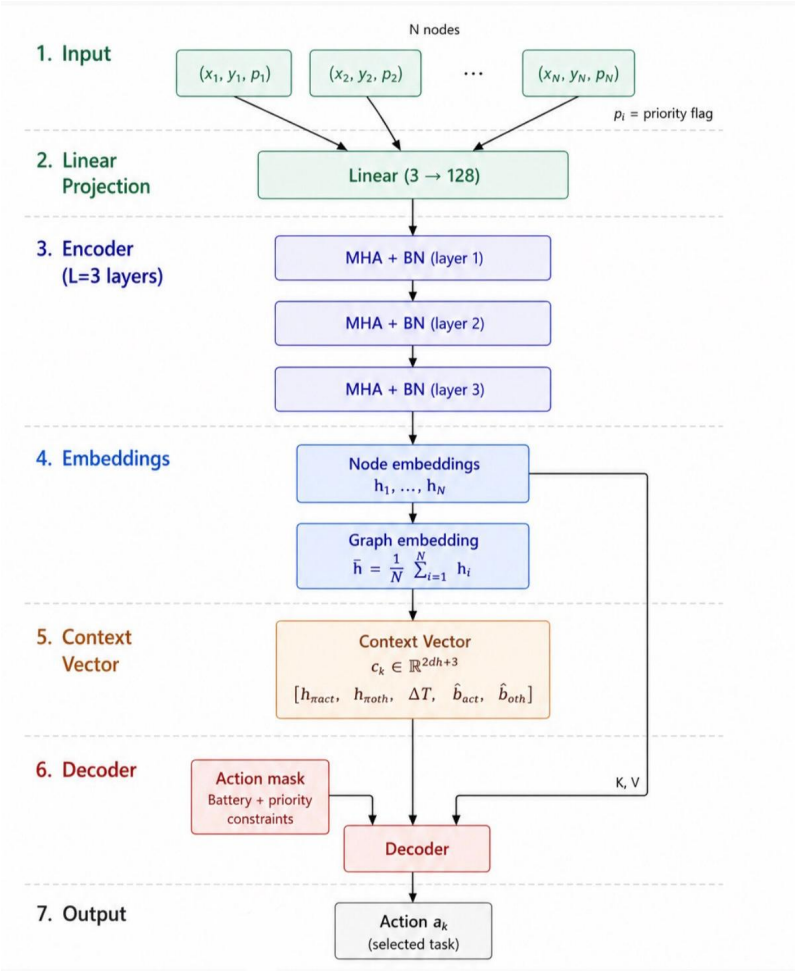


Figure 3.2: Architecture of the proposed attention model for the MRTSP-B with priority tasks.

other urgent tasks or how far it lies from either base. This richer representation is what allows the decoder to make informed scheduling decisions rather than reasoning purely from geometry.

This vector is then progressively refined by $L = 3$ attention layers. In each layer, every node is allowed to ”look at” every other node in the graph and update its own representation based on what it sees. For example, a node representing an urgent task near the top of the map will, after attending to its neighbours, end up with a different embedding than the same task would have if it were isolated in a corner with no nearby tasks and no reachable base: the attention mechanism allows each node's representation to absorb context from the graph as a whole, not just from its own raw coordinates. Repeating this process $L = 3$ times allows information to propagate further across the graph with each layer. After one layer, a node knows about its immediate neighbours. After three, it has absorbed context from nodes further away. Kool et al. [4] found $L = 3$ to be a good trade-off between the quality of the resulting representations and computational complexity, and this value is adopted here without modification.

The specific values $d_h = 128$ and 8 attention heads follow the configuration used by Kool et al. [4], who demonstrate that these same hyperparameters generalise well across five structurally different combinatorial routing problems without problem specific tuning. For the number of encoder layers, Kool et al. [4] note that $L = 3$ represents a good trade-off between solution quality and computational complexity. Adopting this configuration for the MRTSP-B is justified by this demonstrated cross problem robustness: the MRTSP-B shares the same fundamental structure as the problems for which this configuration was validated (a set of nodes with coordinates, visited sequentially one at a time), and independently tuning these values through a systematic ablation study is outside the scope of this dissertation.

3.5.2 Deciding the Next Action

Once every node has an embedding, the decoder uses them to choose the next node to visit. To do this, it needs to answer the question: given where both robots are right now, how much battery they have and what the full task landscape looks like, which node should the active robot visit next?

The active robot's current location is not supplied to the model as a separate pair of coordinates. Instead, the model identifies which node the robot currently occupies and uses that node's embedding as a stand in for its position, keeping everything in the same 128-dimensional representational space as the tasks and bases. When the robot is exactly at a base or has just completed a task, this is simply that node's own embedding. When it is mid route between two locations, a temporary anchor node is inserted into the graph at the robot's exact coordinates (as described in Section 4.2.3) and it is this anchor's embedding that is used instead. The same applies to the partner robot.

At each decoding step k, the decoder forms a query: a single vector that summarises everything the model needs to know before making its decision. This query has two components, the first is a fixed summary of the whole graph: the mean of all node embeddings $\bar{\mathbf{h}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}_i$, which captures the overall structure of the current task landscape in a single vector. The second is a step

varying context vector $\mathbf{c}_k$ that carries the robots' current state (where they are, how much battery they have, and how much time separates them). These two components are combined by a pair of learned weight matrices W_f and W_s:

$$\mathbf{q}_k = W_f \bar{\mathbf{h}} + W_s \mathbf{c}_k. \tag{3.2}$$

Unlike the architectural hyperparameters discussed above, W_f and W_s are not set by hand: they are learned automatically during training, adjusted by gradient descent alongside every other weight in the model.

