# Imitation-Guided World Models for Multi-Agent Train Rescheduling

Max Bourgeat[1,2][0009-0000-1717-0522], Antoine Legrain[1,2,3][0000-0003-2903-9593], and Quentin Cappart[1,2,4][0000-0002-8742-0774]

[1] Polytechnique Montreal, Montreal, Canada
[2] CIRRELT, Montreal, Canada
[3] GERAD, Montreal, Canada
[4] UCLouvain, Louvain-la-Neuve, Belgium
`{max.bourgeat,quentin.cappart,a.legrain}@polymtl.ca`

**Abstract.** Managing railway disruptions is a complex multi-agent routing problem where a single train failure can propagate delays across the network. Traditional approaches rely on heuristic optimization solvers, which are effective but assume access to a global system view and require substantial expert design, limiting their applicability and generalization. Reinforcement learning (RL) offers an alternative by learning adaptive strategies from interactions with the environment. In this paper, we show that none of the available paradigms is sufficient in isolation: (i) heuristic solvers encode valuable global expertise but cannot be deployed directly, (ii) world models improve sample efficiency but struggle to leverage expert knowledge, and (iii) pure RL can adapt policies but often lacks stability without strong guidance. We propose a hybrid framework that integrates the strengths of these approaches. First, imitation learning transfers knowledge from a global expert solver to initialize a neural policy. Then, model-based RL fine-tunes this policy using the DreamerV2 world model to enhance generalization and responsiveness to local perturbations. Our method builds on the Multi-Agent Model-Based Architecture (MAMBA) to model agent interactions and addresses the challenge of transferring expertise from global solvers to decentralized agents operating on local latent observations. Experiments on a train rescheduling problem using the Flatland environment show that our method outperforms MAMBA, improving performance by up to 23% on difficult instances. This highlights the benefit of combining imitation learning with world-model-based multi-agent RL for complex transportation networks. The code is available here `https://github.com/corail-research/Imitation-Guided_World_Models`.

**Keywords:** Multi-Agent Reinforcement Learning · Imitation Learning

## 1 Introduction

*Multi-agent routing* [41] consists in coordinating several agents moving within a network to reach their destinations while minimizing conflicts and delays. This problem appears in logistics [28,22], robotics [16], and public transportation [12], and is characterized by strong combinatorial complexity due to agent interdependencies. Efficient methods are crucial for reducing operational costs and improving the resilience of large-scale systems. In railway networks, disruptions further increase this complexity. A single train breakdown blocks its track segment, potentially propagating delays across the network. Rescheduling trains to mitigate these cascading effects is, therefore, central to modern railway operations. The *Flatland* environment [30], developed with several European railway companies, provides a realistic platform for studying such scenarios. Traditional approaches rely on heuristic or search-based solvers [37,42,3]. While effective, they require handcrafted heuristics tailored to each instance and assume access to a global system view, an unrealistic condition since trains typically observe only local surroundings. Reinforcement learning (RL) [43] offers an alternative by learning from interaction. However, model-free methods often suffer from poor sample efficiency [29], limited generalization [4], and instability in multi-agent settings [11]. Model-based approaches, such as DREAMER [9], improve efficiency by learning latent dynamics but remain limited by model inaccuracies [45] and a performance gap with expert solvers [25]. *Imitation learning* (IL) [15] mitigates exploration challenges by leveraging expert demonstrations. Yet, aligning expert trajectories defined in global real-state space with the local latent space of world models is non-trivial [5].

We propose a hybrid methodology that combines IL and model-based RL for train rescheduling. A neural policy is first initialized from an expert solver [21], then refined using MAMBA system, a multi-agent world model based on DreamerV2 [10] with Transformer-based communication [47]. Our key contribution is an adaptation strategy enabling the transfer of global solver expertise into a decentralized latent setting. Experiments in Flatland show that our method outperforms state-of-the-art baselines, achieving up to 23% improvement on difficult instances. This demonstrates the value of combining imitation and world-model-based RL for scalable decision-making in complex transportation networks.

## 2 Related Work

*Multi-Agent reinforcement learning* (MARL) has become a central research direction, driven by the need to handle non-stationarity, coordination, and competition among multiple interacting agents. Overviews of deep learning methods for addressing scalability, decentralized decision-making, and efficient communication are provided by few recent surveys [14,11]. Beyond model-free approaches, several works have investigated the use of predictive models, or world models, in MARL. For instance, MAMBA [6] coupled a Transformer-based communication [47] between agents with DreamerV2 [10]. This world model introduced learned dynamics models to generate imagined rollouts, thereby improving sample efficiency and reducing reliance on expensive environment interactions. Another architecture, referred to as *global-aware world model* [39] adopted centralized training with decentralized execution, leveraging Transformers to aggregate local observations into a global latent representation. Other methods also explored local predictive structures, such as *models as agents* [52] which optimize multi-step predictions of interactive local models, or hierarchical frameworks (e.g., [48], which employs bi-level latent-variable world models). Together, these methods highlight the potential of world models to improve stability, scalability, and coordination. Yet, they often underperform as the number of agents increases.

On the other hand, *imitation learning* offers a complementary paradigm where agents learn from demonstrations rather than sparse or delayed rewards [46]. In the multi-agent setting, imitation learning must also capture inter-agent dependencies. Recent approaches explicitly model joint behaviors, for instance, with copula-based methods [49] or adversarial extensions of *generative adversarial imitation learning* [13] to cooperative and competitive scenarios [40]. Hybrid strategies further combine IL and RL, such as self-imitation frameworks [31] or offline-to-online training strategies like advantage-weighted regression [32], aiming to accelerate convergence while ensuring robustness. However, most methods assume access to large-scale demonstrations or simplified coordination, which may not hold in real-world systems.

Focusing now on the specific railway domain, the Flatland environment was introduced during the NeurIPS 2020 competition [30] as a benchmark for large-scale multi-agent routing and rescheduling. It provides a simplified yet realistic simulation where multiple trains must coordinate under infrastructure and safety constraints. Flatland also includes stochastic disruptions, such as random breakdowns, making it suitable for studying robust decision-making under uncertainty. Beyond the baseline methods proposed in the competition, the winning solution, *scalable rail planning and replanning* [21], combines Multi-Agent PathFinding (MAPF) techniques, large neighborhood search, safe interval path planning, simulated annealing, minimum-communication policies, and parallel computation, demonstrating strong performance with up to 400 trains. Since then, subsequent research has extended Flatland to practical scheduling. For instance, LCPPO [53] introduces a policy-gradient algorithm tailored to large-scale railway networks, while *curriculum-driven continual DQN expansion* [17] addresses stability-plasticity trade-offs through continual learning. But there is also a rich literature outside Flatland: for instance, Cappart and Schaus [3] proposes a CP model using time-interval variables for real-time rescheduling in a Belgian station, outperforming greedy operator strategies; *Reinforcement learning in railway timetable rescheduling* explores RL methods off-line to enable fast online dispatching under delays [55]; Liao et al. [23] combined deep learning and genetic algorithms to optimize metro energy consumption under random disturbances [23]. Finally, recent models also integrate learning-based model predictive control to adapt rolling stock and train compositions [26].

