robotics

MDPI

Article

Controlling Fleets of Autonomous Mobile Robots with Reinforcement Learning: A Brief Survey

Mike Wesselhöft *, Johannes Hinckeldeyn * and Jochen Kreutzfeldt *

Institute for Technical Logistics, Hamburg University of Technology, 21079 Hamburg, Germany
* Correspondence: mike.wesselhoeft@tuhh.de (M.W.); johannes.hinckeldeyn@tuhh.de (J.H.); jochen.kreutzfeldt@tuhh.de (J.K.)

Abstract: Controlling a fleet of autonomous mobile robots (AMR) is a complex problem of optimization. Many approached have been conducted for solving this problem. . They range from heuristics, which usually do not find an optimum, to mathematical models, which are limited due to their high computational effort. Machine Learning (ML) methods offer another potential trajectory for solving such complex problems. The focus of this brief survey is on Reinforcement Learning (RL) as a particular type of ML. Due to the reward-based optimization, RL offers a good basis for the control of fleets of AMR. In the context of this survey, different control approaches are investigated and the aspects of fleet control of AMR with respect to RL are evaluated. As a result, six fundamental key problems should be put on the current research agenda to enable a broader application in industry: (1) overcoming the "sim-to-real gap", (2) increasing the robustness of algorithms, (3) improving data efficiency, (4) integrating different fields of application, (5) enabling heterogeneous fleets with different types of AMR and (6) handling of deadlocks.

Keywords: reinforcement learning; autonomous mobile robots; multi agent systems; AI-Enabled robotics; fleet control; systematic literature review

Citation: Wesselhöft, M.; Hinckeldeyn, J.; Kreutzfeldt, J. Controlling Fleets of Autonomous Mobile Robots with Reinforcement Learning: A Brief Survey. *Robotics* **2022**, *11*, 85. https://doi.org/10.3390/robotics11050085

Academic Editor: Charalampos P. Bechlioulis

Received: 11 July 2022
Accepted: 19 August 2022
Published: 30 August 2022

1. Introduction

Controlling fleets of autonomous mobile robots (AMR) is becoming increasingly important for many companies [1,2]. Especially in logistics and production, a high priority is being given to the control of fleets of AMR as a part of automation [3]. AMR enable the flexible and automated supply with material in factories and warehouses. Four sub-problems need to be solved in order to control effectively a fleet of AMR [4,5]: routing, scheduling, collision avoidance and the orientation in dynamic environments (see Section 6).

However, these four sub-problems are challenging. A particular challenge is the interdependence of the individual sub-problems. If an optimal result is generated for one of these sub-problems, it does not mean that the entire control system runs optimally. Therefore, the measures of control and optimization need to be coordinated to find the best possible solution. In addition to the strong interdependencies, another challenge is to also scale depending on the size of the company. This scalability increases complexity, which results in a high computing effort [6]. The third challenge is the dynamic environment in which AMR fleets operate. It requires a fast and flexible processing of these operations.

ML methods, especially RL (for formal details, please see Section 4) approaches, have the potential to successfully tackle these challenges. Firstly, RL methods have demonstrated a great potential for solving highly complex problems [7,8]. Furthermore, RL algorithms are capable of controlling agents both centrally and decentrally (see Section 5) and thus solving different problems at once. Therefore, it offers different solutions to realize scalability (see Section 7.1.1). Due to their reward-structured optimization [9], RL algorithms also provide the flexibility of integrating different parameters for optimization and thus have an increased potential to coordinate agents in dynamic environments (see Section 6.4). This

potential has already been recognized by many authors (see Section 3). To the best of our knowledge, no paper has been published yet that provides an overview of existing research about the control of fleets of AMR with RL (see Section 2), which is the motivation of this survey.

2. Related Literature

Existing reviews on the topic of controlling fleets of autonomous mobile robots in conjunction with RL were evaluated for the need of another survey. A total of 18 existing reviews with relevance to this topic were found. Two older than 2015 and thus no longer relevant due to the rapid progress within the area of RL. Five reviews were excluded as they deal with other areas of application than control of robots like control of agents in graphic-based games [10], power distribution [11] or without an explicit focus on RL [12–14]. Taking the remaining ten reviews, two were excluded due to their focus on only one topic, transportation [15] and collision avoidance [16]. Another five papers were excluded as the focus is on theoretical concepts of RL methods and not on potential applications for AMRs [17–21]. The remaining four review papers also describe the control of fleets of AMR. However, [22] focuses on the general use of machine learning methods for controlling fleets of autonomous mobile robots without a specific focus on RL. The review by [23] evaluates the control of single- and multi-agent systems using RL. It addresses problems in the implementation of RL algorithms and deals with exploration and collision avoidance in dynamic environments. Path planning as a whole and scheduling are not discussed. Finally, to mention are [24,25]. Although [24] deals with the control of autonomous vehicles using RL, its focus lies on autonomous driving. On the one hand, [25] provides an extensive analysis on the navigation of mobile robots. However, the focus is on the navigation of individual robots and only six papers of the area of multi-robot navigation are presented. Altogether, the reviews show that the focus on the control of fleets of AMR itself is missing in the existing literature. Centralized and decentralized approaches are rarely compared (see Section 5) and potential sub-problems to solve, such as routing and scheduling, are insufficiently considered (see Section 6). It denotes that no in-depth survey about using RL for the control of fleets of AMR exists up to now. In order to close this gap, the research objective of this study is to investigate control methods for autonomous fleets and their applications in logistics and production in the industry. In doing so, the intention of this review is not to provide an extensive review of the application of RL to AMRs, but to complement the existing literature with a brief synthesis of the latest research findings on AMR fleet management with RL.

3. Survey Methodology

A structured literature survey approach, based on [26], was chosen in order to investigate RL approaches for the control of autonomous fleets.

Scientific publications from the year 2015 onwards in the field of computer science were acquired using one structured and one unstructured literature search approaches. The structured literature search was conducted in the Scopus database and with the search string in Table 1. Taking the search string, 402 papers were found of which 27 were added to the literature database for further investigation. Figure 1 shows the process of reviewing the literature. In an additional unstructured literature search another 18 sources had been added using a track and trace approach. These publications were examined for the use of RL for control of fleets of AMR.

Table 1. Search String for the structured literature search. Each column contains terms that are connected to each other by an AND operation. The rows connected by an OR operation.

Algorithm	Application	Control
Reinforcement Learning	Scheduling	fleet
RL	Path planning	Multi robot
Deep Q Learning	Path finding	Multi agent
DQL	collision avoidance	Swarm

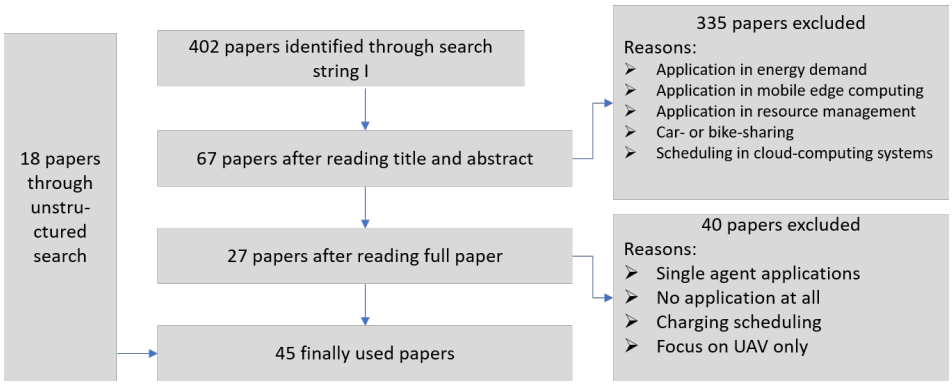

Figure 1. Process of paper selection.