Concretely, the step varying context vector is:

$$\mathbf{c}_k = \left[\mathbf{h}_{\pi^{\text{act}}},\, \mathbf{h}_{\pi^{\text{oth}}},\, \Delta T,\, \hat{b}_{\text{act}},\, \hat{b}_{\text{oth}}\right], \tag{3.3}$$

where $\mathbf{h}_{\pi^{\text{act}}}$ and $\mathbf{h}_{\pi^{\text{oth}}}$ are the embeddings representing the positions of the active and partner robots respectively, $\Delta T = T_{\text{oth}} - T_{\text{act}}$ is the time difference between them (which tells the model how coordinated the fleet currently is) and $\hat{b}_{\text{act}}, \hat{b}_{\text{oth}} \in [0,1]$ are their normalised battery levels. Packaging both robots' state into a single vector is what allows the decoder to condition its decision on the fleet as a whole: the model does not schedule for one robot in isolation, but reasons jointly about both.

Using $\mathbf{q}_k$, the decoder computes a compatibility score between the query and every node's embedding, asking each node "how well do you match what we are looking for right now?". A tanh clipping with constant $c = 10$ is applied to keep these scores in a bounded range. Without this, the compatibility scores can grow without limit during training, causing the model to become overconfident, assigning near zero probability to every node except one, regardless of context, and becoming increasingly difficult to train. Bounding the scores within $[-c, c]$ prevents this, preserving a meaningful distribution over candidate nodes throughout training. The specific value $c = 10$ is adopted from Kool et al. [4] without modification. It represents an empirically validated choice that has been adopted widely in subsequent work using this architecture, rather than a value derived from first principles. The bounded scores are then converted into a probability distribution over all nodes via a softmax, which maps any set of real valued scores to values between 0 and 1 that sum to 1, preserving their relative order while producing a valid distribution that the REINFORCE training objective (Equation 5.1) requires. At inference time, the node with the highest resulting probability is selected deterministically (greedy decoding). The layered action mask presented in Section 3.6 is then applied to zero out any infeasible node, after which the node with the highest remaining probability is selected as a_k.

3.6 Action Masking

Rather than only relying on the learned policy to discover safe and priority respecting behaviour purely from the reward signal, this dissertation enforces both properties structurally, at the decoder, through a mask computed in three layers. Because the mask operates directly on the output

distribution of the decoder, an action found infeasible by any layer can never be selected, independently of how well trained the underlying policy is. This is the central design principle that distinguishes the approach from the RL formulations reviewed in Section 2.4, in which battery is typically represented, at best, as a soft term within the reward signal.

3.6.1 Battery Safety

Node i is blocked unless the active robot can reach it and still return to the farthest charging base without its battery dropping below B_{crit}. An earlier version of this criterion evaluated this return trip margin against the nearest base to node i, which was found to be insufficient. Since the robot is free to choose either base when returning, nothing in that formulation prevented the robot from selecting a task that appeared safe under the nearest base margin, only to run out of charge if it subsequently chose to return via the farther base. This way, a safe return is guaranteed regardless of which base the robot ultimately selects, closing this gap at the cost of a more conservative feasibility criterion.

3.6.2 Priority Enforcement

If at least one priority task is battery feasible under the criterion above, all non priority tasks are blocked, directing the robot exclusively towards urgent work, as specified in Section 3.2. If the robot is currently charging and a priority task becomes feasible with the battery level available at that moment, the corresponding base action is unblocked, allowing the robot to depart before reaching B_{target} without compromising its ability to complete the priority task and return safely.

3.6.3 Premature-return prevention

When the robot is away from a base with battery above 70% of B_{max} and at least one feasible task exists, returning to a base is blocked, preventing premature returns that would otherwise inflate makespan without any corresponding benefit to battery safety. This constraint applies only to the decision to travel to a base, it does not interrupt a charging cycle already in progress, which always proceeds until B_{target} is reached.

The 70% threshold was chosen empirically: at this charge level, a feasible task was available in practice across the scenarios tested during development. A lower threshold of 50% was tested first, and caused deadlocks: the robot was blocked from returning to charge, by this rule, while simultaneously having insufficient battery to complete any available task under the criterion of Section 3.6.1, leaving it with no valid action at all. Raising the threshold to 70% resolved this specific failure mode, though, as discussed in Chapter 4, it did not eliminate every scenario in which the fleet can become unable to make further progress.

3.6.4 Fallback Rule

The three layers above can, in combination, leave every node blocked simultaneously. This scenario is most common when the battery safety criterion of Section 3.6.1 rules out every visible task given the robot's current position and battery level. A fallback rule ensures the mask is never left entirely empty: if all actions are blocked, the nearest base is unconditionally unblocked, guaranteeing the robot always has at least one valid action available, at the cost of temporarily suspending the return margin normally required by Section 3.6.1 for that single decision. The frequency with which this fallback is triggered, and its consequences for episode completion, are examined empirically in Chapter 5.

3.7 Cost Function

The scheduling problem is formulated, at each planning step, as finding a joint route assignment $\pi = (\pi^{r_1}, \pi^{r_2})$ that minimises the cost function

$$C(\pi) = \alpha \cdot T_{\text{span}}(\pi) + \beta \cdot D_{\text{total}}(\pi) + P_{\text{timing}}(\pi) + P_{\text{pri}}(\pi), \tag{3.4}$$

where $T_{\text{span}}(\pi) = \max(T_{r_1}, T_{r_2})$ is the fleet's makespan, $D_{\text{total}}(\pi) = D_{r_1} + D_{r_2}$ is the total distance travelled by both robots, and $\alpha = 10$, $\beta = 1$ are weighting coefficients. The coefficients were chosen to reflect the relative importance of the two objectives and to account for the difference in scale between them. Makespan is the primary objective: completing all tasks as quickly as possible is the central goal of the scheduling problem. Distance is a secondary objective, acting as a tiebreaker between plans of similar makespan and discouraging unnecessarily inefficient routes that would deplete battery faster and leave the fleet in a worse position for subsequent tasks. In a typical episode, T_{span} lies in the range of 200 to 300 seconds while D_{total} lies in the range of 50 to 90 distance units. Setting $\alpha = 10$ and $\beta = 1$ ensures that makespan dominates the gradient signal while distance remains a meaningful secondary signal, rather than being negligible relative to the primary term.

