symmetry

MDPI

Article

GRouteNet: A GNN-Based Model to Optimize Pathfinding and Smart Charging Management for Autonomous Guided Vehicles

Sadia Nishat Kazmi [1] and Syed Muhammad Abrar Akber [2,*]

[1] Department of Distributed Computing and Informatic Devices, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland; sadia.nishat.kazmi@polsl.pl

[2] Department of Computer Graphics, Vision and Digital Systems, Faculty of Automatic Control, Electronics and Computer Science, Silesian University of Technology, 44-100 Gliwice, Poland

* Correspondence: abrar.akber@polsl.pl

Abstract: Autonomous Guided Vehicles (AGVs) play an important role in the automation of material handling and transportation tasks in modern industrial and logistics systems. However, suboptimal path planning and longer waiting times at charging stations significantly affect the operational efficiency of these vehicles. To address these challenges, we leverage the capabilities of Graph Neural Networks (GNN) to find the optimal paths for AGVs. In this paper, we propose GRouteNet, a GNN-based model that effectively finds the shortest path for AGVs. The proposed model utilizes the message-passing mechanism of GNN to determine the neighbor nodes and then aggregates this information to find the shortest path. We compare the results of GRouteNet with a couple of existing state-of-the-art pathfinding models and show that the path length computed by GRouteNet is up to 45% shorter compared to the existing models. Furthermore, we propose a Shortest Charging Time First (SCTF) scheduling algorithm to reduce the long waiting times in the queues at the charging stations. The proposed SCTF algorithm prioritizes the charging of AGVs based on their charging time and charges the AGVs with the shortest charging time first. We compare the results of SCTF with the first-come-first-serve scheduling algorithm and show that SCTF reduces the waiting times at the charging stations by up to 42%.

Keywords: automated guided vehicles; graph neural networks; pathfinding; scheduling; shortest path

check for updates

Citation: Kazmi, S.N.; Akber, S.M.A. GRouteNet: A GNN-Based Model to Optimize Pathfinding and Smart Charging Management for Autonomous Guided Vehicles. *Symmetry* **2024**, *16*, 1573. https://doi.org/10.3390/sym16121573

Academic Editor: Zhixun Su

Received: 6 October 2024
Revised: 10 November 2024
Accepted: 20 November 2024
Published: 25 November 2024

1. Introduction

In the current modern world, the proliferation of Information and Communication Technologies (ICTs) is reshaping every aspect of human life. Over the past decade, we have witnessed a steady growth in the world's population, which has led to accelerated urbanization [1]. This increasing urbanization results in an increased demand for vehicles [2]. Vehicle manufacturers are producing more advanced vehicles by taking advantages of advances in ICTs [3]. Connected and autonomous vehicles (CAVs) equipped with ICTs are capable of exchanging data with other vehicles, infrastructure and the cloud. The integration of ICTs increases the level of safety, optimizes traffic flow and enables advanced functions such as real-time navigation updates [4].

Automatic Guided Vehicles (AGVs) are a special form of CAVs, considering that their common goal is to achieve autonomy. AGVs are very effective and can efficiently perform a variety of tasks such as material handling and customized transportation in modern industrial environments [5]. In the post-COVID-19 era, modern intelligent manufacturing facilities make extensive use of ICTs and AGVs to optimize industrial operations. However, AGVs face several challenges in performing their tasks efficiently. These challenges include optimized path planning, battery optimization and collision avoidance [6].

Automotive manufacturers like Ford and Toyota have integrated AGVs in their assembly plants for transporting materials between lines, optimizing workflows to support just-in-time manufacturing. This improvement has reduced production cycle times, with

Ford reporting efficiency gains of up to 20% by minimizing bottlenecks and improving material flow [7]. Amazon has used more than 750,000 robots, including AGVs, in its fulfillment facilities to optimize logistics, decreasing trip times and enhancing operational throughput [8].

Path planning is one of the most important challenges for AGVs, as most other challenges are directly related to path planning [9,10]. Optimizing the path of AGVs can lead to increased operational time, optimized travel times, reduced delays and lower energy consumption, which in turn increases the efficiency and overall performance of AGVs. In addition, well-optimized routes reduce the possibility of collisions and thus avoid operational inefficiencies. Planning an optimized route for AGV operations is indeed a challenging task as several parameters need to be considered, such as the dynamic operating environment of multiple AGVs, avoiding collisions with other AGVs, humans, and obstacles, etc. Addressing these challenges is vital to improve the efficiency and safety of AGVs.

Anticipating the importance of optimizing paths for AGVs, different approaches are used. Dijkstra's algorithm or the A* (A-star) algorithm are very popular and are widely used for finding the shortest paths [11–13]. These traditional pathfinding algorithms effectively provide predictable and deterministic paths for AGVs. However, these algorithms assume static operating environments and do not take into account the dynamic changes in the environment, such as avoiding congested routes. The wide popularity of artificial intelligence (AI) [14,15] in the recent past encourages its use in the transportation sector too [15–17]. Graph Neural Network (GNN) is one such type of architecture that can be employed to solve the challenges of optimizing routes for vehicles. GNN can quickly and effectively traverse the graph structure and find the shortest path between the nodes. The operating environment of AGVs can be modeled as a connected graph, and GNN can be applied for finding the optimal path between the nodes in the graph. The architecture of GNN enables it to detect local alterations in the graph (e.g., a pathway obstruction or the entry of another AGV) and incorporate this information as it disseminates through the network, facilitating real-time path adjustments. This allows GNN-based models to sustain optimal and effective route planning, even in uncertain environments, which is essential for preserving high operational efficiency and minimizing downtime in such contexts.

Downtime due to charging the batteries of AGVs is another important factor that affects the availability of such vehicles for logistics tasks. All AGVs needs to recharge their batteries at designated charging stations. The waiting times at the charging stations may become a bottleneck and severely affect the up-time of the AGVs. In addition to the optimizing the path planning, the waiting time for charging AGVs is another challenge that needs to be taken into account to maximize operational efficiency.

Symmetry enables a balanced and effective operating environment to maximize the availability of AGVs by optimizing their paths and charging schedule. When planning routes, symmetry ensures that all AGVs have equal access to the optimized routes, minimizes travel times and promotes an equitable distribution of routes. This also applied for charging schedules, where AGVs are prioritized according to their battery status to reduce idle times. Symmetry contributes to a more efficient and coherent system by minimizing delays, maximizing the availability of AGVs and optimizing movement and resource allocation.

In this paper, we propose a GNN-based model, which we call GRouteNet, to leverage the strong capabilities of GNNs to traverse the graph structures for finding the shortest path for operating AGVs. We model the operating environment of AGVs as a graph structure so that the GNNs can be implemented to find the shortest path. The proposed GRouteNet implements the GNNs, especially those using Graph Convolutional Network (GCN) layers, to find the shortest path between the source and destination nodes. We select GCNs for GRouteNet because of their robust ability to capture spatial connections in graph-structured data, which is crucial for pathfinding tasks. In contrast to other GNN types, GCNs effectively consolidate input from adjacent nodes, enabling the model to construct a comprehensive perspective of the network and adjust to changing conditions.