4. Approaches for Reinforcement Learning

4.1. Basic idea of Reinforcement Learning

RL is a fundamental approach of machine learning. In RL, an agent learns independently a strategy to maximize received rewards. The basic concepts of RL and Q-Learning originate from the standard works of [9]. A condition necessary for reinforcement learning is the Markov property. This is fulfilled if the conditional probabilities of future states depend only on the current state. Therefore, the problem of RL itself is defined as a Markov Decision Problem (MDP). An MDP is defined as a quintuple:

$$M = (S, A, R, \mathcal{P}) \tag{1}$$

where S is a set of states, A is a set of actions, $R : S \times A \to \mathbb{R}$ denotes the reward function, giving an agent a reward based on a taken action $a \in A$ in a state $s \in S$. $\mathcal{P}$ denotes the transition probabilities. In a MDP, a policy $\pi(a|s)$ describes the probability of mapping state $s \in S$ to $a \in A$. In order to solve a RL problem, an optimal policy is required. An extension of the MDP that is relevant for RL algorithms is the so-called partial observable MDP (POMDP). Since agents often do not have access to the complete environment, an additional concept is necessary for this. A POMDP is defined as follows [27]:

$$M = (S, A, R, \mathcal{P}, \Omega, O). \tag{2}$$

S, A, R and $\mathcal{P}$ describe a MDP, Ω is a set of observations and $O : S \times A \to \Pi(\Omega)$ is the observation function that gives a probability distribution over possible observations,for each action performed and the state reached by the agent. A POMDP is a MDP in which the agent makes an observation based on an action and a resulting state. The agent's goal is still to maximize the future reward.

4.2. Q-Learning

Q-learning is a model-free RL algorithm to learn the value of an action in a particular state. It can handle problems with stochastic transitions and rewards without requiring adaptations. For any finite MDP, Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over all successive steps, starting from the current state. The simplest form of Q-learning is one-step Q-learning. It is defined by the following equation:

$$Q(S_{t+1}, A_{t+1}) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{3}$$

where α is the learning rate.

4.3. Deep Q-Learning

Building upon DRL, [28] introduced deep Q-Networks (DQN) in 2013. They combine the model-free, off-policy Q-Learning with Deep Neural Networks. Agents can successfully learn control policies through the use of high-dimensional raw inputs from sensors without manually designed features, but with the use of DRL. This is possible due to the end-to-end structure of the network, which is responsible for extracting the relevant functions. The output of the Q network is used to determine the best action in a given state with only a single forward pass. It outputs a probability distribution for every action and was already used in Atari games [28].

4.4. Proximal Policy Optimization

DQN methods can be used to solve a variety of RL problems. However, Q-learning (with function approximation) is poorly understood and also fails on some simple problems. To tackle these problems the authors of [29] have developed proximal policy optimization (PPO). The basis for this algorithm is a policy gradient method. These methods estimate the policy gradient and optimize it using a stochastic gradient ascent algorithm.

5. Type of Control of Autonomous Fleets

There are two main approaches to control a fleet of AMR with RL. One of them is centralized control, where the fleet is being controlled by an algorithm that receives data from all agents and provides control commands for the entire fleet. In decentralized control, the algorithm is trained for each individual agent to control itself. The fleet is then linked to some kind of a "team-concept".

5.1. Centralized Control

RL can be used to centrally control fleets of AMR. In [30], the authors use a centralized control to route a fleet of AMR in a simulated environment. Therefore, the problem is transformed in a discrete time-step decision-making process and modeled as a Markov Decision Process (MDP). Afterwards, an asynchronous DQN algorithm is used to navigate a total of 22 AMR in one grid. In comparison to standard algorithms (such as random policy and regulation method) the asynchronous DQN can handle scenarios with more AMR. Another discrete approach is used in [31]. In this paper a multimodal DQN with action-limited output is used to control a fleet of 50 AMR. The environment is also simulated and it is shown that the combination of action-limitation and multimodal DQN exceeds the results, when action limited and multimodal DQN approaches are used separately. While the final two papers were focusing on path finding problems, [32] tackles the field of collision avoidance. To solve the minimal time-energy problem an integral reinforcement learning algorithm (IRL) is trained in continuous time, action and input. In addition, the algorithm is trained in an environment with unknown disturbances and different constraints to the input. Using IRL the authors had to develop a novel approximate cost function, since the original problem has a finite horizon, while IRL usually needs an infinite-horizon to be applied easily. In addition to proving convergence of the method developed, three robots are controlled in a simulated environment. In [33] compares a DQN based RL-algorithm to known rule-based algorithms for path finding. This approach differs from the previous ones by controlling and synchronizing the individual agents in a herd. Therefore, a partially observable environment is used and tasks are handled in a simulated environment by two to four agents. Another swarm-based algorithm is provided by [34]. In their paper they use a combination of Particle Swarm Optimization (PSO) and Q-Learning (QL) in a discrete action space controlling up to 100 agents in a simulated environment. In [35] take the combination of PSO and QL even further by controlling a fleet of autonomous underwater vehicles (AUV) in continuous time. Even though they are controlling just three agents, the environment gets much more complex since motion happens in three dimensions. In addition to a simulation, real data (from Marine Science Data Center) is used to get a more robust control. In [36] describes another algorithm using also a flocking control

in continuous time, while planning in discrete time to handle collision avoidance. They use a flocking control (FC) for motion planning combined with a RL-algorithm to predict behavior strategies. The combined approach achieves better results in terms of learning efficiency in comparison to a conventional FC and a standard RL algorithm. There exists already some approaches to controlling fleets of autonomous vehicles using RL. So far, these have mainly been tested in simulation environments and are limited by a number of up to 100 agents. RL-based centralized controls require instantaneous communication between different agents. As the number of agents increases, the number of possible actions that can be executed in each step increases as well. This results in a higher demand of computing power for centralized control approaches. Thus, centralized approaches are effective at controlling smaller fleets, but lose performance for larger ones.

5.2. Decentralized Control

Decentralized approaches are also suitable for controlling fleets of AMR. Compared to centralized, decentralized approaches have the potential to reduce computational effort, because only the decisions of a single agent need to be computed. In [37] a decentralized fleet control for path planning is presented. Therefore, a so-called Dyna_Q algorithm is trained on single agents. These agents share their knowledge with each other to function as a fleet. By doing so, agents can profit from each other's experience and the overall training time can be reduced. In the end, the algorithm is tested in a simulated environment with one to three agents. Using similar fleet sizes, [38,39] are also evaluating their algorithms in a simulated environment. While [39] is using the combination of QL with a convolutional neural network (CNN) showing good results in comparison to A* and D* algorithms for routing-problems, [38] developed a QL-based algorithm integrating prior knowledge into the decision process. Evolving these approaches, [40] is combining Deep Q-Learning with Long Short-Term Memory (LSTM) and the concept of imitational learning to control fleets of four up to 1000 agents (algorithm called PRIMAL). In addition to evaluating their algorithm in a simplified simulation environment, the authors also use a hybrid factory mockup to control a fleet consisting of two real and two to three simulated robots. In [41] the algorithm is extended to control up to 2048 agents. Comparing themselves to PRIMAL and expanding its results, [42,43] use different strategies to further evolve RL for decentral path planning. Both are working with partially observable environments. In [42] utilizes a globally guided RL approach (G2RL), for which they developed a novel reward structure to achieve long-term global information. The algorithm itself is the combination of CNNs, followed by LSTM for information extraction and then using a Multilayer Perceptron (MLP) to estimate the correct actions for up to 128 agents. In [43] uses multi-agent path planning with evolutionary reinforcement learning (MAPPER). They use two different inputs: the observation represented as a three channel image and future waypoints planned by an A* planner. Using the combination of both inputs, the algorithm calculates sub goals of the complete path. An evolutionary approach is chosen in order to transfer the algorithm to a multi-agent system. During the training process, agents using bad polices are eliminated and only successful agents continue to be trained. In [44], the authors extend the definition of the problem of MAPPER and PRIMAL and compare with the latter. The Q-learning based algorithm relies on selective communication between the decentrally controlled agents. With the help of this approach, up to 128 agents can be controlled and the success rate is significantly higher than that of PRIMAL and another algorithmic approach DHC [45], especially for larger fleets. Other approaches with partially observable environments are developed in [46] by using Voroni Partitions (VP) combined with QL to control three agents in a simplified simulated environment. The authors of [47] use a promising approach to solve the multi-robot patrolling problem (MRPP). They combined Bayesian Decision Making with a reward based learning technique. Using the decentralized approach, up to 12 agents are successfully coordinated in simulated environments and six in real scenarios. Another interesting approach has been developed in [48]. Using their decentralized and continuous approach, up to four agents are coordinated simultaneously. However, the

paper focuses on reducing the computational complexity of the Q-learning algorithm used. Thus, the memory size of the Q-learning algorithm can be reduced by up to 70%. Decentralized controls have a high potential of controlling fleets of autonomous robots. Compared to centralized controls, they allow significantly more agents to be controlled simultaneously (according to [40,41] up to 2000, seen in Table 2). Furthermore, there are already some approaches that control autonomous robots (e.g., Turtlebot [40]) in a real environment under simplified conditions. A weakness of decentralized approaches is the assignment of rewards. The training of sparse-reward structures is particularly complex because some information can get lost when agents are trained at the same time. This feature leads to an increase of training time and makes it more difficult to find an optimum.

