Distributed Fleet Management in Noisy Environments via Model-Predictive Control

Simon Bøgh[1], **Peter Gjøl Jensen**[2], **Martin Kristjansen**[2], **Kim Guldstrand Larsen**[2], **Ulrik Nyman**[2]

[1] Aalborg University, Department of Materials & Production, Fibigerstræde 16, 9220 Aalborg Øst, Denmark
[2] Aalborg University, Department of Computer Science, Selma Lagerløfs Vej 300, 9220 Aalborg Øst, Denmark
sb@mp.aau.dk, {pgj,mk,kgl,ulrik}@cs.aau.dk

Abstract

We consider dynamic route planning for a fleet of Autonomous Mobile Robots (AMRs) doing fetch and carry tasks on a shared factory floor. In this paper, we propose Stochastic Work Graphs (SWG) as a formalism for capturing the semantics of such distributed and uncertain planning problems. We encode SWGs in the form of a Euclidean Markov Decision Process (EMDP) in the tool UPPAAL STRATEGO, which employs Q-Learning to synthesize near-optimal plans. Furthermore, we deploy the tool in an online and distributed fashion to facilitate scalable, rapid replanning. While executing their current plan, each AMR generates a new plan incorporating updated information about the other AMRs positions and plans. We propose a two-layer Model Predictive Controller-structure (waypoint and station planning), each individually solved by the Q-learning-based solver. We demonstrate our approach using ARGoS3 large-scale robot simulation, where we simulate the AMR movement and observe an up to 27.5% improvement in makespan over a greedy approach to planning. To do so, we have implemented the full software stack, translating observations into SWGs and solving those with our proposed method. In addition, we construct a benchmark platform for comparing planning techniques on a reasonably realistic physical simulation and provide this under the MIT open-source license.

1 Introduction

In modern industrial production, robots are increasingly working in close collaboration with humans in shared workspaces. In recent years, agents such as *automated guided vehicles* (AGV) and *autonomous mobile robots* (AMR), among other approaches, have enabled increased flexibility in manufacturing environments. AMR and AGV systems have thus become paramount to modern production paradigms where increased flexibility in logistics applications are enabled through such technologies (Fazlollahtabar and Saidi-Mehrabad 2015). Nonetheless, fleet management is still a significant challenge to solve and is highly dependent on the particular problem. For example, the fleet manager might need to allocate tasks, plan around resource availability, malfunctioning agents, and avoid blocking each other on paths or at stations.

In Multi-Agent Pickup and Delivery (Salzman and Stern 2021), a set of agents must complete tasks where they must pick up an item to be placed at certain delivery stations. Such methods can be done effectively and quickly if the agents only have one job each. Even if there is a stream of tasks, it can be solved in an online fashion (Ma et al. 2017), but the agent only needs to pick up items from a single location before delivering. We consider agents that must visit multiple stations before delivering the collected items.

In this paper, we consider complex planning of pickup and delivery when the environment contains stochastic timing information stemming from, e.g., noise in the location system, other sensors, or through collaboration with humans. We define in Section 3 the Stochastic Work Graph (SWG), its associated global scheduling problem, and the semantics as defined on a Euclidean Markov Decision Process (EMDP). Later in Sections 4 and 5, we divide the global (and intractable) problem into a two-level planning methodology, where we, on an abstract level, plan the sequence in which to visit stations, and afterward the exact paths to travel in the near future. The planning architecture is a dual Model-Predictive Control (MPC) approach, with each MPC problem solved by online Q-learning, using the tool UPPAAL STRATEGO (David et al. 2015). The Q-learning-based solver synthesizes near-optimal plans for visiting stations and waypoints. We consider a complex planning scenario in Section 7 for concept evaluation that entails 10, 15, and 20 AMRs doing pickup and delivery tasks. Our simulation architecture is based on the ARGoS3 framework (Pinciroli et al. 2012) which simulates the physics of the AMRs. We demonstrate that our approach can learn policies that outperform a greedy approach by up to 27.5% on average. A secondary contribution of the paper is the developed simulation platform. It is released as open-source under the MIT license and can be used to compare different planning approaches against each other.

Motivational Case

As a motivational case, we consider a setup inspired by a world-leading industrial company with many custom orders apart from their primary production. Humans currently hand-pick a series of components to fulfill a single order with no restrictions on the collection sequence. They are self-managed and rely on intuition to avoid congestion at shelves with items in high demand. When all the items have been picked up, they must be delivered to a delivery station.

2 Related Work

Optimal fleet management, and herein task planning and replanning, is a fundamental challenge in multi-AMR systems. In recent years, artificial intelligence methodologies have become prevalent in robotics research. We propose a combined learning and Model-Predictive Control (MPC) approach to the challenge.