Taken together, these works highlight that flexibility, realistic constraints (e.g. variable speeds, train composition, or disruptions) and the ability to compute solutions rapidly in real-world contexts are crucial requirements for MARL or RL applied to this domain.

## 3 Background

*World models* have emerged as a powerful paradigm in model-based reinforcement learning (MBRL) [43], where an agent learns a latent dynamics model of the environment and uses it for planning and policy optimization. Formally, the model consists of a latent state-space model defined by the following conditional distributions:

$$z_t \sim p_\theta(\cdot \mid z_{t-1}, a_{t-1}), \tag{1}$$
$$o_t \sim p_\theta(\cdot \mid z_t), \tag{2}$$
$$r_t \sim p_\theta(\cdot \mid z_t, a_t), \tag{3}$$

where $z_t$ is a latent state, $o_t$ an observation, and $r_t$ a reward obtained at a timestep $t$. Each component is implemented as a neural network $p_\theta$, where $\theta$ denotes the joint vector of all their parameters. A policy $\pi_\phi(a_t \mid z_t)$ is trained entirely in latent space by maximizing the expected discounted return:

$$J(\pi_\phi) = \mathbb{E}_{z_0 \sim p_\theta, a_t \sim \pi_\phi}\left[\sum_{t=0}^{T} \gamma^t r(z_t, a_t)\right], \tag{4}$$

where $\phi$ are our policy's parameters, $z_0$ the initial latent space following the initial distribution $p_\theta$, $\gamma \in (0, 1]$ the discount factor which indicates how much weight we give to short-term reward compared to long-term reward and $T$ the time horizon in the latent space. The *Dreamer* family of algorithms [9,10] demonstrated that training on imagined trajectories in latent space yields state-of-the-art performance while being computationally efficient. In particular, a *recurrent state-space model* [8] with categorical latent variables stabilizing long-horizon predictions and improving scalability is employed by DreamerV2.

### 3.1 Multi-Agent Reinforcement Learning

*Multi-agent reinforcement learning* (MARL) studies sequential decision-making with multiple agents interacting in a shared environment. This setting can be formalized as a Markov game featuring $n$ agents:

$$\mathcal{M} = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^n, T, \{r_i\}_{i=1}^n, \gamma \rangle, \tag{5}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}_i$ is the action space of agent $i$, and the transition dynamics follow

$$s' \sim T(s' \mid s, a_1, \ldots, a_n), \tag{6}$$

where $s \in \mathcal{S}$ and $a_i \in \mathcal{A}_i, \forall i \in \{1, ..., n\}$. Each agent $i$ receives a reward $r_i(s, a_1, \ldots, a_n)$ and optimizes its policy $\pi_\phi^i(a_i \mid o_i)$, a neural network parameterized by $\phi$, based on local observations $o_i$. As before, $\gamma$ is the discount factor. Most of the time, if a world model is used then the policy is trained using the actor-critic approach [20]. In actor-critic methods, the policy (actor) is updated using feedback provided by a value function (critic), which estimates the expected return of state–action pairs. This framework combines the exploration capabilities of policy gradient methods [44,51,36] with the stability of value-based estimation [50,29,2]. In the *centralized training with decentralized execution* (CTDE) paradigm [7], critics are trained with global information while policies rely solely on local observations. A typical policy gradient update is given by:

$$\nabla_\phi J(\pi_\phi) = \mathbb{E}_{s\sim\mathcal{D},a\sim\pi_\theta}\big[\nabla_\phi \log \pi_\phi(a \mid o)\, Q(s,a)\big], \tag{7}$$

where $Q(s,a)$ is the value function output by the critic, it may be learned with centralized knowledge during training and $\mathcal{D}$ is the dataset from which the states are sampled [27,24,44]. To improve sample efficiency, model-based MARL approaches extend latent dynamics models to multi-agent settings. For example, scalable frameworks such as MAMBA [6] introduce communication mechanisms where an agent $i$ shares latent states and actions at step $t$ through a message $m_i^t$ defined as follows:

$$m_i^t = c_\omega\big(z_i^{t-1}, a_i^{t-1}\big), \tag{8}$$

This communication is commonly restricted to neighbors, thereby reducing complexity from $\mathcal{O}(n^2)$ to $\mathcal{O}(n)$. With MAMBA, $c_\omega$ is a Transformer neural network.

### 3.2 Imitation Learning

*Imitation Learning* (IL) leverages expert demonstrations $\mathcal{D}_E = \{(s_i, a_i^{expert})\}_{i=1}^M$, $M$ being the number of samples available, to accelerate reinforcement learning. The simplest approach, *behavioral cloning* (BC) [15], minimizes the negative log-likelihood of expert actions:

$$\mathcal{L}_{\mathrm{BC}}(\pi_\phi) = \mathbb{E}_{(s,a^{expert})\sim\mathcal{D}_E}\big[-\log \pi_\phi(a^{expert} \mid s)\big]. \tag{9}$$

However, combining IL with world models is non-trivial. In world models, the actor operates in the latent space $z$, while demonstrations are expressed in the true state space $s$. A mapping through the learned encoder $e_\theta$ is required:

$$z = e_\theta(s), \tag{10}$$

but this mapping is optimized for compression and prediction, not alignment with expert trajectories. Although the explicit alignment of expert trajectories with latent spaces is uncommon, several recent works tackle this challenge. DITTO [5] introduces an intrinsic latent reward to align agent and expert behaviors, FLARE [54] aligns future latent representations during policy learning, *latent diffusion planning* [1] generates transformed demonstrations to better cover the latent space. These works highlight the open challenge of transferring expert knowledge into latent representations, which motivates our approach to bridge expert solvers with local latent policies.