In addition, we design a Shortest Charging Time First (SCTF) scheduling algorithm to reduce waiting times of AGVs in queues at charging stations. We propose to prioritize the charging of AGVs based on their charging time and to charge the AGVs that have a shorter charging time first. We compare the results of GRouteNet with existing similar work and quantitatively evaluate the effectiveness of our proposed GRouteNet. The specific contributions are as follows;

- Propose GRouteNet, a GNN-based model to find the shortest path for the operation of AGVs;
- Propose a Shortest Charging Time First scheduling algorithm to reduce the waiting times of AGVs in charging queue;
- Implement the proposed model and conduct experiments;
- Evaluate the performance of the proposed model by comparing it with existing similar work.

The rest of the paper is organized as follows: Section 2 provides an insight into the existing literature review of the domain. Section 3 briefly describes the proposed approach, its working and the pseudo code. Section 4 presents the details of the experiment, dataset, methodology and the obtained results along with their evaluation. Lastly, Section 5 concludes the paper.

2. Literature Review

We organize this section into two subsections, firstly, we review the pathfinding approaches for AGVs, followed by their charging management.

2.1. Pathfinding for AGVs

Finding the shortest paths for AGVs is of key importance for the effective operation of these vehicles. Anticipating its importance, lots of research efforts are underway that aim to find the shortest paths to extend the operating times of AGVs. This section provides a brief overview of such existing techniques.

Chen et al. [18] propose Rf-Ant-agent, a routing system based on repulsive potential field. Rf-Ant-agent combines centralized and decentralized control with ant-agent optimization. An ant-agent algorithm with a repulsive potential field is combined with Rf-Ant-agent to find the optimized path. The proposed Rf-Ant-agent is designed to optimize the path selection while reducing the node congestion. The Rf-Ant-agent initially sets up the operational environment and discovers the obstacles and free paths. The optimized paths are then determined by using the optimized ant agent algorithm. It uses pheromone trails to represent congestion levels, while repulsion fields are used to avoid collisions. The proposed approach ensures that the AGVs may instantly and dynamically adjust their paths in response to environmental changes.

Tang et al. [11] use a geometric A-star algorithm for path planning of AGVs. The proposed technique integrate cubic B-spline interpolation with the traditional A-star algorithm. The proposed algorithm unitize a simple grid-like structure and identifies the irregular routes by filtering the nodes using the $P(x,y)$ and $W(x,y)$ functions. The algorithm swaps the polyline paths with cubic B-spline curves to ensure smooth navigation and stability of the AGV in curves. Chen et al. [5] utilize the ensemble learning approach for path planning for AGVs. The proposed approach organizes the scheduling tasks by identifying the source and destination in the port environment. The authors integrate Twin Delayed Deep Deterministic Policy Gradient (TD3) and artificial potential field (APF) frameworks and propose the approach for pathfinding for AGVs. The APF framework makes use of barrier locations to produce repulsive forces while the TD3 algorithms exploit these forces and optimize and converge the path planning via a double-Q network. The TD3 framework in the proposed approach utilizes a reward-based mechanism to continuous learn and find the finds the optimal paths.

Shuo et al. [19] present PF-DDQN, a path planning technique that combines particle filters (PF) and reinforcement learning (RL). The proposed path planning technique repeatedly update the neural network weights using the PF-DDQN approach by considering them as state variables. The optimization of the weights of the neural network leads to a higher efficiency of path planning. PF-DDQN operates in two stages: (i) network training and (ii) network weight updates. In the training phase, the environmental noises are modeled together with the target network and their weights as state variables. Afterwards, the PF algorithm adjusts these state variables. This process leads to an improvement in convergence rate and efficiency in finding the shortest paths for multi-AGV environments.

Lin et al. [20] presents a hybrid evolutionary approach using cultural-particle swarm optimization (C-PSO) to enhance path planning for multiple AGVs in industrial warehouses. This technique aims to achieve task allocation, fault tolerance, and collision avoidance through a dual-layer framework that combines cultural and particle swarm optimization elements. The probabilistic approach within C-PSO updates the search strategy dynamically, improving solution quality while maintaining low iteration counts. Although effective in generating optimal paths quickly, the method's complexity may lead to increased computational demands, unlike simpler algorithms like SCTF, which specifically optimizes queue management and reduces waiting times in AGV charging scenarios.

Fusic et al. [21] propose to use satellite images for calculating collision-free routes for AGVs. The authors construct an occupancy grids from the satellite images representing obstacles in which and free spaces in black color. The clarity of the obstacles is enhanced by processing satellite images so that the AGVs can effectively identify and plan optimal paths. For the processing of these images, various filters such as Sobel, Canny and binary occupancy grids are used. Fusic et al. implement particle swarm optimization (PSO) to ensure optimal pathfinding with a reduced computational cost.

Vlachos et al. [22] explores the integration of AGVs and IoT to create adaptive manufacturing environments aligned with Industry 4.0. It presents a case study demonstrating how AGVs, enhanced with IoT connectivity, improve flexible manufacturing systems (FMS) by optimizing warehousing, routing, and human-machine interfaces. Using a socio-technical systems approach, the study identifies AGV impacts across design, operational, and control levels, highlighting the potential for improved efficiency, reduced human errors, and adaptive capabilities in dynamic production settings. We present a summary of the literature review on pathfinding for AGVs in Table 1.

Table 1. Summary of the literature review on pathfinding for AGVs.

Reference	Approach	Application	Architecture	Limitations
Chen et al. [5]	Reinforcement Learning	Autonomous port management for AGV path planning	Ensemble RL model for optimizing path and task allocation	Higher computational complexity and limited to specific environments
Zhang et al. [1]	Energy-efficient Path Planning	Manufacturing workshops for multi-load AGV	Multi-load AGV path optimization based on energy efficiency	Less flexible for varied environments and may struggle with multi-objective optimization
Tang et al. [11]	Geometric A* Algorithm	Port environments	Geometric modifications to A* for port pathfinding	specific to static layouts and less efficient in real-time scenarios
Chen et al. [18]	Ant-Agent Optimization	Multi-AGV path planning	Ant colony optimization tailored to AGV routing	Limited by convergence speed and adaptability in dynamic tasks
Shuo et al. [5]	RL and Particle Filters	Multi-AGV environments	Combines RL with particle filters for path planning	Higher complexity and slower convergence

Table 1. *Cont.*

Reference	Approach	Application	Architecture	Limitations
Fusic et al. [21]	Heuristic Path Planning	Outdoor AGV navigation	Heuristic model for outdoor path optimization	Not ideal for indoor structured layouts and lacks adaptability in congested areas
Vlachos et al. [22]	AGV with IoT Integration	Flexible manufacturing systems	Integrates IoT with AGVs for smart manufacturing	prone to network issues and less efficient without robust connectivity
Lin et al. [20]	C-PSO for Fault Tolerance	Multi-AGV warehouse environments	Hybrid Cultural-PSO for path optimization with fault tolerance	High computation time due to bio-inspired complexities

2.2. Charging Management for AGVs

Chen et al. [23] focus on reducing waiting times for AGVs at charging stations and propose a model based on queuing theory. Mehta et al. [24] propose a forecasting model that takes into account both the AGV demand patterns and the predicted availability of renewable energy. The proposed model reduces waiting times and provides practical insights for the sustainable operation of AGVs. Liu et al. [25] propose a multi-criteria optimization model for AGV charging scheduling considering both charging time and energy consumption. The proposed approach achieved a significant increase in energy efficiency while maintaining shorter waiting times at the charging stations. Zhang and Wang [26] employ reinforcement learning to propose a dynamic charging scheduling algorithm for AGVs. The proposed scheduling algorithm adapts to the workload and adjusts to the real-time fluctuations of AGV battery levels. The proposed method reduces the total waiting times by 15% compared to static scheduling approaches.