3.7.1 Timing Penalty

The timing penalty P_{timing} ramps up as a robot's battery approaches the critical threshold, in two zones: a warning zone between 25% and 20% of $B_{\max}$ and a danger zone between 20% and 0%. This discourages the policy from operating close to the safety margin even though the action mask, presented in Section 3.6, already prevents outright battery failure. Battery safety is enforced structurally by the mask and a dedicated hard penalty for battery failure was therefore deliberately omitted from the cost function: such a term would never actually be triggered under a correctly functioning mask and including it regardless was found during development to inject unnecessary variance into the training signal without contributing any additional constraint the mask does not already enforce.

3.7.2 Priority Penalty

The priority penalty P_{pri} accumulates a cost C_{pri} per scheduling step in which a priority task remains unresolved beyond the current action, creating a soft ordering preference that complements the hard preemption enforced by the mask in Section 3.6.2, without imposing a hard deadline of its own. This value was reduced from an initial $C_{\text{pri}} = 100$ to $C_{\text{pri}} = 10$ after observing that the larger weight caused the priority term to dominate the gradient signal, accounting for an average of 46% of the total cost across training batches and overriding the makespan and distance objectives it was intended to complement, rather than balance against them. This adjustment illustrates a broader point returned to in Chapter 4: the individual terms of Equation 3.4 were tuned in response to specific, observed training pathologies encountered during development, rather than fixed in advance from first principles.

Together, the state representation, attention based architecture, layered action mask and cost function presented in this chapter define the complete scheduling policy evaluated in Chapter 5.

Chapter 4

Implementation and Development Process

This chapter documents the process behind the method presented in Chapter 3: the computational infrastructure available over the course of this work (Section 4.1), a series of correctness issues discovered in the evaluation harness used to benchmark the trained policy, whose resolution materially changed the conclusions that could be drawn from the experiments (Section 4.2), the steps taken to ensure a fair comparison against the baseline methods introduced in Chapter 5 and the relationship discovered between training duration and deployment performance, which shaped the checkpoint selection methodology used to report the results in Chapter 5.

4.1 Development and Training Infrastructure

Training the policy described in Chapter 3 required continuously available computational resources. The environment used was a workstation equipped with an NVIDIA RTX 6000 Ada Generation GPU with 48 GB of VRAM.

4.2 Correcting the Benchmark Evaluation Harness

The action mask described in Section 3.6 was designed to guarantee, by construction, that a robot can never be left unable to return to a charging base. This guarantee is a property of the mask's logic, evaluated correctly against the true state of the environment at the moment a decision is made. It does not, by itself, guarantee that a separate evaluation harness built to simulate complete episodes and benchmark the trained policy against classical baselines queries that state correctly. Over the course of developing such a harness, four distinct implementation errors were identified that caused it to misrepresent the policy's true behaviour, each corrected once identified.

4.2.1 Stale Action Queues

The first version of the evaluation controller for the RL policy queried the model once per planning event and cached the resulting sequence of actions for each robot, executing them one at a time as the episode progressed, rather than querying the model before every single action. When a robot became idle before its cached queue was exhausted, a new sequence was requested and appended to the existing queue via list concatenation, without first checking whether the queue targeted was actually empty. Since both robots' queues were updated jointly on every planning event this allowed a new sequence to be appended to a queue that still contained unexecuted actions from a previous decision, computed against a state that no longer matched the robot's true position and battery once execution reached that point. The queued actions themselves remained individually valid at the moment they were computed. The fault was that a later append operation could silently discard the guarantee that the queue as a whole reflected the robot's real trajectory, since a task earlier in the queue could be skipped or reassigned without the entries after it being recomputed.

This was corrected by re architecting the controller to query the model for exactly one action immediately before it is required, always against the robot's real, current position and battery, rather than computing and caching sequences of actions in advance.

4.2.2 Unbounded Decoder Queries

The step by step controller resulting from the correction above introduced a new failure mode of its own. If the policy repeatedly selected an action equivalent to remaining at its current base, which the mask permits when no other node is currently feasible, the controller would query the model again on the very next simulated tick, with the environment state essentially unchanged, producing a tight loop of decoder calls that consumed disproportionate compute time without any corresponding progress in the episode. This was observed directly: cumulative decision latency for a single episode rose from under one second to approximately 29 seconds once this pattern was triggered. A simple safeguard was introduced: if a robot receives no actionable decision from the model over a bounded number of consecutive queries, the controller stops querying it until the environment state changes. For instance, when a new task becomes visible, the counter is reset. This eliminated the runaway compute cost without altering any actual scheduling decision, confirmed by comparing episode outcomes before and after the fix on identical seeds.

4.2.3 Missing Position Context

A more subtle issue affected the correctness of decisions themselves rather than only their computational cost. The state representation described in Section 3.4 tracks each robot's current node, used both by the mask and by the decoder's context vector (Equation 3.3). The harness's state initialisation routine defaulted this current node reference to the two charging bases, an assumption that holds only when an episode first begins. Because the step by step controller introduced in Section 4.2.1 constructs a fresh problem instance before every single decision, this default was

re applied at every step, causing the decoder to be told, in effect, that both robots were always located at a base, regardless of their true position on the map.

This was corrected by introducing an anchor mechanism: whenever a robot's true position does not coincide with a base, an additional node is inserted into the problem instance at that exact position, and the current node reference is set to point to it instead of defaulting to a base. This ensures the decoder always reasons from the robot's true location.

4.2.4 Graph Size Overflow

The anchor mechanism introduced in Section 4.2.3 could, in turn, push the total number of nodes presented to the model beyond the fixed graph size the architecture was trained on: two bases plus up to 20 visible tasks. This was resolved by reserving space for the anchors before populating the visible task set, truncating the number of tasks considered in that specific decision if necessary. Tasks excluded in this way remain pending and are reconsidered at the robot's next decision point, so no task is permanently lost as a result.

4.3 Priority Weighting in the Exact Solver

Beyond correctness issues affecting the RL policy specifically, a further issue was identified that affected the fairness of the comparison against the classical baselines evaluated.