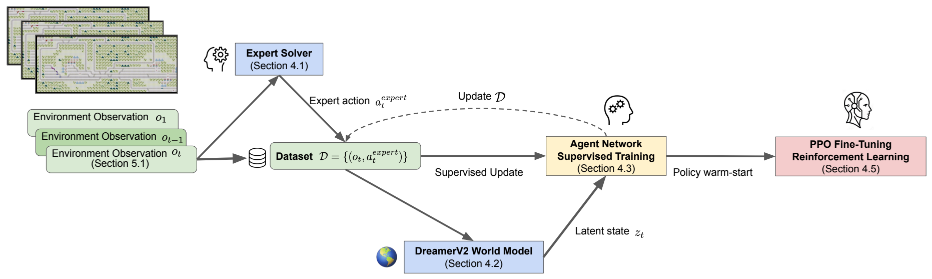
Expert Solver (Section 4.1)
Expert action $a_t^{expert}$
Update $\mathcal{D}$
Environment Observation $o_1$
Environment Observation $o_{t-1}$
Environment Observation $o_t$ (Section 5.1)
Dataset $\mathcal{D} = \{(o_t, a_t^{expert})\}$
Supervised Update
Agent Network Supervised Training (Section 4.3)
Policy warm-start
PPO Fine-Tuning Reinforcement Learning (Section 4.5)
Latent state $z_t$
DreamerV2 World Model (Section 4.2)

**Fig. 1.** Overview of our approach: (1) pretraining DreamerV2 world model, (2) collecting expert-labeled trajectories, (3) iterative supervised pretraining, and (4) fine-tuning with RL. Each main step is detailed in a specific subsection.

## 4 Our Imitation-Guided World Model

In this section, we describe our approach to warm-start both the world model and the agent in a sequential, iterative manner, using expert demonstrations from a Flatland solver to initialize the neural network parameters before reinforcement learning to fine-tune it. Our approach is illustrated in Figure 1.

### 4.1 Expert Demonstrator

A strong reference for large-scale railway rescheduling is the solver of Li et al. [21], the winner of the NeurIPS 2020 Flatland Challenge. It showed that well-engineered planning and optimization techniques can outperform learning-based methods in terms of scalability and robustness under stochastic disruptions.

The solver combines key components from multi-agent pathfinding: an initial feasible solution is built through *prioritized planning* [37], where agents are ordered and planned sequentially; each path is generated with *safe interval path planning* (SIPP) [33] to respect spatiotemporal constraints; and the solution is refined via large neighborhood search (LNS) [38], guided by simulated annealing heuristics [19] to balance exploration and exploitation.

A notable strength is its adaptive replanning: when a breakdown occurs, only the trains affected by the blocked segment are replanned, limiting the propagation of delays. Minimum communication policies [28] further prevent deadlocks in congested areas. The solver scales to 400 trains and achieves top performance in Flatland without using reinforcement learning, confirming the competitiveness of combinatorial optimization in this domain.

However, this approach assumes global knowledge of the full network and plans paths centrally an unrealistic requirement for real railway operations, where trains rely on local observations and limited communication. In contrast, our method leverages imitation learning and world models to enable decentralized decision-making under partial observability. Using this expert, we collect a dataset $\mathcal{D}_E$ of 800K samples, though our approach remains compatible with any alternative expert, including those reflecting real operational practices.

### 4.2 DreamerV2 World Model Pretraining

In addition to the expert's policy, we train a latent dynamics model following the DreamerV2 framework [10] as in Mamba [6] without any modification. DreamerV2 learns a compact latent representation $z_t$ of the environment from observations $o_t$ and actions $a_t$ where $z_t$ is composed of a deterministic recurrent state $h_t$ and a stochastic state $s_t$. This structured representation enables the model to predict future latent states, rewards, and discount factors. The model consists of 6 components. First, we have a recurrent state-space model (RSSM) composed of three models:

$$\begin{aligned} \text{Recurrent Model} \quad & h_t = f_\theta(h_{t-1}, s_{t-1}, a_{t-1}), && (11) \\ \text{Representation Model} \quad & s_t \sim q_\theta(s_t | h_t, o_t), && (12) \\ \text{Transition Predictor} \quad & \hat{s}_t \sim p_\theta(\hat{s}_t | h_t). && (13) \end{aligned}$$

Additionally, we have three more predictors:

$$\begin{aligned} \text{Observation Predictor} \quad & \hat{o}_t \sim p_\theta(\hat{o}_t | h_t, s_t), && (14) \\ \text{Reward Predictor} \quad & \hat{r}_t \sim p_\theta(\hat{r}_t | h_t, s_t), && (15) \\ \text{Discount Predictor} \quad & \hat{\gamma}_t \sim p_\theta(\hat{\gamma}_t | h_t, s_t). && (16) \end{aligned}$$

We denote by $q_\theta$ the neural network parameterizing the generative distributions of the real environment, and by $p_\theta$ the inference networks that approximate them to enable latent imagination. In our case, we reuse the Mamba architecture: our observation predictor, reward predictor and discount predictor are 2 fully-connected layers of 400 neurons each; The transition predictor is a single fully-connected layer of 400 neurons, followed by an attention encoder (3 layers, 8 heads), a gated recurrent unit (GRU) (1 layer, 600 neurons), and two fully-connected layers of 400 neurons; The representation model has the same architecture with an additional MLP (2 layers, 400 neurons per layer) at the end. Concerning the recurrent model (Equation 11), $h_t$ is computed using a GRU referred to as $f_\theta$. The GRU is chosen for its ability to efficiently capture temporal dependencies in sequential data, while being less prone to vanishing gradient issues compared to vanilla RNNs, and is computationally lighter than LSTMs. This is particularly relevant in our multi-agent railway environment, where long-term dependencies matter (e.g., train interactions over multiple time steps) but efficiency is critical due to the large number of agents.

The representation model $q_\theta$ (Equation 12) infers the stochastic latent state from the deterministic hidden state and the current observation. This stochastic component captures uncertainty in the environment and allows the model to represent multiple plausible futures, which is critical in partially observable multi-agent settings such as ours because trains can have information (i.e. their previous actions and states) only from nearby other trains.

The transition predictor $p_\theta$ (Equation 13) predicts the next stochastic latent state purely from the deterministic hidden state, enabling latent imagination without direct access to future observations. This facilitates planning and policy learning entirely in latent space. The observation predictor (Equation 14) reconstructs observations from the latent state, ensuring that the latent representation retains sufficient information about the environment for downstream tasks, such as training policies or critics. The reward predictor (Equation 15) predicts rewards in the latent space, providing the signal necessary for reinforcement learning. By predicting rewards from latent states rather than raw observations, the model enables sample-efficient policy updates. The discount predictor (Equation 16) estimates the expected discount factor for the next time step, allowing the model to handle stochastic episode terminations or partial observability in a principled manner.