The problem of planning multiple agents' paths through a system has been investigated as the Multi-Agent Path Finding (MAPF) problem (Salzman and Stern 2021). In MAPF, a set of agents must travel from a source to a destination where one-shot algorithms can compute such routes. Online (or lifelong) MAPF computes for a time window as the agents execute a stream of tasks (Ma et al. 2017). Extensions include Target Assignment and Path Finding (TAPF), in which the authors investigate the optimal assignment of tasks, and Multi-Agent Pickup Delivery (MAPD) (Li et al. 2020), where a task includes a location where to deliver an item. Others have attempted to combine reinforcement learning with MAPF approaches, such as (Ling, Gupta, and Kumar 2020) who divide the graph into zones and include aspects of uncertainty. Common for them all is that either each agent needs to visit just one location, or the sequence is fixed. In our research, we investigate how a task consisting of multiple locations followed by a delivery station can be visited, which is inspired by Fetch and Carry operations (Bøgh et al. 2014; Staal et al. 2020). In such applications, the agent can visit multiple locations before delivering. MAPF problems are in general formulated using discrete, unit time for movement and work, but (Hönig et al. 2016, 2019) investigate time-varying and simulate with higher-order dynamics. First, they defined MAPF-POST: a process for post-processing a calculated MAPF plan. The plan is updated to include kinematic constraints and define dependencies between the different steps in the plan, but the constraints are single intervals, such as the minimum and maximum duration of traveling along an edge. The benefit of this new plan is that it allows for better mapping to the execution of the robots. Later, they present the idea of post-processing in an online setting, where they also define work as δ time units and not just a single unit, but their implementation is not made available.

Recent research has proposed a Deep Q-Network-based approach to dispatch/navigation plans to serve randomly and constantly incoming orders and reduce time consumption for AGVs (Liu et al. 2021). Ridesharing platforms share many of the same challenges to fleet management in industrial AMR scenarios, e.g., how waiting time is reduced through proactive dispatching strategies. Reinforcement Learning has been applied to optimize dispatching strategies specifically, hence learning policies under uncertainty in the environment (Oda and Joe-Wong 2018). In this paper, we consider a stream of incoming orders, each comprising a set of items to collect from a factory floor. We assume that a simple global algorithm solves the allocation of AMRs to orders. The aim is to reduce the total completion time of the order queue. We differ from (Liu et al. 2021; Oda and Joe-Wong 2018) both in the method applied and by considering plans where multiple points have to be visited, resulting in more choices for each individual AMR. In particular, we leverage recent advances in near-optimal synthesis for Priced Timed Markov Decision Processes using UPPAAL STRATEGO (Jaeger et al. 2019). Our method automatically incorporates rapid replanning, something we believe will lead to faster response time to rare and unpredictable events, situations that require special care when applying Neural Network-based approaches (De Bruin et al. 2015; Schiøler et al. 2018). We apply the tool UPPAAL STRATEGO in the setting of MPC - a combination which has recently seen encouraging performance in various domains (Larsen et al. 2016; Eriksen et al. 2019; Goorden et al. 2021).

As the fleet of agents grows into large-scale systems, the fleet management problem becomes increasingly challenging. Traffic congestion and transportation efficiency become essential factors, which have been investigated through multi-agent deep reinforcement learning methods (Lin et al. 2018). To combat scalability issues, we enable each AMR to optimize its plan while taking the current plans of the other AMRs into account. This allows the individual AMRs to adapt locally to (temporal) congestion. An alternative approach is taken by (Gu et al. 2020), who rely on a centralized computation of navigation plans, which leads to scalability issues.

3 Problem Description

In our setting, a group of AMRs is expected to complete a (multi-)set of tasks, where each task consists of a set of pick-up stations to visit prior to delivering the collected items at a delivery station. The goal of the AMRs is to complete the collection of tasks as fast as possible in a given environment and under the assumption that an AMR can work on at most one task at any given time and that tasks are non-preemptable.

In order to capture the underlying properties of the environment the AMRs operate in, we abstract the real-world into a *Stochastic Work Graph.*

Definition 1 (Stochastic Work Graph) *A stochastic work-graph $G = (V, E, \Delta)$ is an undirected graph where:*

- *V is a set of vertices,*
- *$E \subseteq V \times V$ is a set of edges,*
- *$\Delta : V \cup E \to (\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0})$ denotes a stochastic distribution over the work time at vertices or travel time for edges s.t. $1 = \int_0^\infty \Delta(l)(x)dx$ for any $l \in V \cup E$.*

Notice here that the distributions capture the uncertainty in the environment as such to capture, e.g., human operators of a work-station or expected interference from the environment (or other AMRs). The maps that we operate on are not required to be well-formed infrastructures as defined in (Čáp et al. 2015) implying that AMRs can block routes while performing work.

Given an SWG, we can capture the entirety of the problem as a *Scheduling Problem for SWG*, formally defined as follows.

Definition 2 (Scheduling Problem for SWG) *An SPSWG $S = (R, I, \mathcal{T}, G)$ with $G = (V, E, \Delta)$ s.t.*

- *R is a set of names of AMRs,*
- *$I : R \to V$ is the initial AMR placement,*
- *$\mathcal{T} \subseteq \wp(V)$ is the (multi-)set of tasks, each being a set of vertices to be visited, and*
- *G is an SWG.*

The goal is to have the AMRs complete the set of tasks with the shortest makespan possible.

We will, for simplicity henceforth, assume w.log. that $\mathcal{T}$ is a set. Note that the tasks consist of a *set* of vertices an AMR must visit prior to completing its task, i.e., the sequence of the vertices is not given a priori. We can now define the abstract state of an AMR as follows.