Cheng et al. [27] presents a Twin Delayed Deep Deterministic Policy Gradient (TD3) model aimed at enabling AGVs and vehicles to navigate complex environments based solely on local information. By employing a virtual obstacle mechanism and an actor-critic structure, the TD3 model achieves adaptability in real-time pathfinding without relying on global maps. Che et al. [28] introduces a decentralized, multi-agent deep reinforcement learning (DRL) approach using an actor-critic framework to optimize AGV scheduling at container terminals. The study models the recharging and scheduling of electric AGVs (EAGVs) as a partially observable Markov decision process (Dec-POMDP) to tackle constraints like limited charging station capacity and uncertain vehicle travel times. Employing heterogeneous graph neural networks (HetGNNs) and a multi-agent proximal policy optimization (MAPPO) algorithm, the model allows vehicles to make independent recharging and job-scheduling decisions that collectively minimize makespan.

Liu et al. [29] proposes a dynamic pricing strategy for charging station alliances (CSAs) using a bidirectional real-time communication framework. The strategy leverages evolutionary game theory to optimize charging prices while accounting for information asymmetry and bounded rationality among electric vehicle (EV) users. By modeling charging stations as alliances rather than isolated units, the approach aims to increase CSA profitability by dynamically adjusting prices based on real-time EV status and market conditions. This framework, validated through a case study, improves pricing accuracy and decision-making efficiency. We present a summary of the literature review on pathfinding for AGVs in Table 2.

Table 2. Literature review on charging management for AGVs.

Reference	**Model**	**Application**	**Approach**	**Limitations**
Chen et al. [23]	Queuing theory model	AGV charging station scheduling	Optimizes queuing at charging stations to minimize wait times	Limited adaptability to real-time fluctuations
Mehta et al. [24]	Predictive model with renewable integration	AGV charging with renewable energy	Integrates renewable sources for cost-effective charging	Dependency on renewable energy availability
Liu et al. [30]	Multi-objective optimization	Smart factories	Multi-objective approach for energy-efficient scheduling	May struggle with real-time prioritization
Wang et al. [26]	Reinforcement learning	Dynamic AGV charging scheduling	Dynamic adjustments using RL for adaptive charging	Computationally intensive
Che et al. [28]	Multi-agent DRL with HetGNN	Container terminals	DRL-based with graph neural networks for dynamic decision-making	High computational demand and complexity
Liu et al. [29]	Dynamic pricing and queuing in charging alliances	EV charging alliances	Evolutionary game theory for dynamic pricing and resource allocation	Focused on EVs, not AGVs
Cheng et al. [27]	TD3-based deep reinforcement learning	Autonomous navigation in ITS	TD3 model with virtual obstacle mechanism for complex navigation	Primarily focused on path planning; lacks queuing and prioritization for charging management

3. Proposed Model

This section describes the details of the proposed model and its working. The modelling, pseudocode and working of both GRouteNet and the scheduling algorithm are briefly described in this section.

3.1. Overview of the Proposed GRouteNet

To manage material handling and logistic operations in modern manufacturing warehouses, AGVs are vital to autonomously transferring items across dynamic environments. However, AGVs face challenges in navigating optimal paths within dynamically changing environments. The operating environments may dynamically change due to sudden inclusion of obstacles such as human passing, an abrupt fall of material or other AGVs, etc. GRouteNet is proposed to tackle such obstacles by providing a system that combines real-time path optimization with intelligent charge scheduling.

The proposed GRouteNet aims to address the challenge of finding optimal paths for AGVs, so that extended operational times for such automated vehicles are ensured. GRouteNet leverages the well-known GNN architecture to find the shortest path between a pair of nodes. The reason for choosing the GNN-based architecture for GRouteNet is its inherent ability to process graph-structured data. GRouteNet models the operational road networks for AGVs as a connected graph and applies GNN to find the shortest path between the source and destination nodes. GRouteNet uses a message-passing mechanism among the nodes in the graph and aggregates messages from the shortest path neighborhoods. GRouteNet consists of three main components (1) GCN layers (2) activation function and (3) training process. GCN layers convert the input data into hidden layers and encapsulate the neighbor nodes information. In our proposed GRouteNet, we use the ReLU activation function and in the training phase, the model learns by synthesizing data by identifying relations between nodes.

To address the issue of longer waiting time at the charging station, this work also proposes a scheduling algorithm that prioritizes AGVs by considering the charging time and their arrival times at the charging station. The algorithm set a higher priority to those AGVs that require the least charging time. This approach leads to a reduction in the waiting time in the charging queue.

3.2. System Model for GRouteNet

In a large-scale warehouse, AGVs need to navigate routes that may suddenly become congested due to other AGVs, human workers, or temporary obstacles like loading equipment. Traditional pathfinding algorithms such as the Dijkstra algorithm struggle to find optimal routes in such situations. GRouteNet addresses this limitation by using GNNs and a message-passing mechanism to enable AGVs to make real-time path decisions. We provide the description of notions used through the paper in Table 3.

Table 3. Notions and descriptions.

Notions	Descriptions
$G = (V, E)$	Graph to represent the operating environment of AGVs
V	Set of nodes in graph
E	Edges in the graph
$\mathcal{N}(v)$	Set of neighboring nodes of v
d_v	Degrees of node v
ω	Activation function
$\lambda^{(l)}_{u\to v}$	Message from node u to node v at layer l
μ_{uv}	Weight to represent the distance between the nodes u and v
$\hat{d}_{uv}$	Length of shortest paths between nodes u and v
$\mathbf{\Pi}^{(l+1)}_v$	Output feature vector of node v at layer $l+1$

To find the shortest path between the source node and the destination for the AGVs, we model the road network of AGVs as a graph $G = (V, E)$, where V is the set of nodes (representing roads and intersections, etc.) and E are the edges (to show the connection between the roads). Each node $v \in V$ has a feature vector $\mathbf{x}_v \in \mathbb{R}^d$, where d is the dimension of the feature vector. Each edge $e = (u, v) \in E$ has an associated weight μ_{uv}, which typically represents the distance or travel cost between the nodes u and v.

The GCN layers in GNN perform convolution operations over the given graph structure. We present the output feature vector $\mathbf{\Pi}^{(l+1)}_v$ of node v at layer $l+1$ in Equation (1) [31,32].

$$\mathbf{\Pi}^{(l+1)}_v = \omega\left(\sum_{u\in\mathcal{N}(v)\cup\{v\}} \frac{1}{\sqrt{d_u d_v}}\mu^{(l)}\mathbf{\Pi}^{(l)}_u\right) \tag{1}$$

where $\mathcal{N}(v)$ is the set of neighboring nodes of v, d_u and d_v are the degrees of nodes u and v, where $\mu^{(l)}$ represents the weight matrix of layer l, and ω is the activation function, respectively.