6. Subproblems of Fleet Control

In this paper, we focus on the three most important problems for controlling fleets of AMR for industrial applications. These sub-problems are collision avoidance, path planning/routing and scheduling. One may argue that collision avoidance is a part of path planning, however, there are some scientific papers that explicitly and exclusively refer to this topic [32,49–53], which is why they are discussed separately, while we will focus on the combination of collision avoidance and path planning in Section 6.5.

6.1. Collision Avoidance

A collision avoidance system is designed to assist drivers/ agents to reduce their overall risk of collision with other objects. In [32] tackles this problem in a totally observable environment using IRL. Further input constraints and unknown environmental disturbances are added. To achieve collision avoidance they add an artificial potential field into their approximate cost function. In [49] investigates scenarios, where communication cannot be established reliably. Therefore, the agents have to be coordinated without having access to each other's observations or intentions. This problem is handled by solving the two-robot collision avoidance problem with Deep Reinforcement Learning (DRL) and generalizing the resulting policy to multi-agent collision avoidance. Letting their agents make decisions without any communication between them, [50,52,53] use a hybrid fully decentralized approach to solve the collision avoidance problem. A novel multi-scenario RL algorithm is developed and trained with a robust policy gradient method. Afterwards, the learned policy is combined with traditional control approaches to achieve a more robust control. In the end, the authors also test the hybrid-policy in simplified real-world applications and show that it is running stable in different scenarios. A further decentralized approach is introduced in [51]. In a communication-free environment, they combine a PPO algorithm with an actor-critic approach and are thus able to control up to 24 agents in a simulated, discrete environment. The input was an image-based overview of the test area. In contrast to the other approaches, they use a map-based solution representing the agent's environmental information in an egocentric grid. The approach is then successfully deployed to real robots and it is shown that it generalizes well to new scenarios. Another approach which relies on the use of PPO is [54]. In [54], a map-based PPO approach is chosen to coordinate up to 10 agents in a decentralized, continuous, simulated environment. Compared to conventional PPO and the ORCA algorithm, the algorithm performed better in the experiments and could even be applied in a real-world scenario. The authors of [55] also successfully use an actor-critic approach. In this case, the algorithm is combined with force-based motion planning approach to control up to 10 agents decentrally in a simulated test environment. A higher robustness is achieved by using randomization. Several RL algorithms have already been developed in the field of collision avoidance. The authors mainly focus on limitations like missing communication possibilities between the agents and unknown environments. The approaches handle this kind of problems very well and even show promising implementation possibilities in real applications. To improve this area even further, it is necessary to apply the results to heterogenous fleets with different types of AMR and develop solutions to handle serious interferences, e.g., deadlocks.

6.2. Path Planning

Another field is the so-called path planning problem. In this field the majority of publications can be found (see Table 2). While algorithms with an exclusive focus on collision avoidance are presented in the previous section, the papers of this section view collision avoidance as an integrated part of path planning. In [37,38], the authors present fleet control approaches for one to three agents, which are trained in a totally observable environment. In [37] uses a Dyna_Q to investigate the properties of Dyna algorithms in combination with deep learning. In [38] prior knowledge is used to reduce the data effort of the training process for QL algorithms. The algorithm is then shown to be more efficient than conventional QL approaches. Also using the results of a single-robot-control, [56] develops a classic Deep Deterministic Policy Gradient (DDPG) for single robot mapless navigation tasks to obtain the Parallel Deep Deterministic Policy Gradient (PDDPG). The PDDPG is therefore trained in a Gazebo simulation environment, controlling up to eight agents in a rectangle formation. They use raw 2D lidar sensor as input data and the relative target positions. It is shown that the multi-robot system accomplishes the collaborative navigation tasks with a high arrival rate. In [34,42] provide different approaches to extend algorithms from solving the single robot path planning problem to fleets of over 100 agents. In [34] combines PSO and QL to control the fleet in a partially observable environment (QL is taking the action and PSO is defining the trajectory). The algorithm is then compared to a Genetic Algorithm (GA) and an Artificial Bee Colony Algorithm (ABCA). It outperforms both of them in scenarios with a large fleet size. It is shown that the algorithm has at least 90% of its agents reaching the target in every test. Moreover, [42] introduces G2RL to control up to 128 agents in a partially observable dynamic discrete simulation environment. An A* algorithm is used for the global guidance and a novel reward structure is implemented to encourage the agents to explore every potential solution instead of just following the guidance. Next, the algorithm is compared to five state-of-the-art benchmarks (A* based Global Replanning, Hierarchical Cooperative A* (HCA*), enhanced conflict-based search (ECBS), optimal reciprocal collision avoidance (ORCA), PRIMAL) on different maps with different numbers of agents. Only the centralized approach HCA* outperforms the algorithm and ECBS is achieving similar success rates. It should be noted that PRIMAL, as developed in [40,41], was originally trained in a continuous partially observable environment. The algorithm is able to control up to 2048 agents in a simulated environment and can even control physical robots in a factory mockup. While PRIMAL is performing poorly in small environments with high obstacle density and gets outperformed by CBS, ODrM* and ORCA, it is more successful in larger environments, keeping a high success rate in cases where the other algorithms even fail. As part of their work, [57] use an RL algorithm to plan sub-goals of a complete route. In a dynamic environment, a model predictive control (MPC) is used to generate the complete route. The environment is totally observable and the RL-based algorithm is given a discrete action space to evaluate die routes for the sub-goals. With the help of a simulation, the resulting algorithm is compared with other DRL methods and shows a higher efficiency. The authors of [58] also investigate navigation in dynamic environments. Three robots are controlled in a Gazebo simulation. The travel time can be reduced by up to 14.32% compared to other algorithms, such as ORCA. Furthermore, dynamics randomization is used to facilitate a transition to a real scenario. Working on the routing of a heterogeneous fleet to solve the Capacitated Multi-Vehicle Routing Problem, [59] utilizes a centralized training and decentralized-execution paradigm. It is shown that the developed DRL approach provides near optimal results and is tested in a simulated environment to control three vehicles. In comparison to heuristic approaches, the algorithm yields better results, but does not manage to approach the performance of Google's OR tools [60]. A further method is presented in [61]. The authors combine stochastic Model Predictive Control (SMPC), Simultaneous Localization and Mapping (SLAM) and recurrent neural networks (RNN) with DQN to control heterogeneous fleets of robots. Another interesting approach was chosen by the authors of [62]. They combined an actor-critic algorithm with tree search to decentrally control more than 150 agents in a discrete, simulated environment. The

algorithm achieved promising results especially for high numbers of agents, outperforming PRIMAL among others. As this is the most investigated field, path planning/routing also has the greatest variety of algorithms used. While some authors focus more on the effective completion of orders, others look at scalability in particular. Scalability is the most important aspect. Existing approaches have shown weaker results for fleet sizes of 100 vehicles or more [42]. In contrast, RL algorithms are capable of coordinating up to 2000 vehicles [40,41] without significantly reducing their success rate.

6.3. Dispatching and Scheduling

The third and last field that has been studied is dispatching and scheduling. The order in which driving jobs are assigned is highly relevant for an AMR-fleet and lays the foundation for promising fleet control. In [63] the authors tackle the problem of task allocation for online ride-sharing platforms, which is similar to problems in logistics and production. To solve this problem, they introduce two novel contextual multi-agent RL approaches: contextual multi-agent actor-critic (cA2C) and contextual deep Q-network (cDQN) algorithm to handle the dynamically changing action space. In a simulated environment the algorithm handles up to 5356 agents and is compared to different state-of-the-art approaches. Another problem of scheduling is solved in [64] by scheduling school buses via a RL-based genetic algorithm. To evaluate their model, they test it on a geospatial dataset consisting of road networks, trip trajectories of buses, and the address of students. In comparison to the initial state process, they can reduce the travel distance by 8.63% and 16.92% for buses and students. Travel time can be reduced by 14.95% and 26.58%. In [35] the authors tackle the problem of scheduling for up to three agents in an underwater rescue assignment. They present a QL-based control combined with PSO in a continuous environment. The algorithm is trained on simulated as well as real data in a simulation environment . Afterwards it is compared to frequently used algorithms in the field of real time underwater rescue assignments like a neural network based improved self-organizing map (ISOM) and an improved ant colony optimization (IACO). In [65,66] a Multi-DRL approach for scalable parallel Task Scheduling (MDTS) in autonomous driving is introduced. They evaluate the performance of MDTS by comparing it to other popular schedulers (e.g., PSO, least-connection, A3C) through simulation experiments, showing improvement on various parameters. In [67] aims to solve a multi-AGV flow-shop scheduling problem with a RL method. The QL approach is therefore tested in a simulation (consisting of six machines, 50 jobs and two AGVs) against an existing multi agent method, showing promising results in minimizing the total makespan. In [68] an experience sharing DQN for job scheduling in heterogeneous fleets is developed to solve the dynamic dispatching problem in mining. Compared to other approaches (Shortest Queue, Smart Shortest Queue), the presented algorithm increases productivity in two simulated environments for fleets consisting of 16 and 56 vehicles, respectively. Contrary to expectations, only two approaches for problems in logistics and production have been found in the area of dispatching and scheduling. Instead, the focus here was stronger on ride-sharing and autonomous driving, which are comparable to problems in logistics and production. Most of the applications developed for this purpose do not go beyond a simulation model. Due to the high dependency on routing/path planning, more research should be done in this area.