Definition 3 (State of an AMR) *Let $\Omega = (V \cup E) \times \mathbb{B} \times \mathbb{R}_{\geq 0} \times \wp(V)$ be the set of all states, then the $(\ell, b, x, t) \in \Omega$ is the state of an AMR s.t.*

- *$\ell \in (V \cup E)$ is the location,*
- *$b \in \mathbb{B} = \{tt, ff\}$ indicates activity (such as working),*
- *$x \in \mathbb{R}_{\geq 0}$ is the progress of the current operation (movement along an edge, or operation at a station), and*
- *$t \in \wp(V)$ is the remaining set of locations to visit.*

As a shorthand, we let $\Lambda(r) \in \Omega$ denote the state of the AMR r in the global AMR state vector Λ. We further write $\Lambda[(\ell, b, c, \tau)/r]$ to update the state of $r \in R$.

Given the states Λ of all AMRs and the SWG G, we need an encoding that specifies the behavior and progression of task execution. The semantics of such a system can be aptly given as a Euclidean Markov Decision Process (EMDP) (Jaeger et al. 2019, 2020) which we shall briefly recall:

Definition 4 (Euclidean Markov Decision Process) *An Euclidean Markov Decision Process (EMDP) is defined as a tuple $\mathcal{M} = (S, Act, s_{init}, T, C, \mathcal{G})$, where*

- *$S \subseteq \mathbb{R}^K$ is a K-bounded and closed subset of an euclidean space,*
- *Act is the set of actions,*
- *s_{init} is the initial state,*
- *$T : S \times Act \to (S \to \mathbb{R}_{\geq 0})$ is the transition function that yields a density function $T(s, a)$ over S for each state s and action a,*
- *$C : S \times Act \times S \to \mathbb{R}_{\geq 0}$ is the cost functions over state-action-state triples, and*
- *$\mathcal{G} \subseteq S$ is the set of goal states.*

A benefit of EMDP is the generation and execution of strategies under such models, but we can also evaluate (or estimate) the cost of strategies. Given an EMDP and a strategy $\sigma : S \to (Act \to [0, 1])$, we can define the expected cost of such a strategy as follows.

Definition 5 (Expected cost of a strategy) *Let $\mathcal{G}$ be a set of goal-states and let σ be a strategy. The expected cost of reaching $\mathcal{G}$ starting in a state s – $\mathbb{E}_{\sigma}^{\mathcal{M}}(C, s)$ – is the solution to the following system of equations*[1]*:*

$$\mathbb{E}_{\sigma}^{\mathcal{M}}(\mathcal{G}, s) = \sum_{a \in Act} \sigma(s)(a) \cdot \int_{t \in S} T(s, a)(t) \cdot \big(C(s, a, t) + \mathbb{E}_{\sigma}^{\mathcal{M}}(\mathcal{G}, t)\big) dt$$

when $s \notin \mathcal{G}$ and $\mathbb{E}_{\sigma}^{\mathcal{M}}(\mathcal{G}, s) = 0$ when $s \in \mathcal{G}$.

In our case, we want to reduce the makespan of completing a task, such that a strategy's cost will be its expected time to complete the tasks. This leads to the optimization problem for EMDPs, which asks to find a strategy σ s.t. $\mathbb{E}_{\sigma}^{\mathcal{M}}(C, s_{init})$ is minimized, which will be the makespan in the case of SPSWGs. We now encode the semantics of our SPSWG as an EMDP:

Definition 6 (SPSWG as EMDP) *Given a SPSWG $S = (R, I, \mathcal{T}, G)$ with $G = (V, E, \Delta)$, the induced EMDP $\mathcal{M} = (S, Act, s_{init}, T, C, \mathcal{G})$, is given by*

- *$S = \Omega^R \times \mathcal{T}$ is the global state consisting of the mapping of AMR names to AMR states and the remaining tasks,*
- *Act is the set of actions,*
- *$s_0 = (\Lambda, \mathcal{T})$ s.t. $\Lambda(r) = (I(r), ff, 0, \emptyset)$ for all $r \in R$,*
- *$T : S \times Act \rightharpoonup (S \to \mathbb{R}_{\geq 0})$ is a partial transition function for the next state density function,*
- *$C : S \times Act \times S \to \mathbb{R}_{\geq 0}$ is the cost function, and*
- *$\mathcal{G} = \{(\Lambda, \emptyset) \in S \mid$ for all $r \in R$ we have $(\ell, b, x, \emptyset) = \Lambda(r)\}$.*

Thus, in goal states $\mathcal{G}$, there are no more tasks left in the task set $\mathcal{T}$, and all AMRs have completed their assigned tasks. We further define the set of actions Act as:

$$\begin{aligned} Act = &\{assign_{r,\tau} \mid r \in R, \tau \in \mathcal{T}\} \cup \\ &\{move_{r,e} \mid r \in R, e \in E\} \cup \\ &\{work_{r,v} \mid r \in R, v \in V\} \cup \\ &\{\delta\} \end{aligned}$$

where the four actions are assignment of a task to an AMR, an AMR moving on an edge or working at a vertex, and an abstract delay δ.