The model is trained by minimizing the negative log-likelihood of observations, rewards, and discounts, along with a KL divergence term to regularize the latent distribution:

$$\begin{aligned}\mathcal{L}_{\text{world}}(\theta) = &-\mathbb{E}_{q_\theta(s_{1:t}|h_{1:t},o_{1:t})}[\log p_\theta(\hat{o}_t, \hat{r}_t, \hat{\gamma}_t | h_t, s_t)] \\ &+ \text{KL}\big[q_\theta(s_t|h_t, o_t) \,\big\|\, p_\theta(\hat{s}_t|h_t)\big]\end{aligned} \tag{17}$$

where $s_{1:t}, h_{1:t}, o_{1:t}$ are sequences of stochastic states, deterministic states and observations. This training ensures that the latent representation $z_t = (h_t, s_t)$ captures sufficient information for both planning and policy learning, enabling sample-efficient decision-making in downstream imitation learning and RL stages.

## 4.3 Supervised agent pretraining

Using the expert dataset $\mathcal{D}_E$ and the latent states $z_t$ from the world model, we train the agent network $\pi_\phi(a \mid z)$ in a supervised manner by minimizing the cross-entropy loss over discrete actions:

$$\mathcal{L}_{\text{IL}}(\phi) = -\frac{1}{|\mathcal{D}_E|} \sum_{\substack{a_t^{\text{expert}} \in \mathcal{D}_E \\ z_t \sim q_\theta, f_\theta}} \log \pi_\phi\big(a_t = a_t^{\text{expert}} \mid z_t\big) \tag{18}$$

This loss encourages the policy to mimic the expert in the latent space learned by the world model. Importantly, trajectories are collected by rolling out the agent's current policy $\pi_\phi$. At each timestep, we query the expert for the action $a_t^{\text{expert}}$ they would have taken, but we do not follow the expert's policy. Instead, the agent executes its own actions, while the expert provides labels used for supervision. This setup ensures that the training distribution matches the states actually encountered by the agent, mitigating covariate shift and making the learned policy more robust during deployment. This pre-training phase is carried out using the Adam algorithm [18] with a learning rate $lr = 5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a weight decay of 0.00001.

This supervised pretraining aligns the agent's initial policy with expert behaviors, while still grounding learning in the agent's own state-action distribution. As a result, the agent benefits from expert guidance without becoming overly reliant on demonstrations, providing a strong initialization for subsequent reinforcement learning fine-tuning. The full pseudocode of the supervised agent pretraining can be found in Algorithm 1. It simply consists of iteratively updating the world model and policy weights by backpropagating the loss functions using a batch $\mathcal{B}$ of $k$ samples drawn from the expert dataset (lines 3-4).

**Algorithm 1** `pretraining`(.) - Supervised Agent Pretraining

```
Require: World model $f_\theta, p_\theta, q_\theta$.
Require: Policy $\pi_\phi$.
Require: Dataset $\mathcal{D}$.
Require: Batch size $k$, number of epochs $E$.
1: for step 1 to $E$ do
2:     $\mathcal{B}$ := randomSampling($\mathcal{D}, k$)
3:     $\langle f_\theta, p_\theta, q_\theta \rangle$ := updateWorldModel($\mathcal{B}, \mathcal{L}_{\text{world}}, f_\theta, p_\theta, q_\theta$)
4:     $\pi_\phi$ := updateActor($\mathcal{B}, \mathcal{L}_{\text{IL}}, f_\theta, p_\theta, q_\theta$)
5: end for
6: return $f_\theta, p_\theta, q_\theta$, $\pi_\phi$
```

## 4.4 Fine-Tuning with Reinforcement Learning

After the pretraining phase, we continue to train the agent using reinforcement learning to adapt to situations not perfectly captured by the expert. We employ the *proximal policy optimization* (PPO) algorithm [36], an actor-critic method consisting of two main components: (1) the *actor* $\pi_\phi(a|z)$, which outputs a policy over actions given the latent state $z$ produced by the world model, and (2) the *critic* $V_\psi(z)$, which estimates the expected return (value) from the latent state.

Both neural networks are updated by minimizing the actor loss $\mathcal{L}_{\text{actor}}$ and the critic loss $\mathcal{L}_{\text{critic}}$ as defined in the standard PPO algorithm [36]. In our case, the overall update alternates between minimizing $\mathcal{L}_{\text{actor}}$ with respect to $\phi$ and $\mathcal{L}_{\text{critic}}$ with respect to $\psi$. This separation ensures that the actor is guided by the advantage signal without being biased by value prediction errors, while the critic learns an accurate baseline for stable advantage estimation. Combined with the world model, this actor-critic scheme enables the agent to make effective decisions under partial observability and stochastic disruptions in railway networks.

## 4.5 Pseudo-code of the Full Pipeline

The full training pipeline is described in the Algorithm 2. Starting from an empty training set, we first execute the pretraining phase. At each iteration, new trajectories are collected using the updated policy $\pi_\phi$ (line 4). Then, for each

**Algorithm 2** `fullTraining(.)` - Imitation-Guided World Model

```
Require: World model $f_\theta, p_\theta, q_\theta$.
Require: Actor $\pi_\phi$, critic $V_\psi$.
Require: IL steps $E_{\text{IL}}$, RL steps $E_{\text{RL}}$, batch size $k$.
 1: Initialize randomly $f_\theta, p_\theta, q_\theta, \pi_\phi, V_\psi$.
 2: $\mathcal{D} = \emptyset$
 3: for step 1 to $E_{\text{IL}}$ do
 4:     $\langle o, a, r \rangle :=$ envStep($\pi_\phi$)                      ▷ Following the current policy
 5:     $a^{\text{expert}} :=$ getExpertAction($o$)                                  ▷ Section 4.1
 6:     $\mathcal{D} := \mathcal{D} \cup \{o, a, r, a^{\text{expert}}\}$                               ▷ Updating the dataset
 7:     $\langle f_\theta, p_\theta, q_\theta, \pi_\phi \rangle :=$ pretraining($f_\theta, p_\theta, q_\theta, \pi_\phi, \mathcal{D}$)               ▷ Alg. 1
 8: end for
 9: for step 1 to $E_{\text{RL}}$ do                                            ▷ Section 4.4
10:     $\mathcal{B} :=$ randomSampling($\mathcal{D}, k$)
11:     $\langle f_\theta, p_\theta, q_\theta \rangle :=$ updateWorldModel($\mathcal{B}, \mathcal{L}_{\text{world}}, f_\theta, p_\theta, q_\theta$)
12:     $\pi_\phi :=$ updateActor($\mathcal{B}, \mathcal{L}_{\text{actor}}, f_\theta, p_\theta, q_\theta, \pi_\phi, V_\psi$)
13:     $V_\psi :=$ updateCritic($\mathcal{B}, \mathcal{L}_{\text{critic}}, f_\theta, p_\theta, q_\theta, \pi_\phi, V_\psi$)
14: end for
15: return $f_\theta, p_\theta, q_\theta, \pi_\phi$
```