The mask described in Section 3.6.2 enforces priority preemption as a hard constraint for the RL policy. Replicating an equivalent hard constraint within the exact solver's CP formulation was not straightforward without altering its objective substantially. Instead, a weighting term was added to the solver's objective function, scaling the assignment cost of each priority task by a constant factor so that the solver is biased towards assigning priority tasks to whichever robot can reach them fastest, in addition to its primary makespan objective. This weighting factor was set heuristically, based on observing its effect on priority delay in preliminary runs, rather than tuned thoroughly. This remains a limitation of the comparison discussed further in Chapter 6.

Chapter 5

Experimental Setup and Results

This chapter reports the training procedure, experimental setup, and benchmark results obtained with the method presented in Chapter 3, evaluated using the corrected benchmark harness described in Chapter 4. Section 5.1 describes the training procedure and the checkpoint selected for evaluation. Section 5.2 describes the baseline methods and metrics used for comparison. Section 5.3 reports the results of the benchmark evaluation across 1,000 dynamic episodes.

5.1 Training

The model is trained using the REINFORCE algorithm with a rollout baseline. At each training epoch, a dataset of instances is generated by simulating complete episodes on-policy. The current model is used to make scheduling decisions and each planning moment produces one training instance reflecting the real state of the robots (position and battery) at that point. This on-policy generation ensures the training distribution matches the deployment distribution, avoiding the distribution shift that arises from training on randomly generated static instances. Tasks arrive in batches of 1 to 3 at intervals of 2 to 6 simulated seconds, and 2 to 4 of the tasks visible at any given step are designated as priority, matching the episode parameters introduced in Section 3.3.2. The training objective minimises Equation 3.4 via gradient descent on the policy parameters θ:

$$\nabla_\theta \mathscr{L}(\theta) = \mathbb{E}_{\pi \sim p_\theta} \left[(\mathscr{C}(\pi) - b(s)) \nabla_\theta \log p_\theta(\pi \mid s) \right], \tag{5.1}$$

where $b(s)$ is the rollout baseline estimate. Table 5.1 lists the key training parameters. Architecture and optimisation hyperparameters follow the configuration of Kool et al. [4] and the problem specific values discussed individually in Chapter 3 were tuned in response to observed training behaviour.

5.2 Experimental Setup

The proposed method is evaluated on a benchmark of 1,000 episodes, using the same physical parameters described in Chapter 3. All methods share the same task generation procedure and the

Table 5.1: Key training parameters.

Parameter	Value
Graph size (bases + tasks)	22
Batch size	512
Epoch size	128 000
Embedding dimension d_h	128
Encoder layers L	3
Attention heads	8
Learning rate	10^{-4}
Baseline	Rollout
Warmup epochs	1
Robot speed v	0.5 units/s
Battery capacity $B_{\max}$	3.0
Priority cost C_{pri}	10

same battery safety rule, so that differences in outcome can be attributed to the scheduling method itself, rather than to inconsistent safety rules across methods. Two baseline methods are compared against the proposed policy, NN and Google OR-Tools:

NN is a standard greedy baseline. At each planning step, each robot selects the closest pending task for which a safe round trip is feasible. A robot reroutes to the nearest base whenever $b - d(\mathbf{q},\mathbf{t}) - d_{\text{back}} < B_{\text{crit}}$, where $d(\mathbf{q},\mathbf{t})$ is the distance to the task and d_{back} is the distance from the task to the farthest base.

Google OR-Tools solves the routing problem at each planning step [23], with a time limit of 3 seconds per call, modelling battery feasibility as a constraint mirroring the criterion in Section 3.6.1.

Priority handling follows the natural process of each solver. For NN, feasible priority tasks are considered exclusively when available. For OR-Tools, each priority task distance term is scaled by a weighting factor (x5) in the objective, biasing assignment toward the robot that can reach it fastest. Within a robot's queue, feasible priority tasks are sequenced first. In both baselines, a robot below the 80% recharge target may depart early if a feasible priority task becomes available, mirroring the exception applied to the RL policy.

A Pygame based simulator visualises all three methods side by side in real-time for qualitative analysis, as shown in Fig. 5.1.

Various metrics are reported, including episode completion rate, average makespan and distance travelled for successful episodes, average time spent queueing at a shared charging base, average priority task delay, battery death rate and cumulative computation time per episode. An episode is considered failed by planning timeout if it does not complete all 100 tasks within a fixed budget of 600 simulated seconds, a limit set at more than double the highest average makespan observed among successful episodes across all three methods. Because the timeout budget sub-

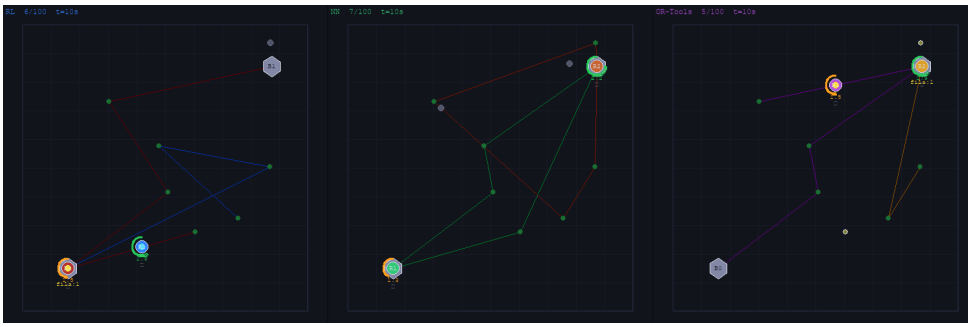


Figure 5.1: Qualitative comparison of the three evaluated methods (RL, NN, OR-Tools, left to right) on the same episode instance.

stantially exceeds a typical episode's duration, a timeout failure does not indicate an episode that was merely slow, it indicates that the fleet became genuinely unable to make further progress, well beyond what any successful episode requires. This is distinct from battery death, in which a robot's charge reaches zero mid route.

5.3 Results

Table 5.2 summarises the outcomes over 1,000 dynamic episodes.

Table 5.2: Benchmark results (1,000 episodes, 100 tasks each).