Now we can define the cost and transition functions completing the EMDP semantics of Definition 6. We denote a transition as follows.

$$(\Lambda, \mathcal{T}) \xrightarrow{\alpha}_{d,D} (\Lambda', \mathcal{T}')$$

Here, Λ is the state of the AMRs, $\mathcal{T}$ is the task set, $\alpha \in Act$ is an action, d is the concrete amount of time elapsed, which also denotes the cost, and D is the density of the transition. As the delay is the cost, assigning tasks, starting work, or initiating movement does not carry any direct cost and is seen as instantaneous. Also, their effects are deterministic, reflected by density 1.

Let us first define the transition-relation for the $assign$, $work$, and $move$ operations.

$$\begin{aligned} (\Lambda, \mathcal{T}) \xrightarrow{assign_{r,\tau}}_{0,1} (\Lambda', \mathcal{T}') \text{ if } & \Lambda(r) = (v, ff, c, \emptyset) \text{ and} \\ & \tau \in \mathcal{T} \text{ and} \\ & \mathcal{T}' = \mathcal{T} \setminus \{\tau\} \text{ and} \\ & \Lambda' = \Lambda[(v, ff, c, \tau)/r] \end{aligned}$$

[1] We shall assume that the equation system has a solution for the considered MDP and a goal set under any strategy.

$$\begin{aligned}(\Lambda, \mathcal{T}) \xrightarrow{work_{r,v}}_{0,1} (\Lambda', \mathcal{T}) \text{ if } &\Lambda(r) = (v, \mathit{ff}, c, \tau) \text{ and}\\ &v \in \tau \text{ and}\\ &\forall r' \in R \text{ it holds that}\\ &\ (v, \mathit{tt}, c', \tau') \neq \Lambda(r') \text{ and}\\ &\Lambda' = \Lambda[(v, \mathit{tt}, 0, \tau)/r]\end{aligned}$$

$$\begin{aligned}(\Lambda, \mathcal{T}) \xrightarrow{move_{r,e}}_{0,1} (\Lambda', \mathcal{T}) \text{ if } &\Lambda(r) = (v, \mathit{ff}, c, \tau) \text{ and}\\ &e = (v, v') \text{ and}\\ &\Lambda' = \Lambda[(e, \mathit{tt}, 0, \tau)/r]\end{aligned}$$

For clarification, we make the following notes: for $assign$, an inactive AMR r having completed its (previous) task may be assigned a new task; for $work$, an inactive AMR r may start working at its current location vertex and will have its clock reset. Note, no other AMR r' must be active and occupying v when $work$ is initiated at v. Finally, an inactive AMR r may start to $move$ along any edge from its current location vertex.

Before defining the transition for the delay action δ, we define two short-hand notations for conditional density and probability. Let $d, \epsilon \in \mathbb{R}_{\geq 0}$ and let $D : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ be a density. Then $D(d \mid \epsilon)$ denotes the conditional density of delaying additionally d assuming a delay of at least ϵ, defined as follows.

$$D(d \mid \epsilon) = \frac{D(\epsilon + d)}{\int_{t=\epsilon}^{\infty} D(t)\, dt} \tag{1}$$

This allows us to define the $D(\geq d \mid \epsilon)$ shorthand, denoting the probability of delaying additionally d assuming a delay of at least ϵ:

$$D(\geq d \mid \epsilon) = \int_{t=d}^{\infty} D(t \mid \epsilon)\, dt \tag{2}$$

Before defining the delay (δ) transition, we first define the set of active AMRs in a given state Λ.

$$Active_\Lambda = \{r \in R \mid \Lambda(r) = (\ell, \mathit{tt}, c, \tau)\}$$

In short, $Active_\Lambda$ denotes the set of AMRs engaged in either working at a vertex or moving along an edge of the work graph. Given Λ, an active AMR $r \in Active_\Lambda$ with $(\ell, \mathit{tt}, v, c) = \Lambda(r)$, we can now define the density that r will be the first active AMR to complete its activity and will do so after a delay of d:

$$D_{r,d,\Lambda} = \Delta(w)(d \mid c) \cdot \prod_{\substack{r' \in Active_\Lambda \text{ and} \\ r \neq r',\, (\ell', b', c', \ell') = \Lambda(r')}} \Delta(w')(\geq d \mid c')$$

Note, that $\Delta(w)(d \mid c)$ is the density of r completing after additionally d time-units given that c has already elapsed, and that the large product $\prod \ldots$ is the probability that all other active AMRs r' will *not* complete their activity before additionally d time units.

Now, finally, we may define the delay transition δ completing active AMRs either working at vertices $v \in V$ or moving along edges $(v, v') \in E$ of the underlying SWG:

$(\Lambda, \mathcal{T}) \xrightarrow{\delta}_{d, D_{r,d,\Lambda}} (\Lambda', \mathcal{T})$ where

a) if $(v, \mathit{tt}, c, \tau) = \Lambda(r)$ s.t. $v \in V$ then

$$\Lambda'(r') = \begin{cases} \Lambda(r') & \text{if } r' \notin Active_\Lambda \\ (v, \mathit{ff}, 0, \tau \setminus \{v\}) & \text{if } r' = r \\ (w', \mathit{tt}, c' + d, \tau') \text{ with} & \\ \quad (w', \mathit{tt}, c', \tau') = \Lambda(r') & \text{otherwise} \end{cases}$$

b) if $((v, v'), \mathit{tt}, c, t) = \Lambda(r)$ s.t. $(v, v') \in E$ then

$$\Lambda'(r') = \begin{cases} \Lambda(r') & \text{if } r' \notin Active_\Lambda \\ (v', \mathit{ff}, 0, \tau) & \text{if } r' = r \\ (w', \mathit{tt}, c' + d, \tau') \text{ with} & \\ \quad (w', \mathit{tt}, c', \tau') = \Lambda(r') & \text{otherwise} \end{cases}$$

Again, the intuition behind the delay transition is that the AMR r is the first of all active AMRs that complete current work (case a)) or travel (case b)).