While navigating, the AGV employs the message-passing mechanism of the GNN to acquire real-time data from adjacent nodes on route conditions. This allows the AGV to modify its route in real-time in response to congestion or obstructions, guaranteeing it adheres to the most efficient and least obstructed path to its destination. In GNN, each node employ a message-passing mechanism for getting and aggregating the information about its neighboring nodes. We model the message passing mechanism in our model in Equation (2) [31,32]. We also present the message passing mechanism in Figure 1.

$$\lambda^{(l)}_{u\to v} = \frac{1}{\sqrt{d_u d_v}}\mu^{(l)}\mathbf{\Pi}^{(l)}_u \tag{2}$$

where $\lambda_{u \to v}^{(l)}$ represents message from node u to node v at layer l. The updated feature representation for node v after including the message passing mechanism is depicted in Equation (3).

$$\mathbf{\Pi}_v^{(l+1)} = \omega \left(\sum_{u \in \mathcal{N}(v) \cup \{v\}} \lambda_{u \to v}^{(l)} \right) \tag{3}$$

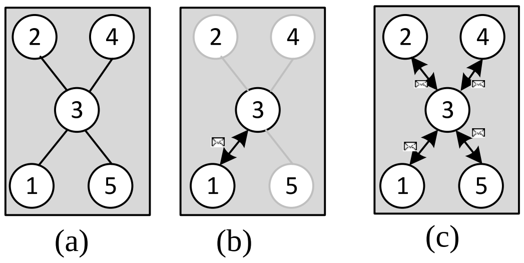


Figure 1. Message-passing mechanism to identify neighboring nodes, 1 to 5 are representative nodes in the graph, lines represent the connection between nodes, and the arrow shows the flow of message passing (**a**) graph representation of underlying model, (**b**) initiating message passing (**c**) neighboring discovery.

The proposed model finds the shortest path by employing $\mathbf{\Pi}_v^{(l+1)}$ from Equation (3), Let $\hat{d}_{uv}$ be the lengths of shortest paths between nodes u and v. We present the equation for the shortest path in Equation (4).

$$\hat{d}_{uv} = f(\mathbf{\Pi}_u^{(L)}, \mathbf{\Pi}_v^{(L)}) \tag{4}$$

We present our objective functions in Equation (5)

$$\min \sum_{(u,v) \in T} \hat{d}_{uv} \tag{5}$$

3.3. Working of GRouteNet

We present the working of our proposed GRouteNet using pseudo code in Algorithm 1.

The proposed GRouteNet utilizes the capabilities of GNN and GCN to find the shortest path. The algorithm takes the graph data as input and sets the hyperparameters. It then initializes the adjacency matrix in adj and the weight matrices for each GCN layer. The GRouteNet establishes the information about the neighboring nodes through the message passing mechanism and aggregates neighboring node information using the adjacency matrix. It iteratively updates the node characteristics over multiple GCN layers and learns the neighborhood information.

After the message-passing mechanism, the algorithm calculates the shortest path based on the aggregated neighborhood information of the nodes in the graph. The distances for directly connected nodes are initialized based on the edge weights. The proposed GRouteNet then iteratively calculates the distance matrix by determining for each pair of nodes i and j whether there is a shorter path via an intermediate node k. If GRouteNet successfully finds a shorter path, it updates the distance matrix.

Algorithm 1 GRouteNet Algorithm for Finding the Shortest Path

```
1: Input: Graph data (nodes, edges, features), Hyperparameters
2: Output: Shortest path predictions
3: Initialize parameters
4: $h \leftarrow$ node features (data.x)
5: $adj \leftarrow$ adjacency matrix (data.edge_index)
6: $W \leftarrow$ weight matrices for each GCN layer
7: for each GCN layer do
8:     Message Passing:
9:     Aggregate neighbor information: $h \leftarrow adj \cdot h$
10:    Apply linear transformation: $h \leftarrow h \cdot W$
11:    Apply non-linearity: $h \leftarrow \text{ReLU}(h)$
12: end for
13: Initialize distance matrix $dist[i][j] \leftarrow$ weight of edge $(i, j)$
14: for each source node $src$ in source_nodes do
15:     for each destination node $dest$ in destination_nodes do
16:         path_length ← model.predict_shortest_path(data, src, dest)
17:         Append $(src, dest, path\_length)$ to shortest_paths list
18:     end for
19: end for
20: for all nodes $i, j, k$ do
21:     $dist[i][j] \leftarrow \min(dist[i][j], dist[i][k] + dist[k][j])$
22: end for
```

3.4. Working of Scheduling Algorithm for Charging

In the operating environment of AGVs, which contains multiple AGVs with different battery capacities, AGVs perform various logistical tasks, so their batteries are exhausted and need to be recharged. The AGVs must visit specific charging stations to recharge their batteries. The AGVs arrive at the charging stations at a random time and may wait in a queue if there is already an AGV at the charging station. This can lead to longer waiting times in the queue at the charging stations and ultimately reduce operational efficiency. To address this issue, we propose a scheduling technique called Shortest Charge Time First (SCTF), which prioritizes AGVs based on their charging time and arrival times at the charging station.

The proposed scheduling algorithm sets a higher priority to those AGVs that need a shorter charging time. Prioritizing AGVs depending on their charging times can lead to a reduction in the waiting time in queues. Whenever a new AGV arrives at the charging station, the AGV queue is updated. By allocating charging resources in a way that considers the shortest charging time, the SCTF scheduling algorithm effectively reduces the overall waiting time.

Let T_i be the required charging time for AGV, A_i the arrival time of AGV i, C_i the remaining battery capacity that must be charged for AGV i, and R the charging rate of the charging station. The charging time T_i for each AGV is calculated as in Equation (6).

$$T_i = \frac{C_i}{R} \tag{6}$$

The scheduling goal is to minimize the total waiting time W across all AGVs, where W_i is the waiting time for AGV i. This is formulated as in Equation (7).

$$W_i = \sum_{j=1}^{i-1} T_j \quad \text{for all AGVs charged before AGV } i \tag{7}$$

Thus, the objective is to dynamically prioritize AGVs by minimizing their charging times and accounting for their arrival times as depicted in Equation (8).

$$\text{Minimize} \quad \sum_{i=1}^{N} W_i \tag{8}$$

where N is the total number of AGVs in the system.

The SCTF scheduling algorithm, presented as Algorithm 2, is executed by prioritizing the AGVs, taking into account their charging times and arrival times at the charging stations. First, an empty queue is created for the AGVs waiting to be charged. For each arriving AGV, the SCTF algorithm determines the required charging time by dividing the remaining battery capacity by the charging rate. As new AGVs arrive, the queue is continuously updated with the newest AGVs and the queue is sorted by shortest charging time first and earliest arrival time second. The AGV with the shortest loading time and earliest arrival is then assigned a charging station, and once charging is complete, the AGV is removed from the queue. SCTF ensures that AGVs with shorter charging needs are charged quickly, minimizing the overall waiting time and ensuring optimal use of charging resources in real time. We present the flow chart for the charging management of AGVs at charging stations in Figure 2.