Metric	**RL**	**NN**	**OR-Tools**
Completion rate (%)	99.8	74.3	97.0
Avg. makespan (s)	290.2	197.3	206.4
Avg. distance travelled	89.9	50.9	49.4
Avg. queue wait (s)	36.0	52.6	35.6
Avg. battery at charge (%)	38.2	42.5	42.2
Priority delay (s)	4.0	2.9	4.4
Battery death rate (%)	0.0	0.0	0.0
Cumulative compute time (s)	15.9	0.0	141.7

Metrics other than completion rate and battery death rate are computed over successful episodes only.

Episode completion: The RL policy completes 998/1,000 episodes (99.8%), matching the reliability of the exact solver (970/1,000, 97.0%) and substantially exceeding the greedy baseline (743/1,000, 74.3%). All episode failures across all three methods stem from planning timeouts rather than battery exhaustion (0% battery death rate for all methods), confirming that the layered action mask, applied consistently to all three approaches, prevents failure regardless of the scheduling strategy used.

Planning timeouts arise when the battery safety mask blocks every visible task simultaneously. In this state, the only unmasked action is a return to base. If the fleet repeatedly re-enters this state,

for instance, when the visible task set remains geometrically unfavourable relative to the farthest base, the robot cycles between recharging and re-evaluating an empty task set without making further progress, eventually exhausting the episode's time budget.

Makespan and travel efficiency: Despite matching the exact solver in completion rate, the RL policy incurs a higher average makespan (290.2s vs. 197.3s and 206.4s) and travels considerably further (89.9 vs. 50.9 and 49.4). A key distinction between the methods underlies this difference: NN and OR-Tools only visit a base when forced to do so, either by the 80% recharge rule or by the battery safety mask, whereas the RL policy actively manages when to recharge as part of its scheduling strategy, anticipating future battery needs rather than reacting to them. This reflects a deliberate coordination strategy rather than inefficient routing.

Queue congestion: The RL policy spends on average 36.0s per episode queueing at a shared charging base, close to the exact solver (35.6s) and notably lower than the greedy baseline (52.6s). This indicates that the joint battery and position context available to the decoder is sufficient to reduce, though not eliminate, the two robots' tendency to converge on the same base simultaneously.

Battery utilisation: The RL policy arrives at a charging base with a lower average battery level than either baseline (38.2% vs. 42.5% for NN and 42.2% for OR-Tools). Since no method experiences battery failure, this indicates that the policy operates with a tighter, more efficient battery margin than either traditional planning method, without compromising safety.

Priority handling: The RL policy resolves priority tasks with an average delay of 4.0s, faster than the exact solver (4.4s) despite the latter's explicit priority weighted objective and slower than the greedy baseline (2.9s), which reconsiders priority feasibility immediately whenever a robot becomes idle. This confirms that the mask based mechanism responds to urgency at least as effectively as an optimisation based approach constrained by a fixed 3 second planning cadence.

Computational cost: The RL policy requires 15.9s of cumulative wall clock time per episode, an order of magnitude less than the exact solver (141.7s) under the same evaluation conditions, roughly $9\times$ lower, confirming the model remains suitable for real-time deployment. NN incurs negligible computational cost, as it performs no optimisation or learned inference.

Together, these results demonstrate that the proposed RL policy matches the reliability of an exact solver and substantially exceeds a greedy heuristic, at a fraction of the exact solver's computational cost, while revealing specific trade-offs, such as route efficiency, that are examined further in Chapter 6.

Chapter 6

Discussion and Limitations

Taken together, the results in Chapter 5 demonstrate that the proposed RL approach establishes a compelling paradigm for multi-robot task scheduling, one in which battery safety is not an afterthought bolted onto a scheduling policy, but a structural, first class property of the policy itself. Most notably, the policy never loses a robot to battery failure, achieving a 99.8% episode success rate across the full 1,000 episode benchmark.

When contrasted with the OR-Tools baseline, the proposed approach reveals a decisive advantage in operational scalability. While exact solvers provide high quality routing, their computational demand renders them increasingly impractical for real-time planning as an episode progresses, incurring a cumulative compute cost of 141.7s per episode. In contrast, the proposed architecture delivers decisions in approximately 16 seconds per episode, offering the responsiveness required for real-time automation in which tasks arrive continuously throughout operation. This confirms that a learned policy of this kind provides a lightweight, real-time capable alternative to repeated exact optimisation, without sacrificing the reliability exact optimisation is valued for.

The RL policy's higher average makespan and travel distance relative to the greedy baseline reflect, in part, the density of the evaluation environment introduced in Chapter 3. With a normalised unit square workspace and up to 20 simultaneously visible tasks, task locations are naturally close to one another, favouring a purely greedy nearest task strategy well suited to short, local routes. This advantage would plausibly narrow in larger or sparser environments, where the greedy baseline's lack of lookahead becomes more costly, a hypothesis this dissertation does not test directly, but that follows naturally from the myopic nature of NN dispatch discussed in Chapter 2. This makespan gap should, in any case, be weighed against reliability: the greedy baseline fails to complete 257 of 1,000 episodes due to planning timeouts arising from its purely reactive, non anticipatory task selection, compared to only 2 for the RL policy, whose learned behaviour anticipates and avoids many of the states that leave the greedy baseline without a feasible next action.

A small number of methodological choices constrain how broadly these results generalise and are worth stating plainly. The comparison against OR-Tools relies on a priority weight that was set

heuristically rather than exhaustively tuned and the model's performance heavily depends on the training data distribution and may not generalise well to significantly different environments, such as larger maps, different task densities or additional robots. Furthermore, the use of deep neural networks reduces interpretability, making it difficult to explain individual decisions. The training process is computationally expensive and relies on the REINFORCE algorithm [37], which is known to suffer from high gradient variance and relatively slow convergence. In addition, the attention mechanism scales poorly with the number of tasks, increasing memory and computational requirements for larger problem instances. Finally, the learned policy is strongly dependent on the accuracy of the simulation environment and may require retraining when facing substantial changes in operating conditions or episode requirements.

None of these limitations affect the central, structural guarantee this dissertation set out to establish (battery failure is impossible by construction) and each identifies a concrete direction for the future work outlined in Chapter 7, rather than a weakness of the underlying approach itself.