Now we have completed the definition of the EMDP that models our scheduling problem, where tasks need to be assigned to AMRs that subsequently must visit the set of locations of the tasks. However, the definition of the semantics of SPSGW comes with openness in the choice as to which order the vertices of a task should be visited. In particular, no restrictions are given as to which is the last station. To better match our motivational case, we enforce that AMRs visit a delivery station after having visited all other stations in its task. To enforce a pick-up-and-delivery semantics, we introduce a set of delivery vertices as a subset $\theta \subseteq V$ of the vertices of the underlying SWG.

Definition 7 (Pick-up and Delivery Problem (PDP)) *Let SPSWG $S = (R, I, \mathcal{T}, G)$ with $G = (V, E, \Delta)$. Now given a set of delivery stations $\theta \subseteq V$, the induced EMDP $\mathcal{M} = (S, Act, s_{init}, T, C, \mathcal{G})$ from Definition 6 is modified as follows:*

- *for each $t \in \mathcal{T}$, $|t \cap \theta| \geq 1$, i.e. each task has at least one delivery station, and*
- *if $(\Lambda, \mathcal{T}) \xrightarrow{work_{r,v}}_{0,1} (\Lambda', \mathcal{T})$ and $v \in \theta$ then*
  - *for $\Lambda(r) = (v, b, c, \tau)$ we have $\tau \subseteq \theta$ implying that only delivery stations are left, and*
- *we modify $(\Lambda, \mathcal{T}) \xrightarrow{\delta}_{d,D} (\Lambda', \mathcal{T}')$ s.t. if $\Lambda(r) = (v, \mathit{tt}, c, \tau)$ with $\tau \subseteq \theta$ and $v \in \theta$, then*
  - *$\Lambda'(r) = (v, \mathit{ff}, 0, \emptyset)$, implying that any delivery station v of the task will suffice to complete it.*

The objective remains to minimize the expected makespan of all tasks.

Both Definitions 6 and 7 defines a global scheduling problem: how to assign tasks, where to move AMRs in the environment, and when and where to initiate work. In our work, we leave optimal task assignments to future work and, therefore, we assume a simple, deterministic allocation of tasks in the experiments. This leaves us with the planning of the movement of AMR – but conjecture that global planning is still infeasible, we thus focus on local (distributed), near-future planning. We do so to avoid an explosion in computation-time with a growing fleet of AMRs where the global optimization problem, even for the fully controllable setting, can be shown to be NP-complete (Garey, Johnson, and Sethi 1976), implying an exponential growth in running-time assuming $P \neq NP$. We also note that the problem appears even less tractable, possibly undecidable in the face of stochastic behavior (Kempf, Bozga, and Maler 2013; Geeraerts, Guha, and Raskin 2018). This leads us to select an approach based on near-optimal plans obtained with machine learning techniques but also in dividing the problem into smaller parts.

To overcome the state space explosion, we plan distributed for the individual AMRs, using two levels of abstraction: station planning for learning the sequence to visit stations in and waypoint planning for finding the path to the next station to visit. First, we develop the theoretical foundation for local planning and, second, we define two-level planning.

4 Local, Near-Future Re-Planning

The size of the state space to search correlated with the number of choices. One disadvantage of the global scheduling problem is that several AMRs make choices simultaneously, implying a large branching factor. If the goal is to reduce the total makespan of a set of tasks, the effects of early choices might be important but hard to learn. We, therefore, reduce the problem to just minimizing the makespan of the currently assigned tasks, recomputing the plans whenever one task is completed. However, this approach will still not scale well with increasing numbers of AMRs. We remedy this by planning locally, implying that each AMR plans its own actions given the current plans of the others, thus distributing the planning process. To do so, we need a notion of partial strategies as a plan is only learned for a subset of the actions available in the EMDP. Intuitively, we desire to partition the strategy synthesis s.t. each individual AMR is responsible for its own actions. We, therefore, introduce partial strategies and the partial synthesis problem.

We let $\rho : S \rightarrow (Act' \rightarrow [0,1])$ define be a partial strategy for $Act' \subseteq Act$ and denote by $Act_\rho = Act'$ the supporting set of actions Act' of ρ. Given a set of partial strategies $P = \{\rho^1, \ldots, \rho^n\}$, we say that P is well-formed if $Act_{\rho^i} \cap Act_{\rho^j} = \{\delta\}$ for all $1 \leq i, j \leq n$. If $Act = \bigcup_{1 \leq i \leq n} Act_{\rho^i}$ we say that P is complete.