6.4. Orientation In Dynamic Environments

Another important area in the control of fleets of AMR is orientation in dynamic environments. Since unpredictable events often occur, especially in areas such as production and logistics, it is necessary for agents to be able to handle dynamic environments. In the context of this work, some papers have been found which deal with orientation in only these environments. In [42] use three different maps, which they swap during the training. In addition, obstacles are distributed and the start and finish positions are randomly assigned. The algorithm is even able to handle deadlocks. A similar approach is taken by the authors of [43]. With the help of curriculum learning, they increase the number of dynamic objects within

the test environment during training, as well as the targets that the fleet can approach. Similar to curriculum learning, the authors of [57] use a subgoal recommendation policy learned by RL. Combined with an MPC, this approach can be used to reduce the complexity of the dynamic learning environment and enable efficient control of multiple agents. A different approach to solving the problem is chosen by [59]. A cooperative architecture is developed that combines experiences of the individual robots to enable target location allocation. These approaches show the potential of RL for navigation within dynamic environments. Using different methods, such as randomizing the training environment or linking information of individual agents, the robustness of the algorithm with respect to dynamic events can be increased. With further research in this area, situational events, such as deadlocks, should be even better resolved.

6.5. Combining Different Problems

While the previous fields all have high relevance in themselves, it is particularly interesting to have an integrated solution. Since the solution of a certain problem has an influence on the solution of others (and vice versa), a nearly optimal overall result is more likely to be achieved. Additionally, the interaction of different algorithms with each other would be avoided. In [46], VP are used in combination with QL to tackle environmental exploration, collision avoidance and path planning all at once. For this purpose, three agents are controlled on the one hand in a simulation and on the other hand in a real application using Turtlebots. A continuous action space is used for this research. Another approach of combining two different fields of application is developed in [69]. The problems of multi-robot task allocation (MRTA) and navigation are tackled simultaneously using DRL. In order to combine these two into a new metric, the Task Allocation Index (TAI) is introduced, enabling the algorithm to measure the quality of its performance on the task. A PSO approach is used for decentralized control of the fleet. The algorithm is trained in the Gazebo simulation environment where it is able to control and coordinate two robots. The authors of [70] also try to address the problems of scheduling and routing simultaneously. The presented centralized control is based on a DQN and guides up to four agents through a totally observable environment. The algorithm achieves better results in comparison to the Shortest Travel Distance rule and works particularly well in more complex scenarios. Another approach for the combination of routing and scheduling is provided by [71]. In their paper they use PPO to control three robots in the game engine Unity and solve a material handling problem. Combined algorithms already show a high degree of optimization potential but can only be found in four papers [46,69–71]. Due to this potential, combined solutions should be further explored.

A simple comparison of the relevant references from the systematic literature review is shown in Table 2. The control type abbreviations are c for central and d for decentral. Data efficiency contains concepts with which the data efficiency could be increased during training or the amount of required training data could be reduced.

Table 2. Comparison of literature from the systematic literature search.

Ref.	App.	Control Type	Algorithm	Agents	Input	Data Efficiency
[30]	PP	c	asynchronous DQN	22	totally	LSTM
[31]	PP	c	actionlimited multimodal DQN	50	totally	action replay
[32]	CA	c	Integrated RL	3	totally	none
[33]	PP	c	DQN	4	partially	none
[34]	PP	c	PSO + QL	10	totally	none
[35]	S	c	PSO + QL	3	totally	none
[36]	CA	c	FC + RL	4	totally	none
[46]	PP + CA	d	VP + QL	3	partially	none
[43]	PP	d	Evolutionary RL	20	partially	none
[42]	PP	d	G2RL	128	partially	LSTM
[38]	PP	d	QL	3	totally	prior knowledge
[39]	PP	d	DQN	4	totally	none
[40]	PP	d	PRIMAL	1024	partially	LSTM
[41]	PP	d	PRIMAL2	2048	partially	LSTM
[37]	PP	d	DynaQ	3	totally	none
[57]	PP	d	DRL + MPC	10	totally	subgoal planning
[69]	PP + S	d	PPO + DQN	4	partially	none
[53]	CA	d	PPO	100	partially	none
[50]	CA	d	PPO + PID	100	partially	none
[62]	PP	d	AC + TS	150	totally	post-processing
[54]	CA	d	AC + PPO	24	totally	selective communication
[51]	CA	d	Map-based PPO	10	partially	none
[55]	CA	d	GA3C-CARDL-NSL	10	partially	none
[45]	PP	d	QL	128	partially	selective communication
[48]	PP	d	QL	4	partially	sample efficient QL
[65]	S	d	MDTS	8	totally	partial data input for A3C

7. Discussion

In this section, the results from the literature review are discussed again in more detail. For this purpose, the chapter is divided into challenges and solutions. Altogether, three challenges are presented that must be overcome in order to apply RL in an industrial context. The three challenges are divided into six key aspects, for which possible solutions are mentioned in the respective section.

A variety of RL algorithms for controlling autonomous fleets of vehicles has been presented in the previous sections. In addition to the types of control, the possible subproblems for such algorithms have been presented in particular. There exists already a number of different promising algorithms for controlling fleets of AMR using RL, especially in environments of higher complexity. While some of the algorithms are already capable of coordinating fleets of up to 2000 vehicles [40,41], others are particularly efficient or capable of taking on tasks that could previously not be solved in this way. The basis is the potential of RL algorithms to handle high complexity problems, as well as the flexibility in solving problems, due to the reward concept. However, there are also challenges that have to be taken in order to develop a control system suitable for a wider application in industry. Six key aspects should be investigated further in order to achieve such a transfer. (1) overcoming the "sim-to-real gap" [23], (2) robust algorithms [72], (3) data efficiency [25], (4) combining different fields of application [69–71], (5) enabling heterogeneous fleets [73] and (6) handling deadlocks [42]. These six key aspects can be divided into the challenges

scalability, interoperability and dynamic environments (see Figure 2). We will deal with these categories separately in the discussion.

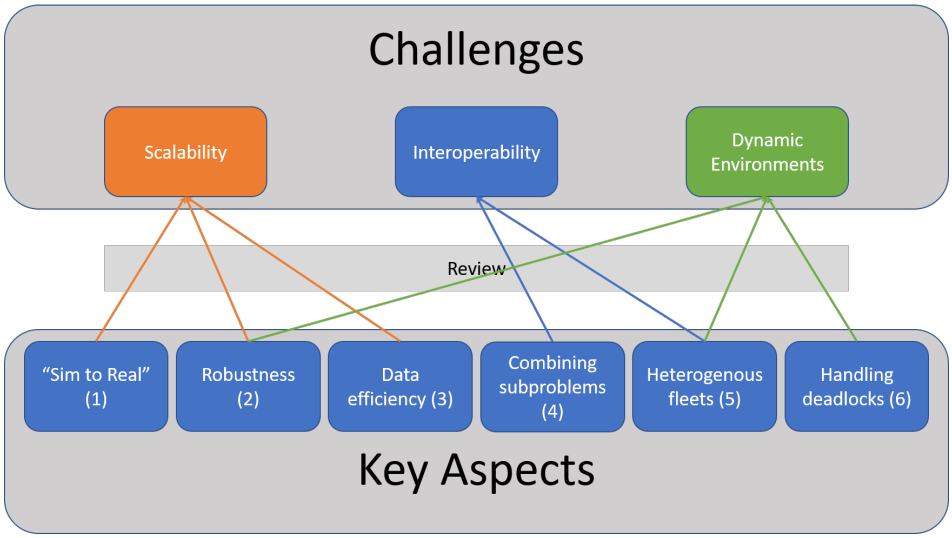

Figure 2. Identified challenges for reinforcement learning controlled fleets of AMR with corresponding key aspects.