observation, the expert is queried about the action they would have taken (line 5). The resulting action is added to the dataset (line 6). The world model and the actor are then retrained to minimize their respective losses on the expanded dataset (line 7). This iterative procedure ensures that the training distribution continually reflects the states actually encountered by the agent, thereby reducing covariate shift compared to one-shot behavioral cloning. As the policy improves, it explores more diverse regions of the state space, which, in turn, allows the expert to provide guidance in situations that were not present in the initial dataset. The world model also benefits from this process, since it is periodically retrained on the newly collected trajectories, improving its predictive accuracy in regions of the state-action space that become more relevant as the agent evolves.

The process is repeated for $E_{\text{IL}}$ steps. We set this number to be high enough to reach convergence, defined as the point where the supervised loss $\mathcal{L}_{\text{IL}}$ no longer decreases significantly or the policy performance on a held-out validation set saturates. In practice, this iterative pretraining acts as a form of data aggregation, similar in spirit to DAgger [34], but adapted to the latent world model setting. It provides a stronger initialization for reinforcement learning fine-tuning (lines 9-14) by combining expert knowledge with the agent's own evolving state distribution. We finally return the policy ($\pi_\phi$) and the world model ($f_\theta, p_\theta, q_\theta$).

## 5 Experiments

This section outlines the experimental protocol used to evaluate the efficiency and reliability of our approach and the obtained results.

### 5.1 Flatland environment

*Flatland* simulates a railway network on a $w \times h$ grid representing $n$ cities. Each unblocked cell contains a rail type that specifies the allowed movement directions. We consider $m$ trains $a_1, \ldots, a_m$, each with a starting cell, initial orientation, and destination (see Fig. 2). Time is discretized from 0 to $T_{\max}$, which depends on the environment size and the number of trains. The objective is to issue commands so that the maximum number of trains reach their destinations before $T_{\max}$, i.e., to maximize the success rate. Rewards are provided directly by the environment. At timestep 0, trains are outside the grid; each is inserted by issuing an entry command, after which they occupy exactly one cell per timestep until reaching their destination. At every step, a train chooses among five actions: *move forward*, *stop*, *turn left*, *turn right*, or *continue* (repeating the previous action). Two actions are in conflict if they cause a collision. Breakdowns interrupt a train at a random moment for a random duration. For each train, breakdown times follow a Poisson process with an unknown rate $\lambda$, and durations are drawn uniformly between 20 and 50 timesteps, becoming known only when the breakdown occurs.

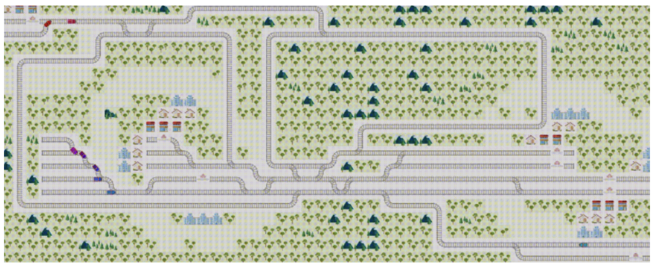

**Fig. 2.** Illustration of the Flatland Environment.

### 5.2 Protocol

We evaluate our approach against four baselines:

1. Random, a policy that selects actions uniformly at random for each train.
2. Greedy, a policy that chooses the action minimizing the immediate distance to the destination, ignoring other trains.
3. Expert, a policy that follows the expert solver described in Section 4.1.
4. Mamba [6], the state-of-the-art approach based on a learning algorithm.

All methods are evaluated on three levels of difficulty: 5, 10, and 15 trains, as we restrict ourselves to instance sizes commonly used in the learning-based literature. Learning-based models were trained on a single Tesla V100-PCIE-32GB GPU for a maximum of 220 hours.

All models were trained until convergence. For the training of our imitation-guided world model, the supervised learning phase was limited to 800K environment steps, corresponding to approximately 196 hours (almost 90% of the total training time) due to the repeated calls to the expert solver, while the reinforcement learning phase was carried out for 950K steps, taking around 24 hours. We note that, unlike the vanilla MAMBA, which trains a separate policy for each environment configuration (5, 10, and 15 trains with breakdown ratios of 1/100, 1/150, and 1/200 respectively), our method requires only a single training session on the intermediate 10-train environment. We refer to MAMBA(10) for version trained only on the 10-train environments and to MAMBA($\star$) for the version trained on the specific scenario. We evaluate the performances using two metrics: $\%r$, the proportion of trains that reach their, and $\%d$, the proportion of trains that become deadlocked. We note that these metrics do not necessarily sum to 100%, as some trains may remain on the grid without arriving or becoming deadlocked, a situation that is less critical than deadlock. All results are averaged over 100 environment instances.

### 5.3 Main Results

**Performances with a fixed breakdown ratio.** The results are summarized in Table 1. First, we observe that the expert achieves near-perfect performance across all scenarios, which is expected given its access to global state information and centralized optimization. Although available at the planning level and during the training phase, such global knowledge cannot be leveraged for real-time operations at the level of individual trains or restricted areas (e.g., a station). This is why the core of our methodology lies in mimicking the expert's behavior, without prompting it at inference time. Random and greedy policies perform poorly, with low arrival rates and high deadlock percentages. Focusing now on learning-based approaches, our sequential method competes with MAMBA in the 5-train environment, and significantly outperforms it in the 10- and 15-train scenarios, with improvements of 14.7% and 23.7% in arrival rates, respectively. These gains come at the cost of a slight increase in deadlocks (3% for 10 trains and 3.1% for 15 trains), which remains within acceptable bounds. We note that deadlock avoidance is not included in the reward function, and the agent is therefore not explicitly trained to minimize this metric.