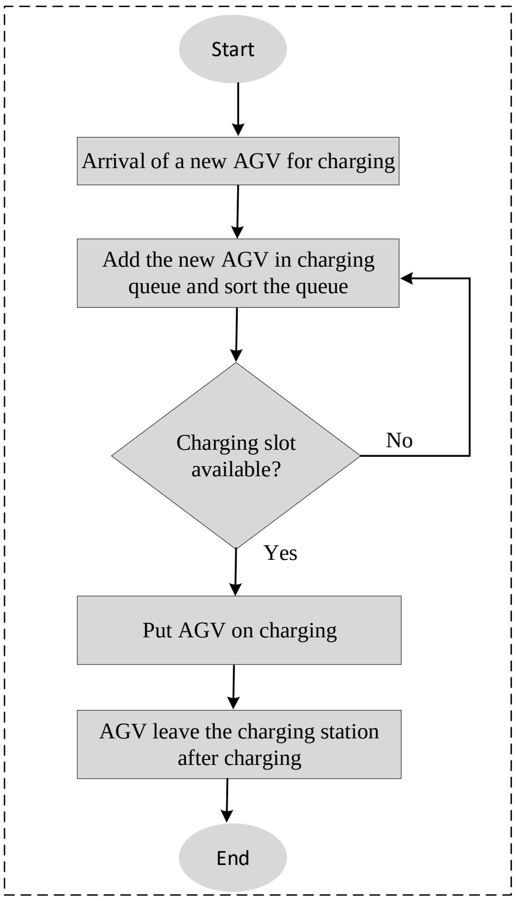

Start
Arrival of a new AGV for charging
Add the new AGV in charging queue and sort the queue
Charging slot available?
No
Yes
Put AGV on charging
AGV leave the charging station after charging
End

Figure 2. Flow chart for the charging management of AGVs at charging stations.

Algorithm 2 Shortest Charge Time First (SCTF) Scheduling Algorithm

```
1:  **Input:** Arrival times of AGV $A$, battery capacities $C$, and charging rate $R$
2:  **Output:** Minimize total waiting time
3:  AGVs_queue
4:  for each AGV $i$ arriving at the station do
5:      Calculate charging time $T[i] = \frac{C[i]}{R}$
6:      Insert AGV $i$ into AGVs_queue
7:  end for
8:  Continuously monitor the arrival of new AGVs
9:  while AGVs_queue is not empty do
10:     Update AGVs_queue with new arrivals and their charging times
11:     Sort AGVs_queue by $T[i]$ (ascending) and then by $A[i]$ (ascending)
12:     Select AGV $i$ with the smallest $T[i]$ and earliest $A[i]$
13:     Assign charging station to AGV $i$
14:     Remove AGV $i$ from AGVs_queue once charged
15: end while
```

4. Experiment, Results and Evaluation

This section describes the experiments, obtained results and their evaluation for the proposed model. We first present the experiment environment.

4.1. Experiment Environment

We conduct practical experiments to verify the working of the proposed GRouteNet. All experiments are conducted on a PC with Intel core i7 processor with 16 GB RAM. We use Python 3.7.9 (64-bit version) and VS Code version 1.91.1 as the IDE. The detailed environment setup is presented in Table 4.

Table 4. Experimental setup.

Description	Details
Hardware	- CPU : Intel(R) core(TM) i7 920 @ 2.67 GHz - RAM: 16 GB - OS: Windows 10 (64 bits) - Graphics Card: NVIDIA GeForce GT 440
Software	- Python 3.7.9 (64 bits) - PyTorch 1.13.1 - VS Code 1.91.1

4.2. Dataset and Methodology

In our experiment to find the shortest path, we use a synthetic dataset consisting of node positions that represent 2D coordinates and are stored in a csv file. This dataset is used to create a graph where each node represents a position and the edges are formed based on the Euclidean distance between the nodes. The dataset contains a total of 1224 edges connecting different nodes. To implement GRouteNet, we first construct the graph from the node positions and convert it into a geometric PyTorch format, including edge indices and edge attributes. Our model consists of three GCNConv layers with different dimensions to capture the spatial relationships between the nodes. We initialize the model and train it over 200 epochs. During training, we use a feed-forward mechanism where the node features are iteratively updated by the GCN layers. Finally, we evaluate the model by finding the shortest paths between a set of source and destination nodes using the output of the GNN to compute the path cost.

For the experiments with the SCTF algorithm, we allow all AGVs to arrive at the charging station when their battery needs to be charged. We conduct the experiments in such a way that all AGVs must visited three times at the charging stations and recharge their batteries. We implement the SCTF algorithm and record the waiting time of each AGV at each visit to the charging station. To evaluate the SCTF algorithm, we compare the result with the first-come-first-serve approach. We present the hyperparameters and simulation setup in Table 5.

Table 5. Experiment parameters.

Model	Hyperparameter	Value
Hyperparameters	Layer 1	GCNConv(2, 16)
	Layer 2	GCNConv(16, 32)
	Layer 3	GCNConv(32, 1)
	Learning Rate	0.1
	Optimizer	Adam
Simulation parameters	Grid size	500 × 500
	Total nodes	50
	Source nodes	3
	Destination nodes	47
	No of AGVs	6

The hyperparameters for GRouteNet were selected to effectively improve pathfinding and charging operations for AGVs. The GCNConv layer arrangement (Layer 1: 2 to 16, Layer 2: 16 to 32, Layer 3: 32 to 1) facilitates the incremental refining of spatial connections within the graph, with each layer adeptly collecting and compressing neighborhood information. A learning rate of 0.1 was established to guarantee steady and efficient learning, while the Adam optimizer adjusts the learning rate according to gradient variations, rendering it suitable for high-dimensional data. These selections provide precise pathfinding with little computing expenses, rendering GRouteNet appropriate for real-time AGV operations in dynamic settings.

4.3. Results and Evaluations

In this subsection, we describe our obtained results for finding the shortest path and charging schedule. Figure 3 mimics the operating environment for AGVs and presents the details of node deployment in the specified operational area for AGVs. It can be seen from Figure 3 that 50 nodes are randomly deployed in the operating area. Once the nodes are deployed, we select three nodes (randomly) to sever as the source nodes and the remaining nodes as the destination nodes. After the node deployment, we execute the GRouteNet, PF-DDQN, and Dijkastra algorithms for finding the shortest paths between the source nodes and each destinations node. We show the calculated shortest paths for PF-DDQN, Dijkstra, and GRouteNet in Figures 4a, 4b and 4c, respectively.

After the execution of all algorithms and obtaining the results, we perform a comparative analysis. For a comprehensive evaluation, we record the results for the shortest paths from each source to the destination for all models separately. We present the results for the shortest path calculated from each source node for all three models in Figure 5. The results in Figure 5a–c show the comparison between the shortest path computed by GRouteNet and the existing approach of source node 1, source node 2 and source node 3, respectively. GRouteNet consistently generates shorter paths with reduced fluctuation among various path IDs compared to other approaches, showing its ability to identify more optimum and stable routes. PF-DDQN and Dijkstra exhibit significant variations in path length, particularly from Source 2 and Source 3, but GRouteNet has a consistent decreasing trend, signifying its efficacy in minimizing path lengths across many contexts.