Ultimately, this work demonstrates that an attention based RL agent can autonomously navigate the trade-off between task urgency, battery safety and fleet coordination, matching the reliability of classical exact methods while requiring a fraction of their computational cost. The results suggest that structural enforcement of operational constraints at the decision level, rather than through reward shaping alone, is a promising direction for building robust scheduling policies in continuously evolving environments.

Chapter 7

Conclusion and Future Work

7.1 Contributions

This dissertation set out to design and evaluate an RL approach to multi-robot task scheduling that integrates battery management and priority handling natively into a learned policy, rather than through an external safety module layered on top of it. Chapter 2 identified three specific gaps in the existing literature motivating this approach: attention based RL methods for routing and scheduling typically ignore battery constraints entirely or treat them as a soft, unguaranteed penalty. The resulting policies are predominantly trained and evaluated on static instances rather than on the dynamically evolving, continuously operating environments in which battery aware scheduling matters most. In addition, no reviewed work combines battery safety with priority aware dispatch within a single mechanism.

Chapter 3 addressed these gaps through a unified state representation, in which battery dynamics, charging queues and timing penalties share a single source of physical truth between training and deployment, and a layered action mask that enforces battery safety and priority preemption structurally at the decoder, guaranteeing by construction that no task directed action can leave a robot unable to return to a charging base, independently of how well trained the underlying policy is.

Chapter 4 documented the development process behind this method, including a series of correctness issues discovered in the benchmark evaluation harness, alongside one further correction needed to ensure a fair comparison against the classical baselines evaluated.

Chapters 5 and 6 demonstrated that the resulting policy matches the episode completion rate of an exact Google OR-Tools solver (99.8% vs. 97.0%) and substantially exceeds a greedy NN baseline (74.3%) across 1,000 dynamic episodes, while requiring roughly 9$\times$ less cumulative computation time than the exact solver and never losing a robot to battery failure. These results support this dissertation's central premise: native, mask based battery management allows a learned scheduling policy to match the reliability of classical exact methods in a dynamic, priority aware, multi-robot setting, without incurring their computational cost.

7.2 Future Work

Several directions naturally extend the work presented in this dissertation.

The priority weight used in the OR-Tools objective and the priority cost C_{pri} used to train the RL policy were both adjusted heuristically, in response to specific observed pathologies, rather than tuned systematically. A more thorough sweep of these values, for both the proposed method and the exact solver baseline, would allow a tighter comparison of each method's best achievable priority handling.

The architecture presented in Chapter 3 is specific to a two robot fleet operating on a unit square workspace with two fixed charging bases. Extending the state representation and context vector to support fleets of arbitrary size, heterogeneous battery capacities and charging rates across robots, and a single or larger number of shared charging bases would broaden the applicability of this approach considerably. Testing this generalisation would also allow direct evaluation of the hypothesis raised in Chapter 6: that the greedy baseline's relative advantage in makespan is partly an artefact of the density of the evaluated environment, and would narrow in larger or sparser settings where its lack of lookahead becomes more costly.

The environment presented in Chapter 3 assumes robots move at a constant speed between locations computed under the Euclidean metric, and a workspace free of obstacles or constrained paths. Incorporating a kinematic or path planning model, together with map topologies that include obstacles, would bring the environment closer to a physical deployment and is a natural precursor to validating the transition from simulation to physical AGV platforms.

This dissertation models battery capacity within a single episode and does not track degradation across repeated charge cycles. Extending the state representation to track battery health across multiple episodes is a promising direction for connecting the scheduling decisions made in this dissertation to the longer term operational costs of an industrial AMR fleet.

As an initial step in this direction, a preliminary interface was developed to connect the trained policy to ROS (Robot Operating System) topics, allowing scheduling decisions produced by the model to be published in a format consumable by a ROS based robot control stack. This interface remains a proof of concept: it has not been evaluated against a physical robot, or even within a ROS based simulator such as Gazebo. Extending this interface and validating it first in simulation and subsequently on physical hardware is a natural and comparatively well defined next step towards demonstrating this dissertation's approach beyond a purely simulated setting.

Taken together, these directions do not revise the central, structural contribution of this dissertation, battery safety can be guaranteed by construction within a learned scheduling policy, but extend its scope, efficiency and applicability beyond the two robot, single episode setting evaluated here.

References

[1] C. Rema, P. Costa, M. Silva, and E. J. S. Pires, "Task scheduling with mobile robots - a systematic literature review," *Robotics*, vol. 14, no. 6, p. 75, 2025.

[2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: W. H. Freeman, 1979.

[3] Y. Bengio, A. Lodi, and A. Prouvost, "Machine learning for combinatorial optimization: A methodological tour d'horizon," *European Journal of Operational Research*, vol. 290, no. 2, pp. 405–421, 2021.

[4] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[5] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takáč, "Reinforcement learning for solving the vehicle routing problem," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[6] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[7] J. Li, "Comparative study and hybrid modeling of vehicle routing problem and job shop scheduling problem," in *Recent Trends in Decision Science and Management*, ser. Advances in Intelligent Systems and Computing, T. S. Wang, A. Ip, M. Tavana, and V. Jain, Eds., vol. 1142, Singapore: Springer, 2020.

[8] L. He, R. Chiong, and W. Li, "Energy-efficient open-shop scheduling with multiple automated guided vehicles and deteriorating jobs," *Journal of Industrial Information Integration*, vol. 30, p. 100 387, 2022.

[9] A. Avhad, H. Arnarson, C. Schou, and O. Madsen, "Implementing swarm production system with multi-robot simulation," *Procedia Computer Science*, vol. 232, pp. 934–945, 2024.

[10] Q. V. Dang, T. Martagan, I. Adan, and J. Kleinlugtenbeld, "Order release strategies for a collaborative order picking system," in *Proceedings of the Winter Simulation Conference (WSC)*, Singapore, 2023, pp. 1521–1532.