**Performances when varying the breakdown ratio.** To assess generalization to stochastic disruptions, we also vary the train breakdown ratio between 1/10 (i.e., $\lambda$ from the Poisson law) and 0 for each difficulty level. For this experiment, our model is always the one trained on the 10-train environment with a breakdown ratio of 1/150, while MAMBA is trained on the original breakdown ratios associated with the instances (5, 10, and 15 trains with breakdown ratios of 1/100, 1/150, and 1/200 respectively). Results are reported in Table 2. Interestingly, our approach shows stable and consistent performance across configurations. While MAMBA performs best on the specific breakdown ratio it was

**Table 1.** Results of our method and the baselines (Up: non learning methods, Down: learning-based methods).

| *Policy* | *5 trains* | | *10 trains* | | *15 trains* | |
|---|---|---|---|---|---|---|
| | %r | %d | %r | %d | %r | %d |
| Random | 25.0 | 63.0 | 11.8 | 80.7 | 8.2 | 84.2 |
| Greedy | 18.4 | 55.0 | 10.7 | 79.1 | 8.5 | 81.3 |
| Expert | **100.0** | **0.0** | **98.0** | **1.8** | **98.1** | **1.5** |
| Mamba(10) | 48.4 | 0.4 | 66.1 | **0.2** | 30.9 | **1.6** |
| Mamba($\star$) | 97.2 | 2.0 | 66.1 | **0.2** | 47.6 | 4.6 |
| Our approach | **98.2** | **0.0** | **80.8** | 3.2 | **71.3** | 7.7 |

trained on, its performance deteriorates on unseen ratios. In contrast, the performance of our approach tends to increase smoothly as breakdowns decrease, demonstrating better adaptability in practical scenarios where disruptions vary seasonally or across lines.

**Main take-away from the results.** A key advantage of our method is its ability to transfer knowledge from the global solver to a decentralized local policy. Only a single training on the intermediary 10-train environment is sufficient to achieve good and consistent performance across all difficulty levels, whereas Mamba requires separate training for each configuration, as it fails to generalize without retraining. Overall, these results show that our imitation-guided world model approach not only improves performance compared to standard model-based MARL, but also generalizes better to new configurations and breakdown ratios, while requiring fewer training resources, as it needs only a single 220 hours training session, whereas Mamba requires three separate runs of 220 hours each (one per instance type), totaling 660 hours.

## 5.4 Ablation Study

To better understand the contribution of each component of our method, we perform an ablation study. All results (still averaged over 100 scenarios and with the model trained on the 10-train environment) are reported in Table 3 and are commented in the following paragraphs.

**Ablation 1: degrading the expert quality.** We investigate the importance of the expert used for imitation. To do so, we replace the solver expert with the random policy and the greedy policy, while keeping the rest of the methodology unchanged. As seen in Table 3, the sequential approach trained with a random or greedy expert still achieves decent performance, though it is lower than when using the solver. Interestingly, the random-expert model can still solve a substantial fraction of the simpler environments, suggesting that the imitation phase, even with weak guidance, can help the policy discover basic navigation

**Table 2.** Results of varying the breakdown ratio for 5, 10, and 15 trains.

*5 trains (original environment is 1/100)*

| *Policy* | *1/10* | | *1/50* | | *1/100* | | *1/500* | | *1/1000* | | *1/10000* | | *0* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d |
| Mamba(10) | 39.4 | **0.0** | 47.8 | **0.0** | 48.4 | 0.4 | 52.8 | **0.0** | 53.6 | **0.0** | 52.4 | **0.0** | 50.8 | **0.0** |
| Mamba(⋆) | **94.8** | **0.0** | **99.2** | 0.8 | 97.2 | 2.0 | 98.8 | 1.2 | 99.2 | 0.8 | 97.6 | 2.4 | **99.2** | 0.8 |
| Our approach | 50.6 | 0.4 | 96.6 | **0.0** | **98.2** | **0.0** | **99.0** | 0.4 | **100.0** | **0.0** | **100.0** | **0.0** | **99.2** | 0.4 |

*10 trains (original environment is 1/150)*

| *Policy* | *1/10* | | *1/50* | | *1/150* | | *1/500* | | *1/1000* | | *1/10000* | | *0* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d |
| Mamba(10) | 30.0 | **0.2** | 53.3 | **0.0** | 66.1 | **0.2** | 56.1 | **0.2** | 58.6 | **0.0** | 59.1 | **0.0** | 56.8 | **0.2** |
| Mamba(⋆) | 30.0 | **0.2** | 53.3 | **0.0** | 66.1 | **0.2** | 56.1 | **0.2** | 58.6 | **0.0** | 59.1 | **0.0** | 56.8 | **0.2** |
| Our approach | **42.3** | 1.4 | **68.2** | 2.4 | **80.8** | 3.2 | **83.7** | 2.8 | **78.9** | 3.4 | **84.7** | 2.6 | **82.7** | 2.4 |

*15 trains (original environment is 1/200)*

| *Policy* | *1/10* | | *1/50* | | *1/200* | | *1/500* | | *1/1000* | | *1/10000* | | *0* | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d | %r | %d |
| Mamba(10) | 19.0 | 2.9 | 30.1 | **0.7** | 30.9 | **1.6** | 35.7 | **0.7** | 34.5 | **1.3** | 35.3 | **1.3** | 36.0 | **2.3** |
| Mamba(⋆) | 30.9 | **2.7** | 39.3 | 6.2 | 47.6 | 4.6 | 48.8 | 5.0 | 49.5 | 6.3 | 48.3 | 6.0 | 47.3 | 7.4 |
| Our approach | **31.8** | 5.1 | **52.7** | 5.8 | **71.3** | 7.7 | **72.3** | 8.0 | **71.0** | 12.1 | **77.7** | 7.8 | **70.8** | 9.8 |

and coordination patterns. The greedy-expert model performs better than the random one on smaller environments but deteriorates on larger ones, indicating that partial heuristic knowledge can be beneficial but insufficient to capture the complex interdependencies present in high-density traffic.

**Ablation 2: disabling the world model.** Next, we evaluate the role of the world model by re-training the agent without it. Without a learned dynamics model, the agent is unable to capture temporal dependencies or plan ahead, resulting in poor results on the training configuration (10 agents) and a complete failure to generalize to smaller (5 agents) or larger (15 agents) scenarios. These results underline the crucial role of the world model in enabling generalization beyond the training distribution. This confirms that the world model is a critical component for learning in complex multi-agent environments such as Flatland. The result highlights that local observation alone, without the latent predictive model, does not provide sufficient information for coordination, especially when trains must anticipate interactions many steps in advance.

**Ablation 3: disabling RL finetuning.** We then assess the necessity of the reinforcement learning fine-tuning stage by training a policy purely with supervised imitation for the same budget as our full approach. Pure imitation alone yields very poor performance, particularly in larger environments with many agents. This demonstrates that expert-guided initialization is insufficient to handle the stochasticity of the environment, the variability of train breakdowns, and the emergent interactions between agents. It further emphasizes that reinforcement learning is essential for adapting the policy to situations not captured in the expert trajectories.