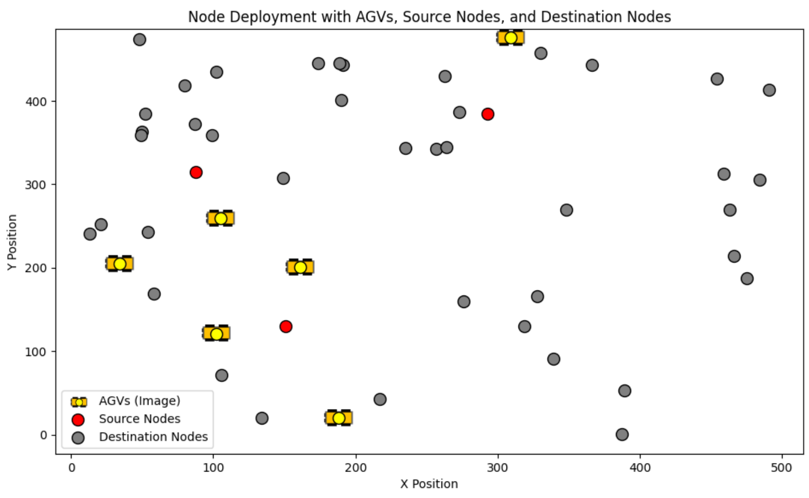
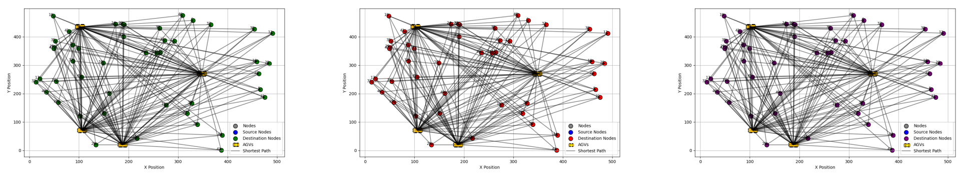
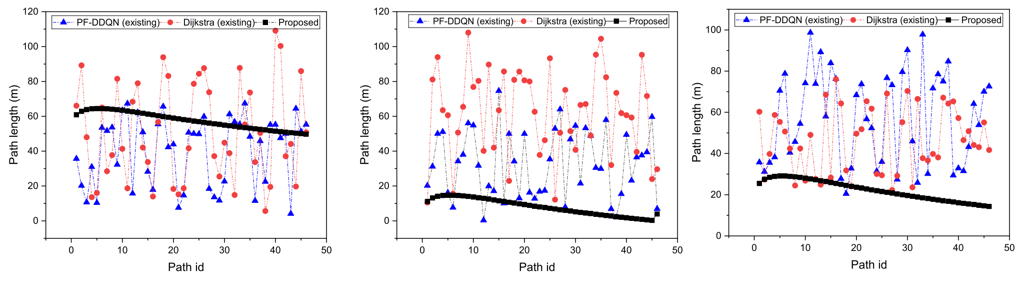


Figure 3. Node deployment.



(**a**) Shortest path for PF-DDQN (**b**) Shortest path for Dijkstra (**c**) Shortest path for GRouteNet

Figure 4. Finding shortest paths.



(**a**) Source 1 (**b**) Source 2 (**c**) Source 3

Figure 5. Comparison of shortest paths from all sources. (**a**) Shortest paths for each task for source 1, (**b**) shortest paths for each task for source 2, and (**c**) shortest paths for each task for source 3.

We conclude our comparison for the shortest path by presenting an overall comparison of the results aggregated from all sources in Figure 6. The values shown in Figure 6a show the aggregated values of the shortest path calculated from each source. It can be seen that the calculated values for the shortest path of source 1 are higher for GRouteNet. However, for source node 2 and source node 3, GRouteNet effectively reduces the length of the shortest paths compared to the existing approach.

The analysis of the initial values being slightly higher in GRouteNet is due to its dependency on local information at the beginning of the computation. As GRouteNet aggregates neighboring node information iteratively, it needs several passes to accurately compute the shortest path. The algorithm refines its path calculations as more information from neighboring nodes becomes available, resulting in more accurate and lower shortest path values over time. This process allows GRouteNet to eventually surpass PF-DDQN and Dijkstra's algorithm in finding the optimal paths.

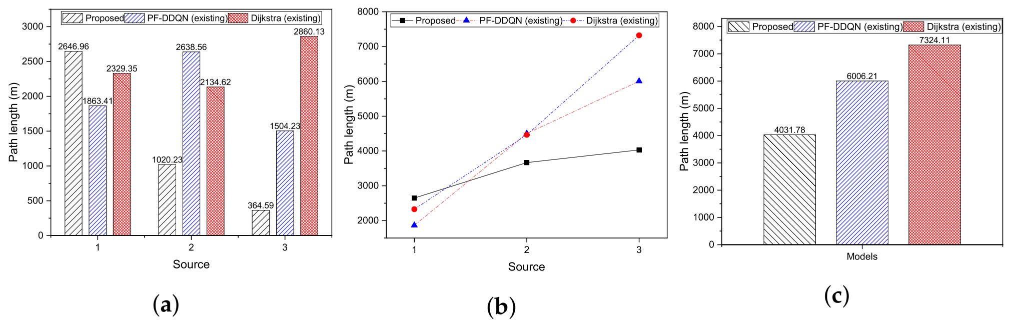


Figure 6. Comparison of shortest path from all sources. (**a**) Total paths calculated from all sources. (**b**) Cumulative path lengths from all sources. (**c**) Total of all paths from each source.

Figure 6b shows the cumulative lengths of the calculated shortest paths from all sources. It can be seen that the existing approaches steadily increases the cumulative calculated shortest paths, while GRouteNet contributes very little to increasing the total path lengths over the course of the model. The results in Figure 6c show the values for the total shortest path length of all sources from both approaches. It can be seen that GRouteNet effectively reduces the length of the shortest paths compared to the existing approach. Specifically, GRouteNet's total path length is 32.88% shorter than PF-DDQN and 44.96% shorter than Dijkstra. These results underscore GRouteNet's effectiveness in minimizing path lengths, thereby enhancing AGV travel efficiency and reducing operational costs in dynamic environments. This comparison shows that GRouteNet greatly reduces the length of the shortest paths.

Reducing path lengths for AGVs is an important aspect as it directly influences operational efficiency by reducing travel time and energy expenditure. GRouteNet demonstrates its effectiveness in reducing overall path lengths up to about 45% resulting in enhanced processes and optimal resource usage in industrial settings. These enhancements not only save operational expenses but also foster a more sustainable and efficient logistics process by diminishing the overall energy footprint of AGV operations.

We also record the execution time of different models for finding the shortest paths and present the results quantitatively in Table 6. The results in Table 6 show that GRouteNet attains a path length of 4031.7, representing a 32.88% enhancement compared to PF-DDQN and a 44.96% enhancement relative to Dijkstra. While GRouteNet's execution time of 9140 mS is not the least, it is more efficient than PF-DDQN, which has the longest execution time at 14,570 mS. Dijkstra's approach exhibits the fastest execution time (3900 mS); however, this comes at the cost of a much longer path length, indicating less optimal route efficiency. Although GRouteNet does not possess the shortest execution time, it is favored for its capacity to deliver a significantly reduced path length, resulting in decreased travel time and energy expenditure for AGVs.