[11] Z.-J. Liu, H.-Y. Sang, C.-Z. Zheng, H. Chi, K.-Z. Gao, and Y.-Y. Han, "An effective multi-restart iterated greedy algorithm for multi-AGVs dispatching problem in the matrix manufacturing workshop," *Expert Systems with Applications*, vol. 252, p. 124 223, 2024. DOI: 10.1016/j.eswa.2024.124223

[12] S. Yi and J. Luo, "Heuristic scheduling for robotic job shops using Petri nets and artificial potential fields," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 7556–7568, 2025. DOI: 10.1109/TASE.2024.3397386

[13] S. Bakshi, T. Feng, Z. Yan, and D. Chen, "A regularized quadratic programming approach to real-time scheduling of autonomous mobile robots in a prioritized task space," in *2019 American Control Conference (ACC)*, 2019, pp. 1361–1366. DOI: 10.23919/ACC.2019.8814986

[14] M. S. Kim et al., "A dynamic programming-based heuristic algorithm for a flexible job shop scheduling problem of a matrix system in automotive industry," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, Mexico City, Mexico, 2022, pp. 777–782.

[15] X. Chen and S. Liu, "Distributed cooperative task planning for autonomous mobile robots in intralogistics," in *Proceedings of the 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, Rome, Italy, 2023, pp. 870–875.

[16] N. Sahar, A. Basile, and B. Yan, "A tightened formulation for job shop scheduling with mobile robots," in *2024 IEEE 20th International Conference on Automation Science and Engineering (CASE)*, 2024, pp. 3237–3242. DOI: 10.1109/CASE59546.2024.10711558

[17] Y. Yao, Q.-H. Liu, X.-Y. Li, and L. Gao, "A novel MILP model for job shop scheduling problem with mobile robots," *Robotics and Computer-Integrated Manufacturing*, vol. 81, p. 102 506, 2023. DOI: 10.1016/j.rcim.2022.102506

[18] J.-E. Justkowiak, M. Y. Kovalyov, and E. Pesch, "A dynamic programming algorithm for order picking in robotic mobile fulfillment systems," *Networks*, vol. 84, pp. 481–490, 2024. DOI: 10.1002/net.22245

[19] S. Riazi and B. Lennartson, "Using CP/SMT solvers for scheduling and routing of AGVs in flexible manufacturing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 218–229, 2021. DOI: 10.1109/TASE.2020.3014156

[20] G.-B. Zhang, H.-B. Li, X.-T. Liu, and Y.-J. Peng, "Simulation budget allocation for improving scheduling and routing of automated guided vehicles in warehouse management," *Journal of the Operations Research Society of China*, vol. 13, pp. 775–809, 2025. DOI: 10.1007/s40305-024-00553-0

[21] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," in *Proceedings of the 30th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, 2021, pp. 1017–1023.

[22] J. K. Behrens, R. Lange, and M. Mansouri, "A constraint programming approach to simultaneous task allocation and motion scheduling for industrial dual-arm manipulation tasks," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8705–8711.

[23] L. Perron and V. Furnon. "OR-Tools," Google LLC. [Online]. Available: https://developers.google.com/optimization

[24] W. Li, H. Li, Y. Wang, and Y. Han, "Optimizing flexible job shop scheduling with automated guided vehicles using a multi-strategy-driven genetic algorithm," *Egyptian Informatics Journal*, vol. 25, p. 100 437, 2024. DOI: 10.1016/j.eij.2023.100437

[25] X. Han et al., "A dual population collaborative genetic algorithm for solving flexible job shop scheduling problem with AGV," *Swarm and Evolutionary Computation*, vol. 86, p. 101 538, 2024. DOI: 10.1016/j.swevo.2024.101538

[26] A. K. Cechinel and E. R. De Pieri, "Centralized multi-robot logistic system: An approach using the island model genetic algorithm as task scheduler," *International Journal of Advanced Robotic Systems*, vol. 21, 2024. DOI: 10.1177/17298806241279595

[27] S. M. J. Alzubairi, A. Petunin, and A. J. Humaidi, "Multi-robot task allocation based on an automatic clustering strategy employing an enhanced dynamic distributed PSO," *International Review of Applied Sciences and Engineering*, 2025. DOI: 10.1556/1848.2025.00935

[28] M. Mousavi, H. J. Yap, S. N. Musa, F. Tahriri, and S. Z. Md Dawal, "Multi-objective agv scheduling in an FMS using a hybrid of genetic algorithm and particle swarm optimization," *PLOS ONE*, vol. 12, no. 3, e0169817, 2017.

[29] Y. Cui, B. Jia, H. Sang, L. Meng, B. Zhang, and W. Zou, "An effective discrete Jaya algorithm for multi-AGVs scheduling problem with dynamic unloading time," *IEEE Access*, vol. 12, pp. 101 701–101 716, 2024. DOI: 10.1109/ACCESS.2024.3432594

[30] C. Sun, Y. Wang, F. Zhang, D. Li, Y. Tan, and J. Zhang, "Chaotic sparrow search algorithm based scheduling for flexible job shop with automatic guided vehicle," in *2023 4th International Conference on Information Science, Parallel and Distributed Systems (ISPDS)*, 2023, pp. 580–584. DOI: 10.1109/ISPDS58840.2023.10235475

[31] Y. Zheng, Y. Xiao, and Y. Seo, "A tabu search algorithm for simultaneous machine/AGV scheduling problem," *International Journal of Production Research*, vol. 52, no. 19, pp. 5748–5763, 2014.

[32] Q.-V. Dang, H. Rudová, and C. T. Nguyen, "Adaptive large neighborhood search for scheduling of mobile robots," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, 2019, pp. 224–232.

[33] Z. Pan, L. Wang, J. Zheng, J.-F. Chen, and X. Wang, "A learning-based multipopulation evolutionary optimization for flexible job shop scheduling problem with finite transportation resources," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1590–1603, 2023. DOI: 10.1109/TEVC.2022.3219238

[34] Z. Lin, P. Ding, and J. Li, "Task scheduling and path planning of multiple AGVs via cloud and edge computing," in *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2021, pp. 1–6. DOI: 10.1109/ICNSC52481.2021.9702191

[35] G. Xu, Q. Bao, and H. Zhang, "Multi-objective green scheduling of integrated flexible job shop and automated guided vehicles," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 106 864, 2023. DOI: 10.1016/j.engappai.2023.106864

[36] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. "Neural combinatorial optimization with reinforcement learning." arXiv preprint arXiv:1611.09940, 2016.