**Take-away from the ablation study.** Overall, the experiments confirm that each component (expert demonstrations, predictive world model, and reinforcement learning) plays a complementary role. The combination enables efficient policy learning and robust coordination under uncertainty for multi-agent routing problems in Flatland.

**Table 3.** Results of the ablation study.

| *Policy* | *5 trains* | | *10 trains* | | *15 trains* | |
|---|---|---|---|---|---|---|
| | %r | $\Delta$ | %r | $\Delta$ | %r | $\Delta$ |
| Our approach | **98.2** | **-** | **80.8** | - | **71.3** | - |
| With RANDOM | 75.4 | -22.8 | 76.6 | -3.4 | 58.1 | -13.2 |
| With GREEDY | 92.0 | -6.2 | 65.2 | -14.8 | 43.7 | -27.6 |
| No world model | 0.0 | -98.2 | 30.0 | -50.8 | 0.0 | -71.3 |
| Pure imitation | 2.8 | -95.4 | 0.4 | -80.4 | 0.0 | -71.3 |

## 6 Conclusion and Future Work

In this work, we proposed an imitation-guided world model to tackle a train rescheduling problem, using the Flatland environment, a simplified simulation of railway networks. First, we leverage imitation learning to mimic the behavior of an expert solver specifically designed for this task. This solver has access to a global view of the environment, which can be used during a training phase but is impractical for real-time operations. The second phase employs model-based reinforcement learning, following the DreamerV2 architecture, allowing each train to make decisions based solely on its local observations, reflecting realistic operational constraints. Our approach demonstrates superior performance compared to baseline methods, requires significantly fewer computational resources for training and exhibits robust generalization to previously unseen breakdown ratios. As future work, we plan to extend beyond the Flatland environment and address more realistic settings and constraints, such as strict arrival time windows, and priority rules reflecting passengers or cargo importance. Constrained reinforcement learning [35], which allows soft penalization of constraint non-adherence, appears to be a promising direction in this regard. For instance, it can be used to reduce the number of deadlocks as an ad-hoc objective.

**Acknowledgments.** This work was supported by Mitacs Grants IT29211 and IT35069. The authors gratefully acknowledge the discussions with SNCF researchers.

## References

1. Barcellona, L., Zadaianchuk, A., Allegro, D., Papa, S., Ghidoni, S., Gavves, E.: Dream to manipulate: Compositional world models empowering robot imitation learning with imagination. In: The Thirteenth International Conference on Learning Representations (2025), `https://openreview.net/forum?id=3RSLW9YSgk`
2. Bellman, R.: Dynamic programming. science **153**(3731), 34–37 (1966)
3. Cappart, Q., Schaus, P.: Rescheduling railway traffic on real time situations using time-interval variables. In: International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. pp. 312–327. Springer (2017)
4. Cobbe, K., Klimov, O., Hesse, C., Kim, T., Schulman, J.: Quantifying generalization in reinforcement learning. In: International conference on machine learning. pp. 1282–1289. PMLR (2019)
5. DeMoss, B., Duckworth, P., Hawes, N., Posner, I.: Ditto: Offline imitation learning with world models. arXiv preprint arXiv:2302.03086 (2023)
6. Egorov, V., Shpilman, A.: Scalable multi-agent model-based reinforcement learning. In: Proceedings of the 21st International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2022). pp. 381–389. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) (2022), `https://arxiv.org/abs/2205.15023`, online, May 9–13, 2022
7. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018). pp. 2974–2982 (2018), `https://arxiv.org/abs/1705.08926`
8. Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., Davidson, J.: Learning latent dynamics for planning from pixels. In: Proceedings of the 36th International Conference on Machine Learning (ICML). pp. 2555–2565 (2019), `https://arxiv.org/abs/1811.04551`
9. Hafner, D., Lillicrap, T.P., Ba, J., Norouzi, M.: Dream to control: Learning behaviors by latent imagination. In: International Conference on Learning Representations (ICLR) (2020), `https://arxiv.org/abs/1912.01603`, spotlight
10. Hafner, D., Lillicrap, T.P., Norouzi, M., Ba, J.: Mastering atari with discrete world models. In: International Conference on Learning Representations (ICLR) (2021), `https://openreview.net/forum?id=5r6uxv1h5V`, accepted as spotlight
11. Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems **33**(6), 750–797 (2019)
12. Ho, F., Goncalves, A., Salta, A., Cavazza, M., Geraldes, R., Prendinger, H.: Multi-agent path finding for uav traffic management: Robotics track pp. 131–139 (2019)
13. Ho, J., Ermon, S.: Generative adversarial imitation learning. Advances in neural information processing systems **29** (2016)
14. Huh, D., Mohapatra, P.: Multi-agent reinforcement learning: A comprehensive survey. arXiv preprint arXiv:2312.10256 (2023)
15. Hussein, A., Gaber, M.M., Elyan, E., Jayne, C.: Imitation learning: A survey of learning methods. ACM Computing Surveys (CSUR) **50**(2), 1–35 (2017)