Table 6. Comparison of shortest path length and execution time for different models.

Model	Shortest Path Length	Execution Time (mS)
PF-DDQN (existing)	6006.2	14,570
Dijkstra (existing)	7324.1	3900
GRouteNet (Proposed)	4031.7	9140

To assess the scalability of GRouteNet, we conduct experiments of finding shortest paths from three different sources node within a consistent environmental for all models. The results show that although the initial path length from the first source node was not optimized, GRouteNet substantially decreased path lengths from the second and third sources, leading to a comprehensive reduction in total path length across all sources. This illustrates GRouteNet's capacity to dynamically adapt and enhance pathfinding efficiency, particularly when additional information emerges inside the network. We performed experiments from three distinct source nodes, indicating that GRouteNet consistently identifies the shortest path under varied situations, hence illustrating its robustness in handling multiple pathfinding scenarios. This multi-source testing validates the model's scalability in operational situations, emphasizing its dependability and efficiency in dynamically adaptive route optimization. For SCTF, we record results after ensuring each AGV must have visited three times at the charging station. This approach enabled us to evaluate the algorithm's performance throughout several cycles, showing its capacity to efficiently manage charging priorities across repeated contacts. The findings validate that SCTF can reliably decrease waiting times, even with rising demand for charging resources, highlighting its scalability.

We present the results for the SCTF scheduling algorithm in Figure 7. Figure 7a shows the arrival times of the AGVs at the charging station. It can be seen that the arrival time is random and that all AGVs visited the charging stations three times during the experiment. Figure 7b shows the results of the waiting times for both algorithms for each AGV for each visit to the charging station. It can be seen that the SCTF greatly reduces the waiting time for each AGV each time it visits the charging station. In Figure 7c, we conclude the results of the waiting times for both algorithms by showing the commutative waiting time during the experiment. It can be seen from Figure 7c that the SCTF scheduling algorithm significantly reduces the waiting times in the queue at the charging stations. We also present the waiting times for all AGVs for both algorithms in Table 7.

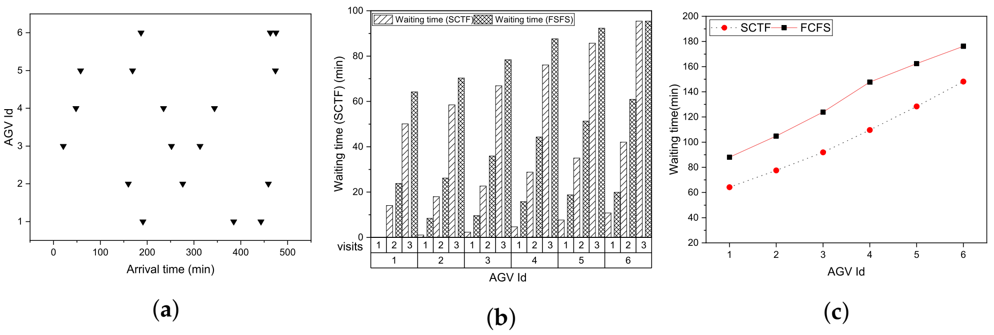


Figure 7. Comparison of waiting times at charging stations. (**a**) Arrival time of AGVs at charging station. (**b**) Waiting time of AGVs at charging stations. (**c**) Cumulative waiting time for all AGVs.

Table 7. Waiting times of AGVs with SCTF and FCFS scheduling.

AGV Id	Visit Number	Waiting Time (SCTF)	Waiting Time (FCFS)
1	1	0.0	0.0
	2	14.1	23.8
	3	50.1	64.2
2	1	1.1	8.4
	2	22.7	35.9
	3	58.5	70.3
3	1	2.3	9.6
	2	28.8	44.3
	3	66.9	78.4
4	1	4.7	15.8
	2	35.0	51.3
	3	76.1	87.6
5	1	7.7	18.8
	2	35.2	55.3
	3	85.7	92.3
6	1	10.8	19.8
	2	42.0	60.9
	3	95.4	95.4

The efficacy of the SCTF scheduling algorithm in minimizing waiting times at charging stations is based on its priority mechanism. SCTF reduces the average waiting time for AGVs by prioritizing those with shorter charging needs. The results show that SCTF significantly decreases waiting times relative to FCFS approach, hence maximizing charging station use and facilitating faster turnaround for AGVs with reduced energy requirements.

4.4. Limitations

Although GRouteNet shows considerable advancements in pathfinding and charging management for AGVs, it exhibits certain limitations. One limitation is its execution time, which, although faster than some existing approaches, is not the fastest. Some traditional pathfinding algorithms, such as Dijkstra's algorithm, may compute paths faster than GRouteNet; however, the path length of such traditional algorithm is much longer than GRouteNet. In this situation, GRouteNet may not be a good choice in environments where computational speed is emphasized above path length. Another potential limitation of GRouteNet lies in its reliance on local data from neighboring nodes. At the beginning, when this information is not yet fully available, the paths calculated by GRouteNet may not be optimal.

5. Conclusions

This paper addresses the fundamental challenge of finding the shortest path for the operation of AGVs and proposes GRouteNet. The proposed approach utilizes the capabilities of GNN architecture to find the shortest path between a pair of nodes. The proposed GRouteNet models the operational environment of AGVs as a connected graph and applies GNN to find the shortest path between the source and destination nodes in the graph. It uses a message passing mechanism and aggregates messages from the neighborhoods with the shortest path in the graph. Furthermore, the paper also addresses the challenge of longer waiting times at the charging station and proposes a Shortest Charging Time First scheduling algorithm that prioritizes the AGVs considering the charging time. The algorithm gives higher priority to those AGVs that require the shortest charging time. This approach leads to a reduction in the waiting time in the charging queue. The experimental results show that the proposed work outperforms the existing model. GRouteNet's real-time path optimization may be used for logistics, such as refining delivery routes for autonomous drones and regulating traffic flow in intelligent transportation systems, hence improving efficiency throughout autonomous transport networks.

Author Contributions: Conceptualization, S.N.K.; formal analysis, S.M.A.A.; investigation, S.N.K.; methodology, S.N.K. and S.M.A.A.; validation, S.N.K. and S.M.A.A.; visualization, S.M.A.A.; writing—original draft, S.N.K.; writing—review and editing, S.M.A.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no known conflicts of interest.