[37] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 1992.

[38] P. Durst, X. Jia, and L. Li, "Multi-objective optimization of AGV real-time scheduling based on deep reinforcement learning," in *Proceedings of the 2023 42nd Chinese Control Conference (CCC)*, 2023, pp. 5535–5540. DOI: 10.23919/CCC58697.2023.10240543

[39] M. P. Li, P. Sankaran, M. E. Kuhl, R. Ptucha, A. Ganguly, and A. Kwasinski, "Task selection by autonomous mobile robots in a warehouse using deep reinforcement learning," in *Proceedings of the Winter Simulation Conference (WSC)*, IEEE, 2020, pp. 680–688.

[40] M. Tejer, R. Szczepanski, and T. Tarczewski, "Robust and efficient task scheduling for robotics applications with reinforcement learning," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107 300, 2024. DOI: 10.1016/j.engappai.2023.107300

[41] B. Cheng, T. Xie, L. Wang, Q. Tan, and X. Cao, "Deep reinforcement learning driven cost minimization for batch order scheduling in robotic mobile fulfillment systems," *Expert Systems with Applications*, vol. 255, p. 124 589, 2024. DOI: 10.1016/j.eswa.2024.124589

[42] Y. Shida, T. Jimbo, T. Odashima, and T. Matsubara, "Reinforcement learning of multi-robot task allocation for multi-object transportation with infeasible tasks," in *2025 IEEE/SICE International Symposium on System Integration (SII)*, 2025. DOI: 10.1109/SII59315.2025.10870902

[43] Z. Zhang, X. Jiang, Z. Yang, S. Ma, J. Chen, and W. Sun, "Scalable multi-robot task allocation using graph deep reinforcement learning with graph normalization," *Electronics*, vol. 13, no. 8, p. 1561, 2024. DOI: 10.3390/electronics13081561

[44] M. Lai et al., "RoboBallet: Planning for multirobot reaching with graph neural networks and reinforcement learning," *Science Robotics*, vol. 10, no. 106, eads1204, 2025. DOI: 10.1126/scirobotics.ads1204

[45] M. Goarin and G. Loianno, "Graph neural network for decentralized multi-robot goal assignment," *IEEE Robotics and Automation Letters*, vol. 9, no. 5, pp. 4051–4058, 2024. DOI: 10.1109/LRA.2024.3371254

[46] T. M. Ho, K.-K. Nguyen, and M. Cheriet, "Federated deep reinforcement learning for task scheduling in heterogeneous autonomous robotic system," *IEEE Transactions on Automation Science and Engineering*, vol. 21, pp. 528–540, 2024. DOI: 10.1109/TASE.2022.3232291

[47] W. Gu, S. Liu, Z. Guo, M. Yuan, and F. Pei, "Dynamic scheduling mechanism for intelligent workshop with deep reinforcement learning method based on multi-agent system architecture," *Journal of Manufacturing Systems*, vol. 68, pp. 242–257, 2023. DOI: 10.1016/j.jmsy.2023.03.003

[48] Y. Zhao, K. Zhu, X. Song, and J. Zhang, "An AGV task scheduling method based on multi-agent reinforcement learning," in *Proceedings of the IEEE 12th Data Driven Control and Learning Systems Conference (DDCLS)*, Xiangtan, China, 2023, pp. 1504–1509.

[49] S. Geng, Y. Guo, S. Huang, and A. Sitahong, "Multi-agent deep reinforcement learning based integrated scheduling of machines and AGVs in discrete manufacturing workshop," in *2024 IEEE International Conference on Automatic Control and Intelligent Systems (I2CACIS)*, 2024, pp. 59–64. DOI: 10.1109/I2CACIS61270.2024.10704826

[50] A. Agrawal, S. Won, T. Sharma, M. Deshpande, and C. McComb, "A multi-agent reinforcement learning framework for intelligent manufacturing with autonomous mobile robots," in *Proceedings of the Design Society*, vol. 1, 2021, pp. 161–170. DOI: 10.1017/pds.2021.17

[51] A. Bolu and O. Korcak, "Adaptive task planning for multi-robot smart warehouse," *IEEE Access*, vol. 9, pp. 27 346–27 358, 2021. DOI: 10.1109/ACCESS.2021.3058190

[52] X. Dong, G. Wan, and P. Zeng, "Flexible job shop machines and AGVs cooperative scheduling on the basis of DQN algorithm," in *2024 IEEE 6th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, 2024, pp. 1808–1814. DOI: 10.1109/IMCEC59810.2024.10575538

[53] J. Zhang, Q. Jia, S. Zhang, and G. Chen, "Dynamic and prioritized task scheduling of heterogeneous multi-robot systems using deep reinforcement learning," *Neurocomputing*, vol. 638, p. 130 184, 2025.

[54] S. Lassoued, L. S. Baheti, N. Weiß-Borkowski, S. Lier, and A. Schwung, "Flexible manufacturing systems intralogistics: Dynamic optimization of AGVs and tool sharing using coloured-timed Petri nets and actor-critic RL with actions masking," *Journal of Manufacturing Systems*, vol. 82, pp. 405–419, 2025.

[55] T. Ito, R. Funada, M. Sampei, and G. Notomista, "Energy-aware task allocation for teams of multi-mode robots," *IEEE Robotics and Automation Letters*, 2025. DOI: 10.1109/LRA.2025.3566561

[56] 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. DOI: 10.1109/TMC.2023.3331573

[57] A. S. Chavan and M. Brocanelli, "Towards high-quality battery life for autonomous mobile robot fleets," in *Proceedings of the IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, 2022, pp. 61–70.

[58] J. Li, S. Li, J. Chu, W. Li, and D. Chen. "Fleet-level battery-health-aware scheduling for autonomous mobile robots." arXiv preprint arXiv:2603.22731, 2026.