7.1. Challenges

In this section, the key aspects are mapped to their respective challenges and potential problems in the implementation of fleets of AMR are examined.

7.1.1. Scalability

Scalability is an important feature in the field of fleet control. In this context, scalability is the ability to handle increased workload by repeatedly applying a cost effective strategy for extending a system's capacity [74]. Especially in the area of logistics and production, it is necessary to control a large number of different agents in order to complete the extensive tasks in the necessary time frame. For this reason, it is necessary for a fleet of AMR to be well scalable to fit the given circumstances. Due to the high computational cost of mathematically optimal solutions and the lack of optimality of heuristic approaches, RL methods offer a good alternative to their scalability. In the course of the research, several papers have been identified, which prove this property. In [40,41] the developers of PRIMAL prove that their algorithm can be used to control up to 2048 agents. Thus, a high potential for scaling fleets of AMR already exists and needs to be explored in more detail. The first three key aspects can be summarized in scalability. Aspect one is closing the "gap between simulation and reality". In order to enable scalability of RL methods and thus their use in real applications, it is necessary to close this gap. Some authors already address this point and approach the implementation of fleet control systems in reality [40,46,51]. Until now, however, these run mainly in simplified environments. An interesting approach was chosen by the authors of [35], who trained their algorithm on real data to control a fleet of AUV. This approach enables the testing of such systems under almost real conditions and thus allows the incorporation of irregularities and fuzziness of sensor data into the training of the algorithm. At the same time, costs of real applications can be reduced. Future work should focus on real-world applications and not simulations or laboratory settings. The second aspect that needs further improvement is the robustness of the algorithms. Robustness is a key factor in the scalability of RL methods. Since the problems become correspondingly more complex with increasing scale, it is necessary that the algorithm has a low susceptibility to errors. Within the scope of this work, few papers have been found that place a central focus on the robustness of their algorithm. For example, several algorithms are compared with PRIMAL [42,45,62] and perform very

well compared to it, but this is often due to the fact that PRIMAL has not been trained for the selected scenario. Accordingly, poor comparability is ensured and it is revealed how poor the robustness of individual algorithms is to new environments. In [58], an attempt was made to address this problem by a control structure. Similar approaches need to be investigated more in the future to allow for some generalizability and thus robustness of RL algorithms. Robustness can also be increased even further through the third key aspect: data efficiency. The more data is available, the more extensive the training can be designed and thus the robustness of the algorithms can be increased. Data efficiency is the third important aspect that needs to be explored in more depth. High data efficiency can reduce costs and the complexity of the problem which is necessary for scalability. There exists some algorithms that are associated with concepts such as LSTM ([30,40,42]), Low-Cost Q-Learning [48] and action replay ([31]). In addition, [38] use prior knowledge to enable more efficient training. However, new concepts are needed, that further reduce the amount of data required. Approaches need to be developed to increase the amount of data collected and simplify the form of input data. RL algorithms have the potential to control fleets of over 2000 agents [40,41]. In order to implement this scalability in industry, points (1)–(3) need to be better explored.

7.1.2. Interoperability

Interoperability is the ability of two or more software components to cooperate despite differences in language, interface, and execution platform [75]. There is a very high priority when controlling fleets of autonomous vehicles in order to produce the best possible results. Key aspects (4) combine different fields of application and (5) enabling heterogenous fleets fall into this challenge. Thereby (4) describes the combination of sub-problems, which have high dependencies to each other, such as scheduling and routing. Research in these cross-field algorithms is needed as they enable smooth and efficient control. Within the scope of this work, only four approaches ([46,69–71]) have been found, which combine different sub-problems for controlling fleets of AMR. Since RL algorithms are capable of solving very complex problems almost optimally, the combined solution of these problems should be further investigated. The fifth key aspect, that needs to be explored to make RL algorithms useful for controlling fleets of AMR, is the consideration of heterogeneous fleets using different types of robots. For this survey, only four sources ([59,61,68,76]) could be found that deal with the use of heterogeneous fleets. The flexible use of heterogeneous fleets is particularly important as it allows companies to solve different challenges simultaneously and easily integrate the agents needed for this purpose. In order to use these algorithms in industrial areas, such as logistics and production, a flexible integration of different agents is necessary, due to the complexity of the environment. The need for interoperability for the industry enabled by the control of heterogeneous fleets has been described in the context of [73].

7.1.3. Dynamic Environments

In this context, dynamic environments are characterized as systems, whose states are changing frequently and quickly, often randomly and unpredictable. For example, it can be the case when objects or persons move without fixed paths throughout a building at the same time as the AMRs. Dynamic environments can become major challenges for algorithms because they cannot be planned and thus require a high degree of flexibility. To tackle this challenge, in addition to a more detailed exploration of key aspects (2) robustness and (5) enabling heterogeneous fleets, the successful handling of deadlocks (6) is particularly important. The occurrence of deadlocks usually happens unpredictably and ensures that the algorithm must react within the shortest possible time to avoid delays. It is the reason why real world applications in fields of logistics and production involve very dynamic environments. A control algorithm must be able to react to blocked paths in order to work successfully. While some authors have already studied orientation in such environments [42,43,57,58], a solution for this problem has only been considered in [42].

Future work should place a higher priority on randomly occurring events, as well as a higher degree of randomization of the training environments.

7.2. Solutions

There are several potential solutions to overcome the shortcomings, described previously. This section will discuss possible solutions for dealing with key aspects (1)–(6).

1. Overcoming the "sim-to-real gap"
   (a) *Domain Randomization*
   It is necessary to increase the randomization of the training to better prepare the algorithm for random events that may occur in reality with increased probability. The more the training scenario is randomized, the lower is the chance the algorithm will experience a scenario in reality that it has not yet experienced in training. In order to implement this, there are different approaches, ranging from simple (active) domain randomization of the training environment [77,78] to noise in the data [79] in order to simulate errors from sensors, latencies or other influences.
   (b) *Real data and real applications*
   Another important step is the use of real data beyond laboratory settings. It is necessary to generate scenarios, which are also performed in reality [80]. For this purpose, detailed process data must be generated within a simulation that represent reality as closely as possible. This data must be as close as to reality that the gap between simulation and reality is as small as possible. In order to achieve this, a close cooperation with the industry is necessary to integrate data from already existing systems into the training. Alternatively, it is possible to use real scenarios whose complexity are increased depending on the success rate within the scenario. The aim for this approach is to increase the complexity step by step beyond that of a laboratory setting to enable use in a real-world context.
   (c) *State space reduction*
   As a further step to enable the transfer of the simulation into a real use case, one could adjust the state space. For example, one could create a grid within the agents move with the help of motion primitives [81,82]. Depending on the fineness of the grid, this would reduce the complexity of the environment on the one hand, and on the other hand reduce movement sensitivity, since the motion primitives could be programmed in advance. Another possibility to limit the state space is to change the observations. Compared to a camera-based input, a laser-based or distance-based input is much smaller and can reduce the reality gap [83]. If additional localization information is used, a better result can probably be achieved with less computing power.
2. Robust algorithms
   (a) *Reward shaping*
   One way to increase robustness is through reward shaping. By reward shaping, problems like dense rewards can be solved. For example, the reward structure could be revised in such a way that there are additional rewards for approaching goals or staying away from obstacles as well as the use of customized reward functions. However, adapting the reward function also takes away some of the generalizability of the algorithm, which is why a certain expertise is necessary to develop such a concept in the best possible way. In order to this and automate reward shaping, an approach has been developed by [84].
   (b) *Imitational learning*
   Another way to improve the robustness of the algorithm is to use expert knowledge. With the help of imitational learning, as it is used in [41] among others, data generated by experts can be used for training the algorithm. This

is also a support for data efficiency (3) and enables the algorithm to learn, how to act correctly in particularly error-prone situations.