We say that a full strategy σ conforms to a well-formed set of partial strategies P iff for all $s \in S$ and $\alpha \neq \delta$ we have the following.

$$\sigma_P(s)(\alpha) = \frac{1}{|P|} \sum_{\rho \in P} \rho(s)(\alpha)$$

We let $\sigma_P(s)(\delta)$ be defined in a similar manner only if P is complete.

This intuitively implies that a uniform strategy resolves any conflicting choice between partial strategies, and a "residual" probability is available for the missing, partial strategy. We can thus define the Partial Synthesis Problem.

Definition 8 (Partial Synthesis) *Given an EMDP $\mathcal{M} = (S, Act, s_{init}, T, C, \mathcal{G})$ and an incomplete and well-formed set of partial strategies $P = \{\rho^1, \ldots, \rho^n\}$ compute a full strategy σ_P conforming to P s.t. $\mathbb{E}^{\mathcal{M}}_{\sigma_P}(s_{init})$ is minimized.*

We denote by $\rho = \sigma_P \setminus P$ the partial strategy s.t. $\sigma_P = \sigma_{P \cup \{\rho\}}$. Intuitively this denotes the constructed partial strategy ρ s.t. if ρ is joint with P, it yields the found full strategy for the partial synthesis problem σ_P.

We can now partition the action-space of our SPSWG as follows; we let $Act_r = \{work_{r,v} \in Act \mid v \in V\} \cup \{move_{r,e} \in Act \mid e \in E\} \cup \{\delta\}$ be the action-partitions of AMRs with $r \in R$ and let $Act_{assign} = \{assign_{r,t} \in Act \mid \tau \in \mathcal{T}, r \in R\} \cup \{\delta\}$ be the action-partition of the global task scheduler. Given assumptions of the behaviour on e.g. the scheduler ρ^{assign} and a set of AMRs $\{r_1, \ldots, r_n\} = R \setminus \{r\}$, we can find the optimal partial strategy ρ_r when assuming $P = \{\rho^{assign}_, \rho^1, \ldots, \rho^n\}$.

5 Two-Level Planning

To further reduce the complexity of the optimization problem, for a given task $\tau \in \mathcal{T}$, we separate the optimization of each AMR into two parts: 1) deciding on the order of visiting vertices in τ (*station plan*), and 2) deciding on the exact path between the current position and the next task in τ (*waypoint plan*).

Vertices in a graph can be partitioned into two types: stations we conduct work at (V_ϕ) or vertices (waypoints) that are there to help AMRs navigate around obstacles such as corners (V_ω). The waypoints are not locations where AMRs work but are needed if the graph represents coordinates in a factory setting. For a given SWG $G = (V, E, \Delta)$, we thus let $V_\phi \cup V_\omega = V$.

We here re-use the framework for optimization on SPSWG and define two projections of an SPSWG $S = (R, I, \mathcal{T}, G)$ into S_ϕ and S_ω as the station and waypoint projection, respectively. This assumes a partition of V of the underlying graph $G = (V, E, \Delta)$ into V_ϕ and V_ω, and furthermore, a definition of delivery stations $\theta \subseteq V_\phi$. For simplicity we assume that for all $\tau \in \mathcal{T}$ we have $\tau \subseteq V_\phi$ and that $I(r) \in V_\phi$ for all $r \in R$.

Definition 9 (Waypoint SPSWG) *Given an SPSWG $S = (R, I, \mathcal{T}, G)$ with $G = (V, E, \Delta)$ the graph $S_\omega = (R, I, \{v\}, G)$ is the corresponding waypoint SPSWG to reach a vertice $v \in V$.*

The station graph is more involved and abstracts away details about specific intermediate waypoints. To do so, we abstract away the probability distributions over the traversal times of the edges and abstract away potential congestion problems. This allows us to generate a new graph-based on all-pairs shortest paths between any stations in S_ϕ. We therefore introduce an abstraction over the edge-distribution to single-points, to allow for a shortest-path computation using Dijkstras algorithm and let $(\Delta_1 \oplus \Delta_2)(t) = \int_{\rho=0}^{t} \Delta_1(\rho) \cdot \Delta_2(t - \rho) d\rho$ denote the convolution of the distributions δ_1

and δ_2. Then, we abstract away the points between the stations by defining an Waypoint Abstracted SWG as follows.

Definition 10 (Waypoint Abstracted SWG) *Let $G = (V, E, \Delta)$ be a SWG and let S_ϕ be a set of stations, then $G_\phi = (V', E', \delta')$ is the induced waypoint abstracted graph where*

- $V' = S_\phi$,
- $E' = \{(v_1, v_n) \in V' \mid$ *if there exists a path* $v_1, \dots, v_n$ *in* G *s.t.* $v_i \notin V'$ *for* $1 < i < n$, *and*
- $\Delta' : V' \cup E' \to (\mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0})$ *s.t.*
  - *for* $v \in V'$, *then* $\Delta'(v) = \Delta(v)$, *and*
  - *for* $(v_1, v_n) \in E'$ *then* $\Delta((v_1, v_n))(t) = \oplus_{1 \leq i \leq n-1} \Delta((v_i, v_{i+1}))(t)$ *where* $v_1, \dots, v_n$ *is the shortest path in* G *where* $v_i \notin V'$ *for all* $1 < i < n$ *and using the expectation of each edge-distribution as weight* ($\int_{t=0}^{\infty} \Delta(t)t\, dt$).