References

1. Zheng, G.; Chai, W.K.; Duanmu, J.L.; Katos, V. Hybrid deep learning models for traffic prediction in large-scale road networks. *Inf. Fusion* **2023**, *92*, 93–114. [CrossRef]
2. Baccari, S.; Hadded, M.; Ghazzai, H.; Touati, H.; Elhadef, M. Anomaly Detection in Connected and Autonomous Vehicles: A Survey, Analysis, and Research Challenges. *IEEE Access* **2024**, *12*, 19250–19276. [CrossRef]
3. Akber, S.M.A.; Khan, I.A.; Muhammad, S.S.; Mohsin, S.M.; Khan, I.A.; Shamshirband, S.; Chronopoulos, A.T. Data volume based data gathering in wsns using mobile data collector. In Proceedings of the 22nd International Database Engineering & Applications Symposium, Villa San Giovanni, Italy, 18–20 June 2018; pp. 199–207.
4. Kopelias, P.; Demiridi, E.; Vogiatzis, K.; Skabardonis, A.; Zafiropoulou, V. Connected & autonomous vehicles–Environmental impacts—A review. *Sci. Total Environ.* **2020**, *712*, 135237. [PubMed]

5. Chen, X.; Liu, S.; Zhao, J.; Wu, H.; Xian, J.; Montewka, J. Autonomous port management based AGV path planning and optimization via an ensemble reinforcement learning framework. *Ocean. Coast. Manag.* **2024**, *251*, 107087. [CrossRef]
6. Zhang, Z.; Wu, L.; Zhang, B.; Jia, S.; Liu, W.; Peng, T. Energy-efficient path planning for a multi-load automated guided vehicle executing multiple transport tasks in a manufacturing workshop environment. *Environ. Sci. Pollut. Res.* **2024**, 1–21.. [CrossRef]
7. Reuters. Ford Investors Impatient as Automaker Rev Up Efficiency Efforts. *Reuters* 2024. Available online: https://www.reuters.com/business/autos-transportation/ford-investors-impatient-automaker-rev-up-efficiency-efforts-2024-10-30/?utm_source=chatgpt.com (accessed on 3 November 2024).
8. Amazon. How Amazon Deploys Robots in Its Operations Facilities. 2023. Available online: https://www.aboutamazon.com/news/operations/how-amazon-deploys-robots-in-its-operations-facilities (accessed on 3 November 2024).
9. Kao, C.K.; Ke, Q.E.; Tang, K.Z.; Lin, P.J. AGV Scheduling Optimization of Automated Port Based on Disruption Management. *J. Transp. Technol.* **2024**, *14*, 423–444. [CrossRef]
10. Wu, X.; Zhang, Q.; Bai, Z.; Guo, G. A self-adaptive safe A* algorithm for AGV in large-scale storage environment. *Intell. Serv. Robot.* **2024**, *17*, 221–235. [CrossRef]
11. Tang, G.; Tang, C.; Claramunt, C.; Hu, X.; PeipeiZhou. Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. *IEEE Access* **2021**, *9*, 59196–59210. [CrossRef]
12. Zhang, D.; Chen, C.; Zhang, G. AGV path planning based on improved A-star algorithm. In Proceedings of the 2024 IEEE 7th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 5–17 March 2024; Volume 7, pp. 1590–1595.
13. Ghiaskar, A.; Amiri, A.; Mirjalili, S. Polar fox optimization algorithm: a novel meta-heuristic algorithm. *Neural Comput. Appl.* **2024**, *36*, 20983–21022. [CrossRef]
14. Akber, S.M.A.; Kazmi, S.N.; Mohsin, S.M.; Szczęsna, A. Deep learning-based motion style transfer tools, techniques and future challenges. *Sensors* **2023**, *23*, 2597. [CrossRef]
15. Ullah, Z.; Elkadeem, M.R.; Wang, S.; Akber, S.M.A. Optimal planning of RDS considering PV uncertainty with different load models using artificial intelligence techniques. *Int. J. Web Grid Serv.* **2020**, *16*, 63–80. [CrossRef]
16. Gruffeille, C.; Perrusquía, A.; Tsourdos, A.; Guo, W. Disaster area coverage optimisation using reinforcement learning. In Proceedings of the 2024 International Conference on Unmanned Aircraft Systems (ICUAS), Chania, Greece, 4–7 June 2024; pp. 61–67.
17. Mohsin, S.M.; Khan, I.A.; Akber, S.M.A.; Shamshirband, S.; Chronopoulos, A.T. Exploring the RFID mutual authentication domain. *Int. J. Comput. Appl.* **2021**, *43*, 127–141. [CrossRef]
18. Chen, J.; Zhang, X.; Peng, X.; Xu, D.; Peng, J. Efficient routing for multi-AGV based on optimized Ant-agent. *Comput. Ind. Eng.* **2022**, *167*, 108042. [CrossRef]
19. Shuo, S. Multi-AGV Path Planning Method via Reinforcement Learning and Particle Filters. *arXiv* **2024**, arXiv:2403.18236.
20. Lin, S.; Liu, A.; Wang, J.; Kong, X. An improved fault-tolerant cultural-PSO with probability for multi-AGV path planning. *Expert Syst. Appl.* **2024**, *237*, 121510. [CrossRef]
21. Fusic, J.; Kanagaraj, G.; Hariharan, K.; Karthikeyan, S. Optimal path planning of autonomous navigation in outdoor environment via heuristic technique. *Transp. Res. Interdiscip. Perspect.* **2021**, *12*, 100473.
22. Vlachos, I.; Pascazzi, R.M.; Ntotis, M.; Spanaki, K.; Despoudi, S.; Repoussis, P. Smart and flexible manufacturing systems using Autonomous Guided Vehicles (AGVs) and the Internet of Things (IoT). *Int. J. Prod. Res.* **2024**, *62*, 5574–5595. [CrossRef]
23. Chen, Y.; Li, W.; Huang, Y. A queuing theory-based model for AGV charging station scheduling. *Int. J. Prod. Res.* **2020**, *58*, 3569–3589.
24. Mehta, R.; Sahni, Y.; Khanna, R. Integration of renewable energy in AGV charging schedules: A predictive approach. *Appl. Energy* **2022**, *310*, 118575.
25. Liu, J.; Zhang, H.; He, K.; Jiang, S. Multi-objective optimization for energy-efficient AGV charging scheduling in smart factories. *J. Clean. Prod.* **2021**, *288*, 125558.
26. Zhang, L.; Wang, X. Dynamic charging scheduling for AGVs using reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 4356–4367.
27. Cheng, C.; Zhang, H.; Sun, Y.; Tao, H.; Chen, Y. A cross-platform deep reinforcement learning model for autonomous navigation without global information in different scenes. *Control Eng. Pract.* **2024**, *150*, 105991. [CrossRef]
28. Che, A.; Wang, Z.; Zhou, C. Multi-Agent Deep Reinforcement Learning for Recharging-Considered Vehicle Scheduling Problem in Container Terminals. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 16855–16868. [CrossRef]
29. Liu, Z.; Zhou, Y.; Feng, D.; Xu, S.; Yi, Y.; Li, H.; Wang, H. Dynamic Pricing of Electric Vehicle Charging Station Alliances Under Information Asymmetry. *arXiv* **2024**, arXiv:2408.06645.
30. Zhao, L.; Liu, T. Smart charging strategies for AGVs with renewable energy integration. *Energy Convers. Manag.* **2023**, *277*, 116513.
31. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:1609.02907.
32. Scarselli, F.; Gori, M.; Tsoi, A.C.; Hagenbuchner, M.; Monfardini, G. The Graph Neural Network Model. *IEEE Trans. Neural Netw.* **2009**, *20*, 61–80. [CrossRef]