3. Data efficiency
   (a) *Improving memory/knowledge*
   One possibility to increase data efficiency is to improve the memory of the algorithm. While many methods already have simple experience replay or use LSTM, there is still a high potential for improvement. In this case, one could fall back on procedures like the double experience replay [85], as well as prioritized experience replay [86] or making use of prior knowledge like the authors of [38]. With the help of such methods a higher generalizability is possible even with less data.
   (b) *Curriculum learning*
   CL [43] is a method which is used to give an algorithm step by step more complex tasks to enable a faster and better training success. Especially for sparse rewards this method is useful. Since the probability of completing a simple task is significantly higher than for a complex one, the algorithm learns the necessary correlations much faster. This knowledge can be transferred back to more complex scenarios, ensuring faster learning and more data-efficient training.
4. Combining different fields of application
   (a) *Tuning the reward structure*
   As simple as it may sound, the best way to combine different applications is to integrate them into the training. In doing so, it is necessary that the algorithmic architecture is designed at the beginning of the training in such a way that two or more problems can be solved with it simultaneously. The reward structure is one of the biggest challenges. Since each sub-problem has its own reward structure, it must be ensured that the rewards are set to such a degree that the overall problem is solved optimally and not one of the sub-problems.
   (b) *Reducing redundant information in problems*
   Another method to improve the combination of different problems is to choose the correct input information. In order to combine several problems, they must be interdependent at first. However, this dependency on each other ensures that the information, that can be obtained from each problem also has dependencies on each other. For example, similar variables are of interest for scheduling and routing in a warehouse, such as the position of the object to be collected, the position to which it should be delivered and the layout of the warehouse itself. Accordingly, the algorithm needs this information only once and not once for each sub-problem. In the conceptual design of the algorithm it is necessary to find these dependencies and exclude redundant information. Due to the increasing complexity of the whole problem, it is especially important to avoid high computational efforts.
5. Enabling heterogeneous fleets
   (a) *Decentralized Training*
   As already mentioned in the paper, decentralized training is particularly flexible, which results in a better scalability for multiple agents (see Table 2) and an easy integration of different agents. Consequently for the implementation of heterogeneous fleets, decentralized training of different agents is a central concept and necessary to enable the easy integration of additional vehicles.
   (b) *Communication standard*
   The decentralized training of individual agents also requires a central communication interface that follows a standard [73]. Since different agents are combined in heterogeneous fleets, consideration must be given to the fact that these also bring different fundamentals. In addition to different transport

characteristics and dimensions, the agents have different drive procedures and localization methods. In order to enable a common use of such agents, it is necessary to create a communication standard in which the information received from the agents can be processed in such a standardized way that an easy integration of the agents as well as a successful completion of tasks is possible.

(c) *Standard for better comparability*
In order to develop such a standard [73], a certain comparability of the agents is necessary. During the research of this paper, it was noticed that different algorithms have been trained in different scenarios. Accordingly, a comparability of the individual algorithms among each other is hardly possible in most cases. In order to enable a heterogeneous fleet in the best possible way and to be able to integrate a communication standard, such comparability is necessary. Therefore, a further solution for the implementation of heterogeneous fleets is the development and definition of a training standard. With the help of this standard different agents are on the same level (experience wise) and an integration into the system is simplified.

6. Handling deadlocks

(a) *Deadlock detection and communication*
One technique to better handle deadlocks is not only to detect them, but also to communicate within the fleet [45,54]. In the event that an agent detects a deadlock or other disturbance within the test area, the information should be distributed within the fleet. However, only to the agents that are affected by this disturbance, in order to save unnecessary computing effort. This allows other agents to adjust their route early and thus reduces the chance that a deadlock will cause a significant delay in the completion of orders.

(b) *Subgoal Integration*
By integrating sub-goals [57] into the flow of the algorithm, the route to the destination can be planned in a more fine-grained way. Such fine-grained planning subsequently makes it possible to easily bypass blocked driving areas and adjust the route accordingly. If, for example, a single route is disrupted, the algorithm does not need to re-plan the entire route, but can simply find a alternative route for the blocked location and then return to the original route. This also ensures that the route change has less impact on the rest of the fleet.

RL algorithms already show high potential in the area of fleet control of AMR. For this reason, further research into these algorithms is promising. If the six key aspects mentioned above are addressed, the integration of such systems in real operations is only a matter of time.

8. Conclusions

Over the past few years, a number of developments have improved the control of AMR-fleets and thus expanded their areas of application. RL methods could now offer the next step in this development. In simulation environments, these algorithms are already capable of solving problems more efficiently, controlling larger fleets and navigating dynamic environments. This paper provides a comprehensive and systematic overview of research on fleet control of AMR using RL. Control approaches and application areas for the respective algorithms have been presented. Subsequently, three challenges have been evaluated, which in turn have been divided into six key aspects and possible solutions for these have been discussed. We expect that this review will help researchers to get an overview of current solutions for fleet control, to further improve the state of the art, and to gradually enable their use in an industrial context.

Author Contributions: Conceptualization, M.W. and J.H.; methodology, M.W., J.H. and J.K.; validation, M.W.; formal analysis, M.W.; investigation, M.W.; resources, M.W.; data curation, M.W.;

writing—original draft preparation, M.W.; writing—review and editing, M.W., J.H. and J.K.; visualization, M.W.; supervision, J.H. and J.K.; project administration, J.K.; funding acquisition, J.K. All authors have read and agreed to the published version of the manuscript.

Funding: Publishing fees funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation)—Projektnummer 491268466 and the Hamburg University of Technology (TUHH) in the funding programme *Open Access Publishing*.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data of the literature search are available on request.

Conflicts of Interest: The authors declare no conflict of interest.

Sample Availability: Samples of the compounds are available from the authors.

References

1. International Federation of Robotics. *World Robotics Report 2020*; International Federation of Robotics: Frankfurt am Main, Germany, 2020.
2. International Federation of Robotics. *Robot Sales Rise Again*; International Federation of Robotics: Frankfurt am Main, Germany, 2021.
3. The Logistics IQ. *AGV-AMR Market Map 2021*; The Logistics IQ: New Delhi, India, 2021.
4. Steeb, R.; Cammarata, S.; Hayes-Roth, F.A.; Thorndyke, P.W.; Wesson, R.B. Distributed Intelligence for Air Fleet Control. In *Readings in Distributed Artificial Intelligence*; Elsevier: Amsterdam, The Netherlands, 1981; pp. 90–101. https://doi.org/10.1016/b978-0-934613-63-7.50011-5.
5. Naumov, V.; Kubek, D.; Więcek, P.; Skalna, I.; Duda, J.; Goncerz, R.; Derlecki, T. Optimizing Energy Consumption in Internal Transportation Using Dynamic Transportation Vehicles Assignment Model: Case Study in Printing Company. *Energies* **2021**, *14*, 4557. https://doi.org/10.3390/en14154557.
6. Alexovič, S.; Lacko, M.; Bačík, J.; Perduková, D. *Introduction into Autonomous Mobile Robot Research and Multi Cooperation*; Springer, Cham, Schweiz, 2021 ; pp. 326–336.
7. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354.
8. Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; Ribas, R.; et al. Solving Rubik's Cube with a robot hand. *arXiv* **2019**, arXiv:1910.07113.
9. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2011 .
10. Kiumarsi, B.; Vamvoudakis, K.G.; Modares, H.; Lewis, F.L. Optimal and Autonomous Control Using Reinforcement Learning: A Survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2018**, *29*, 2042–2062. https://doi.org/10.1109/TNNLS.2017.2773458.
11. Vázquez-Canteli, J.R.; Nagy, Z. Reinforcement learning for demand response: A review of algorithms and modeling techniques. *Appl. Energy* **2019**, *235*, 1072–1089. https://doi.org/10.1016/j.apenergy.2018.11.002.
12. Pandey A.; Pandey S.; Parhi, D.R. Mobile Robot Navigation and Obstacle Avoidance Techniques: A Review. *Int. Robot. Autom. J.* **2017**, *2*, 96–105. https://doi.org/10.15406/iratj.2017.02.00023.
13. Panchpor, A.A.; Shue, S.; Conrad, J.M. A survey of methods for mobile robot localization and mapping in dynamic indoor environments. In Proceedings of the 2018 Conference on Signal Processing and Communication Engineering Systems (SPACES), Vaddeswaram, India, 4–5 January 2018; pp. 138–144. https://doi.org/10.1109/SPACES.2018.8316333.
14. Shabbir, J.; Anwer, T. A Survey of Deep Learning Techniques for Mobile Robot Applications. *arXiv* **2018**, arXiv:1803.07608.
15. Farazi, N.P.; Ahamed, T.; Barua, L.; Zou, B. Deep Reinforcement Learning and Transportation Research: A Comprehensive Review. *arXiv* **2020**, arXiv:2010.06187.
16. Singh, P.; Tiwari, R.; Bhattacharya, M. Navigation in Multi Robot system using cooperative learning: A survey. In Proceedings of the 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), New Delhi, India, 11–13 March 2016; pp. 145–150. https://doi.org/10.1109/ICCTICT.2016.7514569.
17. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. https://doi.org/10.1109/TCYB.2020.2977374.
18. OroojlooyJadid, A.; Hajinezhad, D. A Review of Cooperative Multi-Agent Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1908.03963.
19. Rizk, Y.; Awad, M.; Tunstel, E.W. Decision Making in Multiagent Systems: A Survey. *IEEE Trans. Cogn. Dev. Syst.* **2018**, *10*, 514–529. https://doi.org/10.1109/tcds.2018.2840971.
20. Madridano, Á.; Al-Kaff, A.; Martín, D. Trajectory Planning for Multi-Robot Systems: Methods and Applications. *Expert Syst. Appl.* **2021**, *173*, 114660. https://doi.org/10.1016/j.eswa.2021.114660.
21. Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; Levine, S. How to Train Your Robot with Deep Reinforcement Learning; Lessons We've Learned. *Int. J. Robot. Res.* **2021**, *7*, 027836492098785. https://doi.org/10.1177/0278364920987859.