This allows us to define the station SPSWG as follows.

Definition 11 (Station SPSWG) *Given an SPSWG* $S = (R, I, \mathcal{T}, G)$ *with* $G = (V, E, \Delta)$, *then* $S_\omega = (R, I, \mathcal{T}, G_\phi)$ *is the station SPSWG where the underlying SWG is replaced by the waypoint abstracted equivalent.*

For convenience, we will denote these two abstractions directly on a given SPSWG S and simply write $\omega(S, v)$ and $\phi(S)$ for the waypoint and the station SPSWG respectively, which naturally yields the EMDPs $\mathcal{M}_\omega$ and $\mathcal{M}_\phi$, respectively. We now have the needed constructions to formally introduce our distributed planning algorithm in Algorithm 1. However, for simplicity, we shall make some simplifying assumptions:

- a global, first-come-first-serve, deterministic planner ρ^{assign} – leaving the optimization of this for future work,
- initially AMRs have the strategy $\rho(s)(\delta) = 1$ for any s,
- the full task-list of the global planner is hidden, and
- the global state Λ is observable and has the projections Λ_ϕ and Λ_ω into the station and waypoint EMDPs, respectively.

6 Uppaal Stratego

The tool UPPAAL STRATEGO is a solver for Stochastic Priced Timed Games (SPTG) (David et al. 2015), a formalism which facilitates a natural encoding of the SPSWG problem. In particular, such SPTG (under certain restrictions) can have their semantics given as EMDPs (Jaeger et al. 2019).

Optimization of SPTG is known to be undecidable as it is undecidable even for the untimed fragment (Brihaye, Bruyère, and Raskin 2005). UPPAAL STRATEGO therefore repeatedly samples episodes from the system and utilizes Q-learning converge to a near-optimal solution. What is particular for the Q-learning method employed by UPPAAL STRATEGO is its ability to conduct partition/refinement of the observed state variables (Jaeger et al. 2019). This allows the tool to generalize the observations onto sets of states, thereby extending the applicability of the Q-learning methods to the continuous domain – which is required to learn near-optimal plans for EMDPs. Examples of UPPAAL STRATEGO models for the station and waypoint plans are provided in the supplementary material.

Algorithm 1: Distributed Planning Algorithm

```
Data: The id of the AMR $r \in R$ and the SPSWG
      $S = (R, I, \emptyset, G)$ with $G = (V, E, \Delta)$
$P_r = \{\rho^{assign}, \dots\}$ is the initial strategies, excluding one for $r$;
while /*loop forever*/ do
    Receive a task $\tau$ from coordinator (according to $\rho^{assign}$);
    while $\tau \neq \emptyset$ do
        Receive updates to $P$ from other AMRs;
        Make an observation $\Lambda$ projected into $\Lambda_\phi$;
        Assume w.log. that $\rho'((\mathcal{T}, \Lambda_\phi))(\delta) = 1$ for all $\rho' \in P$;
        Compute $\sigma_P^\phi$ for $\phi(S)$ starting in state $\Lambda_\phi$ with $\Lambda(r)$ updated with task $t$;
        Let $v$ be the vertex picked by $\sigma_P^\phi$ in $\Lambda_\phi$;
        Send $\sigma_P^\phi \setminus P$ to other AMRs;
        while $\Lambda(r) \neq (v, b, c, \tau)$ for any values of $b, c, \tau$ do
            Receive updates to $P$ from other AMRs;
            Make an observation $\Lambda$ projected onto $\Lambda_\omega$;
            Assume w.log. that $\rho'((\mathcal{T}, \Lambda_\phi))(\delta) = 1$ for all $\rho' \in P$;
            Compute $\sigma_P^\omega$ for $\omega(S, v)$ starting in state $\Lambda_\omega$;
            Send $\sigma_P^\omega \setminus P$ to other AMRs;
            Let $w$ be the vertex picked by $\sigma_P^\omega$ in $\Lambda_\omega$;
            Send coordinates of $w$ to low-level navigation controls;
        end
        Execute work at $v$;
        $\tau \leftarrow \tau \setminus \{v\}$;
    end
end
```

7 Experimental Setup & Results

To evaluate the proposed methods, we have implemented a simulation architecture based on the ARGoS3 framework (Pinciroli et al. 2012), which simulates a physical environment and offers the choice between a 3D visualization and no visualization.[2] For our results, the simulator has no stochastic behavior but contains complex behavior, such as collision avoidance and the bug-tangent algorithm for low-level navigation. We thus use the stochastic nature of our model to capture these imprecisions when translating observations in the simulation into the SPSWG instances. The experimental setup also ensures determinism of the simulation; if the planners produce the same plans in two different executions, the execution will be identical. Due to the stochastic nature of UPPAAL STRATEGO, different executions may lead to different plans, leading to different simulations. The map used in all experiments is shown in Fig. 6.

[2] https://doi.org/10.5281/zenodo.6385690 contains the code, the experiment data, and example Uppaal Models.