16. Hönig, W., Preiss, J.A., Kumar, T.K.S., Sukhatme, G.S., Ayanian, N.: Trajectory planning for quadrotor swarms. IEEE Transactions on Robotics **34**(4), 856–869 (2018). `https://doi.org/10.1109/TRO.2018.2853613`
17. Jaziri, A., Künzel, E., Ramesh, V.: Mitigating the stability-plasticity dilemma in adaptive train scheduling with curriculum-driven continual dqn expansion. CoRR **abs/2408.09838** (2024), `https://doi.org/10.48550/arXiv.2408.09838`
18. Kinga, D., Adam, J.B., et al.: A method for stochastic optimization. In: International conference on learning representations (ICLR). vol. 5. California; (2015)
19. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). `https://doi.org/10.1126/science.220.4598.671`, `https://doi.org/10.1126/science.220.4598.671`
20. Konda, V., Tsitsiklis, J.: Actor-critic algorithms. Advances in neural information processing systems **12** (1999)
21. Li, J., Chen, Z., Zheng, Y., Chan, S.H., Harabor, D., Stuckey, P.J., Ma, H., Koenig, S.: Scalable rail planning and replanning: Winning the 2020 flatland challenge. Proceedings of the International Conference on Automated Planning and Scheduling **31**(1), 477–485 (May 2021). `https://doi.org/10.1609/icaps.v31i1.15994`, `https://ojs.aaai.org/index.php/ICAPS/article/view/15994`
22. Li, J., Tinka, A., Kiesel, S., Durham, J.W., Kumar, T.S., Koenig, S.: Lifelong multi-agent path finding in large-scale warehouses pp. 11272–11281 (2021)
23. Liao, J., Zhang, F., Zhang, S., Gong, C.: A real-time train timetable rescheduling method based on deep learning for metro systems energy optimization under random disturbances. Journal of Advanced Transportation **2020**(1), 8882554 (2020). `https://doi.org/https://doi.org/10.1155/2020/8882554`, `https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/8882554`
24. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
25. Liu, S., Zhang, Y., Tang, K., Yao, X.: How good is neural combinatorial optimization? a systematic evaluation on the traveling salesman problem. IEEE Computational Intelligence Magazine **18**(3), 14–28 (2023)
26. Liu, X., da Silva, C.F.O., Dabiri, A., Wang, Y., De Schutter, B.: Learning-based model predictive control for passenger-oriented train rescheduling with flexible train composition. arXiv preprint arXiv:2502.15544 (2025)
27. Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems **30** (2017)
28. Ma, H., Li, J., Kumar, T.S., Koenig, S.: Lifelong multi-agent path finding for online pickup and delivery tasks pp. 837–845 (2017)
29. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
30. Mohanty, S., Nygren, E., Laurent, F., Schneider, M., Scheller, C., Bhattacharya, N., Watson, J., Egli, A., Eichenberger, C., Baumberger, C., et al.: Flatland-rl: Multi-agent reinforcement learning on trains (2020)
31. Oh, J., Guo, Y., Singh, S., Lee, H.: Self-imitation learning. In: Proceedings of the 35th International Conference on Machine Learning (ICML 2018). pp. 3878–3887. PMLR (2018), `https://proceedings.mlr.press/v80/oh18b.html`
32. Peng, X.B., Kumar, A., Zhang, G., Levine, S.: Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. arXiv preprint arXiv:1910.00177 (2019)

33. Phillips, M., Likhachev, M.: SIPP: Safe interval path planning for dynamic environments. In: 2011 IEEE international conference on robotics and automation. pp. 5628–5635. IEEE (2011)
34. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
35. Roy, J., Girgis, R., Romoff, J., Bacon, P.L., Pal, C.J.: Direct behavior specification via constrained reinforcement learning. In: International Conference on Machine Learning. pp. 18828–18843. PMLR (2022)
36. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017), `https://arxiv.org/abs/1707.06347`
37. Sharon, G., Stern, R., Felner, A., Sturtevant, N.: Conflict-based search for optimal multi-agent path finding. In: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence. p. 563–569. AAAI'12, AAAI Press (2012)
38. Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming (CP'98). pp. 417–431. Springer (1998). `https://doi.org/10.1007/BFb0054171`, `https://doi.org/10.1007/BFb0054171`
39. Shi, Z., Liu, M., Zhang, S., Zheng, R., Dong, S., Wei, P.: Gawm: Global-aware world model for multi-agent reinforcement learning. arXiv preprint arXiv:2501.10116 (2025), `https://arxiv.org/abs/2501.10116`
40. Song, J., Ren, H., Sadigh, D., Ermon, S.: Multi-agent generative adversarial imitation learning. Advances in neural information processing systems **31** (2018)
41. Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., et al.: Multi-agent pathfinding: Definitions, variants, and benchmarks. In: Proceedings of the International Symposium on Combinatorial Search. vol. 10, pp. 151–158 (2019)
42. Surynek, P.: Problem compilation for multi-agent path finding: a survey. In: Raedt, L.D. (ed.) Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22. pp. 5615–5622. International Joint Conferences on Artificial Intelligence Organization (7 2022). `https://doi.org/10.24963/ijcai.2022/783`, `https://doi.org/10.24963/ijcai.2022/783`, survey Track
43. Sutton, R.S., Barto, A.G., et al.: Reinforcement learning: An introduction, vol. 1. MIT press Cambridge (1998)
44. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems **12** (1999)
45. Talvitie, E.: Self-correcting models for model-based reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)
46. Tang, J., Swamy, G., Fang, F., Wu, S.: Multi-agent imitation learning: Value is easy, regret is hard. In: The Thirty-eighth Annual Conference on Neural Information Processing Systems (2024)
47. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 5998–6008 (2017). `https://doi.org/10.5555/3295222.3295349`, `https://arxiv.org/abs/1706.03762`
48. Venugopal, A., Milani, S., Fang, F., Ravindran, B.: Mabl: Bi-level latent-variable world model for sample-efficient multi-agent reinforcement learning. In: Proceedings of the 23rd International Conference on Autonomous Agents and MultiAgent

Systems (AAMAS 2024). pp. 1865–1873. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) (2024), `https://www.ifaamas.org/Proceedings/aamas2024/pdfs/p1865.pdf`

49. Wang, H., Yu, L., Cao, Z., Ermon, S.: Multi-agent imitation learning with copulas. In: Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD). pp. 139–156. Springer (2021), `https://arxiv.org/abs/2107.04750`
50. Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**(3), 279–292 (1992)
51. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3), 229–256 (1992)
52. Wu, Z., Yu, C., Chen, C., Hao, J., Zhuo, H.H.: Models as agents: Optimizing multi-step predictions of interactive local models in model-based multi-agent reinforcement learning. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023). pp. 26241–26249. AAAI Press (2023), `https://ojs.aaai.org/index.php/AAAI/article/view/26241`
53. Zhang, Y., Deekshith, U., Wang, J., Boedecker, J.: Improving the efficiency and efficacy of multi-agent reinforcement learning on complex railway networks with a local-critic approach. Proceedings of the International Conference on Automated Planning and Scheduling **34**(1), 698–706 (May 2024). `https://doi.org/10.1609/icaps.v34i1.31533`, `https://ojs.aaai.org/index.php/ICAPS/article/view/31533`
54. Zheng, R., Wang, J., Reed, S., Bjorck, J., Fang, Y., Hu, F., Jang, J., Kundalia, K., Lin, Z., Magne, L., Narayan, A., Tan, Y.L., Wang, G., Wang, Q., Xiang, J., Xu, Y., Ye, S., Kautz, J., Huang, F., Zhu, Y., Fan, L.: FLARE: Robot learning with implicit world modeling. In: Structured World Models for Robotic Manipulation (2025), `https://openreview.net/forum?id=rFiLBM4YCh`
55. Zhu, Y., Wang, H., Goverde, R.M.: Reinforcement learning in railway timetable rescheduling. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). p. 1–6. IEEE Press (2020). `https://doi.org/10.1109/ITSC45102.2020.9294188`, `https://doi.org/10.1109/ITSC45102.2020.9294188`