22. Xiao, X.; Liu, B.; Warnell, G.; Stone, P. Motion Control for Mobile Robot Navigation Using Machine Learning: A Survey. *arXiv* **2020**, arXiv:2011.13112.
23. Jiang, H.; Wang, H.; Yau, W.Y.; Wan, K.W. A Brief Survey: Deep Reinforcement Learning in Mobile Robot Navigation. In Proceedings of the 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA), Kristiansand, Norway, 9–13 November 2020; pp. 592–597. https://doi.org/10.1109/ICIEA48937.2020.9248288.
24. Aradi, S. Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 740–759. https://doi.org/10.1109/tits.2020.3024655.
25. Zhu, K.; Zhang, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **2021**, *26*, 674–691. https://doi.org/10.26599/tst.2021.9010012.
26. Tranfield, D.; Denyer, D.; Smart, P. Towards a Methodology for Developing Evidence-Informed Management Knowledge by Means of Systematic Review. *Br. J. Manag.* **2003**, *14*, 207–222. https://doi.org/10.1111/1467-8551.00375.
27. Kaelbling, L.P.; Littman, M.L.; Cassandra, A.R. Planning and acting in partially observable stochastic domains. *Artif. Intell.* **1998**, *101*, 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X.
28. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
29. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
30. Lu, C.; Long, J.; Xing, Z.; Wu, W.; Gu, Y.; Luo, J.; Huang, Y. Deep Reinforcement Learning for Solving AGVs Routing Problem. *Int. Conf. Verif. Eval. Comput. Commun. Syst.* **2020**, *12519*, 222–236.
31. Liu, H.; Hyodo, A.; Akai, A.; Sakaniwa, H.; Suzuki, S. Action-limited, Multimodal Deep Q Learning for AGV Fleet Route Planning. In Proceedings of the Proceedings of the 5th International Conference on Control Engineering and Artificial Intelligence, Sanya, China, 14–16 January 2021; Zhang, D., Ed.; Association for Computing Machinery: New York, NY, USA, 2021; pp. 57–62. https://doi.org/10.1145/3448218.3448219.
32. He, C.; Wan, Y.; Gu, Y.; Lewis, F.L. Integral Reinforcement Learning-Based Multi-Robot Minimum Time-Energy Path Planning Subject to Collision Avoidance and Unknown Environmental Disturbances. *IEEE Control. Syst. Lett.* **2021**, *5*, 983–988.
33. Zhi, J.; Lien, J.M. Learning to Herd Agents Amongst Obstacles: Training Robust Shepherding Behaviors Using Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4163–4168. https://doi.org/10.1109/LRA.2021.3068955.
34. Meerza, S.I.A.; Islam, M.; Uzzal, M.M. Q-Learning Based Particle Swarm Optimization Algorithm for Optimal Path Planning of Swarm of Mobile Robots. In Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT 2019), Dhaka, Bangladesh, 3–5 May 2019; pp. 1–5. https://doi.org/10.1109/ICASERT.2019.8934450.
35. Wu, J.; Song, C.; Ma, J.; Wu, J.; Han, G. Reinforcement Learning and Particle Swarm Optimization Supporting Real-Time Rescue Assignments for Multiple Autonomous Underwater Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 6807–6820.
36. Wang, M.; Zeng, B.; Wang, Q. Research on Motion Planning Based on Flocking Control and Reinforcement Learning for Multi-Robot Systems. *Machines* **2021**, *9*, 77. https://doi.org/10.3390/machines9040077.
37. Vitolo, E.; Miguel, A.S.; Civera, J.; Mahulea, C. Performance Evaluation of the Dyna-Q algorithm for Robot Navigation. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018; Vogel-Heuser, B., Ed., IEEE: New York, NY, USA, 2018; pp. 322–327. https://doi.org/10.1109/COASE.2018.8560457.
38. Li, B.; Liang, H. Multi-Robot Path Planning Method Based on Prior Knowledge and Q-learning Algorithms. *J. Physics: Conf. Ser.* **2020**, *1624*, 042008. https://doi.org/10.1088/1742-6596/1624/4/042008.
39. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-Robot Path Planning Method Using Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 3057. https://doi.org/10.3390/app9153057.
40. Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.K.S.; Koenig, S.; Choset, H. PRIMAL: Pathfinding via Reinforcement and Imitation Multi-Agent Learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2378–2385. https://doi.org/10.1109/LRA.2019.2903261.
41. Damani, M.; Luo, Z.; Wenzel, E.; Sartoretti, G. $PRIMAL_2$: Pathfinding Via Reinforcement and Imitation Multi-Agent Learning—Lifelong. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2666–2673. https://doi.org/10.1109/LRA.2021.3062803.
42. Wang, B.; Liu, Z.; Li, Q.; Prorok, A. Mobile Robot Path Planning in Dynamic Environments through Globally Guided Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6932–6939.
43. Liu, Z.; Chen, B.; Zhou, H.; Koushik, G.; Hebert, M.; Zhao, D. MAPPER: Multi-Agent Path Planning with Evolutionary Reinforcement Learning in Mixed Dynamic Environments. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 11748–11754.
44. Ma, Z.; Luo, Y.; Pan, J. Learning Selective Communication for Multi-Agent Path Finding. *IEEE Robot. Autom. Lett.* **2022**, *7*, 1455–1462. https://doi.org/10.1109/lra.2021.3139145.
45. Ma, Z.; Luo, Y.; Ma, H. Distributed Heuristic Multi-Agent Path Finding with Communication. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021. https://doi.org/10.1109/icra48506.2021.9560748.
46. Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F. Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14413–14423.
47. Portugal, D.; Rocha, R.P. Cooperative multi-robot patrol with Bayesian learning. *Auton. Robot.* **2016**, *40*, 929–953. https://doi.org/10.1007/s10514-015-9503-7.

48. Ajabshir, V.B.; Guzel, M.S.; Bostanci, E. A Low-Cost Q-Learning-Based Approach to Handle Continuous Space Problems for Decentralized Multi-Agent Robot Navigation in Cluttered Environments. *IEEE Access* **2022**, *10*, 35287–35301. https://doi.org/10.1109/access.2022.3163393.
49. Chen, Y.F.; Liu, M.; Everett, M.; How, J.P. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 285–292. https://doi.org/10.1109/icra.2017.7989037.
50. Fan, T.; Long, P.; Liu, W.; Pan, J. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int. J. Robot. Res.* **2020**, *39*, 856–892. https://doi.org/10.1177/0278364920916531.
51. Yao, S.; Chen, G.; Pan, L.; Ma, J.; Ji, J.; Chen, X. Multi-Robot Collision Avoidance with Map-based Deep Reinforcement Learning. In Proceedings of the 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020.
52. Fan, T.; Long, P.; Liu, W.; Pan, J. Fully Distributed Multi-Robot Collision Avoidance via Deep Reinforcement Learning for Safe and Efficient Navigation in Complex Scenarios. *arXiv* **2018**, arXiv:1808.03841.
53. Long, P.; Fan, T.; Liao, X.; Liu, W.; Zhang, H.; Pan, J. Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018. https://doi.org/10.1109/icra.2018.8461113.
54. Zhai, Y.; Ding, B.; Liu, X.; Jia, H.; Zhao, Y.; Luo, J. Decentralized Multi-Robot Collision Avoidance in Complex Scenarios With Selective Communication. *IEEE Robot. Autom. Lett.* **2021**, *6*, 8379–8386.
55. Semnani, S.H.; Liu, H.; Everett, M.; de Ruiter, A.; How, J.P. Multi-Agent Motion Planning for Dense and Dynamic Environments via Deep Reinforcement Learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3221–3226. https://doi.org/10.1109/lra.2020.2974695.
56. Chen, W.; Zhou, S.; Pan, Z.; Zheng, H.; Liu, Y. Mapless Collaborative Navigation for a Multi-Robot System Based on the Deep Reinforcement Learning. *Appl. Sci.* **2019**, *9*, 4198.
57. Brito, B.; Everett, M.; How, J.P.; Alonso-Mora, J. Where to go Next: Learning a Subgoal Recommendation Policy for Navigation in Dynamic Environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 4616–4623. https://doi.org/10.1109/lra.2021.3068662.
58. Han, R.; Chen, S.; Hao, Q. Cooperative Multi-Robot Navigation in Dynamic Environment with Deep Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020.
59. Vera, José Manuel, und Andres G. Abad. Deep Reinforcement Learning for Routing a Heterogeneous Fleet of Vehicles. In Proceedings 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guayaquil, Ecuador, 11–15 November 2019; pp. 1–6.
60. Google Inc. *Google's Optimization Tools (Or-Tools)*; Google Inc.: Mountain View, CA, USA, 2019.
61. Schperberg, A.; Tsuei, S.; Soatto, S.; Hong, D. SABER: Data-Driven Motion Planner for Autonomously Navigating Heterogeneous Robots. *IEEE Robot. Autom. Lett.* **2021**, *6*, 8086–8093. https://doi.org/10.1109/lra.2021.3103054.
62. Zhang, Y.; Qian, Y.; Yao, Y.; Hu, H.; Xu, Y. Learning to Cooperate: Application of Deep Reinforcement Learning for Online AGV Path Finding. *19th Int. Conf. Auton. Agents Multiagent Syst.* **2020**, 2077–2079.
63. Lin, K.; Zhao, R.; Xu, Z.; Zhou, J.. Efficient Large-Scale Fleet Management via Multi-Agent Deep Reinforcement Learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1774–1783.
64. Köksal Ahmed, E.; Li, Z.; Veeravalli, B.; Ren, S. Reinforcement learning-enabled genetic algorithm for school bus scheduling. *J. Intell. Transp. Syst.* **2022**, *26*, 269–283.
65. Qi, Q.; Zhang, L.; Wang, J.; Sun, H.; Zhuang, Z.; Liao, J.; Yu, F.R. Scalable Parallel Task Scheduling for Autonomous Driving Using Multi-Task Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 13861–13874.
66. Zhang, L.; Qi, Q.; Wang, J.; Sun, H.; Liao, J. Multi-task Deep Reinforcement Learning for Scalable Parallel Task Scheduling. In Proceedings of the 2019 IEEE International Conference on Big Data, Los Angeles, CA, USA, 9–12 December 2019; Baru, C., Ed.; IEEE: New York, NY, USA, 2019.
67. Xue, T.; Zeng, P.; Yu, H. A reinforcement learning method for multi-AGV scheduling in manufacturing. In Proceedings of the 2018 IEEE International Conference on Industrial Technology (ICIT), Lyon, France, 20–22 February 2018; pp. 1557–1561. https://doi.org/10.1109/ICIT.2018.8352413.
68. Zhang, C.; Odonkor, P.; Zheng, S.; Khorasgani, H.; Serita, S.; Gupta, C.; Wang, H. Dynamic Dispatching for Large-Scale Heterogeneous Fleet via Multi-agent Deep Reinforcement Learning. In Proceedings of the 2020 IEEE International Conference on Big Data (Big Data), Atlanta, GA, USA, 10–13 December 2020. https://doi.org/10.1109/bigdata50022.2020.9378191.
69. Elfakharany, A.; Ismail, Z.H. End-to-End Deep Reinforcement Learning for Decentralized Task Allocation and Navigation for a Multi-Robot System. *Appl. Sci.* **2021**, *11*, 2895.
70. Li, M.P.; Sankaran, P.; Kuhl, M.E.; Ptucha, R.; Ganguly, A.; Kwasinski, A. Task Selection by Autonomous Moblie Rrobots in a warhouse using Deep Reinforcement Learning. In Proceedings of the 2019 Winter Simulation Conference (WSC)E, National Harbor, MD, USA, 8–11 December 2019; pp. 680–689.
71. Agrawal, A.; Won, S.J.; Sharma, T.; Deshpande, M.; McComb, C. A multi-agent reinforcement learning framework for intelligent manufactoring with autonomous mobile robots. *Proc. Des. Soc.* **2021**, *1*, 161–170. https://doi.org/10.1017/pds.2021.17.

72. Lütjens, B.; Everett, M.; How, J.P. Certified Adversarial Robustness for Deep Reinforcement Learning. In Proceedings of the Conference on Robot Learning, Virtual, 16–18 November 2020; Kaelbling, L.P., Kragic, D., Sugiura, K., Eds.; PMLR: Cambridge, MA, USA: 2020; Volume 100, pp. 1328–1337.
73. Verband der Automobilindustrie. *Interface for the Communication between Automated Guided Vehicles (AGV) and a Master Control: VDA5050*; VDA: Berlin, Germany, June 2020.
74. Weinstock, C.B.; Goodenough, J.B. *On System Scalability*; Defense Technical Information Center: Fort Belvoir, VA, USA, 2006. https://doi.org/10.21236/ada457003.
75. Wegner, P. Interoperability. *ACM Comput. Surv.* **1996**, *28*, 285–287. https://doi.org/10.1145/234313.234424.
76. Qin, W.; Zhuang, Z.; Huang, Z.; Huang, H. A novel reinforcement learning-based hyper-heuristic for heterogeneous vehicle routing problem. *Comput. Ind. Eng.* **2021**, *156*, 107252.
77. Mehta, B.; Diaz, M.; Golemo, F.; Pal, C.J.; Paull, L. Active Domain Randomization. *Conf. Robot. Learn.* **2020**, *100*, 1162–1176.
78. Vuong, Q.; Vikram, S.; Su, H.; Gao, S.; Christensen, H.I. How to Pick the Domain Randomization Parameters for Sim-to-Real Transfer of Reinforcement Learning Policies? *arXiv* **2019**, arXiv:1903.11774.
79. He, Z.; Rakin, A.S.; Fan, D. Certified Adversarial Robustness with Additive Noise; In Proceedings of the 32th Conference Advances in neural information processing systems, Vancouver, BC, Canada, 8–14 December 2019.
80. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 737–744. https://doi.org/10.1109/SSCI47803.2020.9308468.
81. Stulp, F.; Theodorou, E.A.; Schaal, S. Reinforcement Learning With Sequences of Motion Primitives for Robust Manipulation. *IEEE Trans. Robot.* **2012**, *28*, 1360–1370. https://doi.org/10.1109/tro.2012.2210294.
82. Sledge, I.J.; Bryner, D.W.; Principe, J.C. Annotating Motion Primitives for Simplifying Action Search in Reinforcement Learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, 1–20. https://doi.org/10.1109/TETCI.2021.3132365.
83. Shi, H.; Shi, L.; Xu, M.; Hwang, K.S. End-to-End Navigation Strategy With Deep Reinforcement Learning for Mobile Robots. *IEEE Trans. Ind. Inform.* **2020**, *16*, 2393–2402. https://doi.org/10.1109/tii.2019.2936167.
84. Chiang, H.T.L.; Faust, A.; Fiser, M.; Francis, A. Learning Navigation Behaviors End-to-End With AutoRL. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2007–2014. https://doi.org/10.1109/LRA.2019.2899918.
85. Wu, J.; Wang, R.; Li, R.; Zhang, H.; Hu, X. Multi-critic DDPG Method and Double Experience Replay. In Proceedings of the 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 7–10 October 2018. https://doi.org/10.1109/smc.2018.00039.
86. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2016**, arXiv:1511.05952. https://doi.org/10.48550/arXiv.1511.05952.