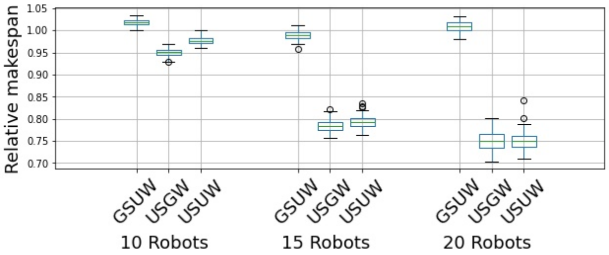

Figure 1: Uniform, 1000 episode budget.

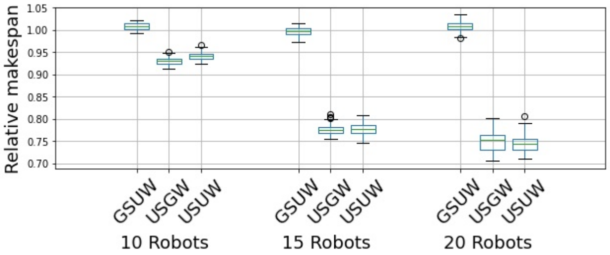

Figure 2: Uniform, 5000 episode budget.

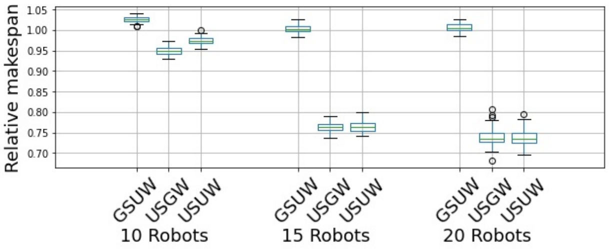

Figure 3: Triangular, 1000 episode budget.

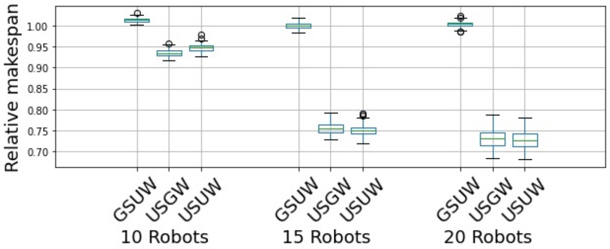

Figure 4: Triangular, 5000 episode budget.

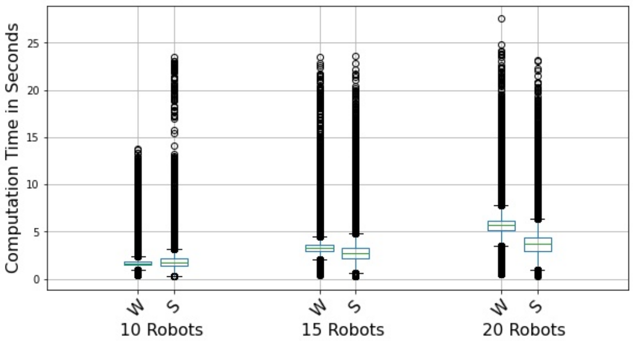

Figure 5: CPU time spent for synthesizing a single waypoint (W) or station (S) plan with a 1000 episode budget.

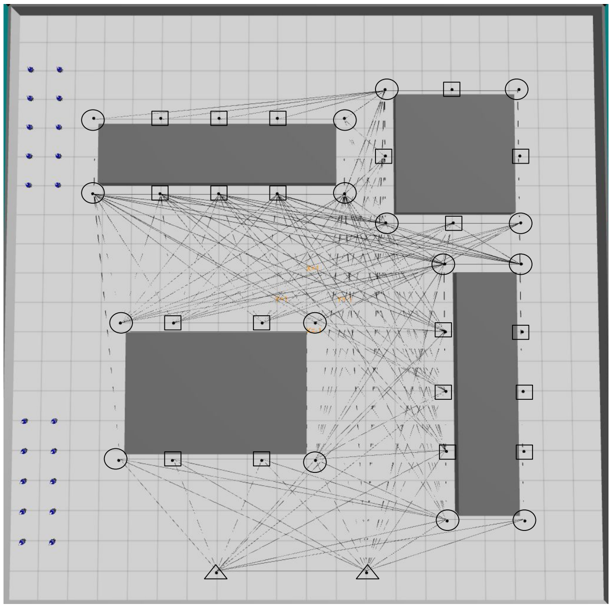

Figure 6: Factory map used in the experiments, containing four static obstacles, 20 workstations (□), 16 waypoints (◯), and two drop-off stations (△). 20 AMRs are placed in two groups to the left.

The experiments use two queues of 200 tasks, where all tasks' size is determined by uniformly sampling between 2 and 4 work stations plus 1 delivery station, and the AMR needs to work for 30-time units in all the locations. We then sample the specific n stations to visit using either a uniform or a triangular distribution for the station allocation. When using the triangular distribution, we sample station i with the probability $\frac{2i}{k(k+1)}$ to be part of the task, assuming k workstations indexed from 1 to k. The triangular distribution represents situations where certain items are requested more often. We use the same queue of 200 tasks in the simulations for all experiments. The task allocator assigns tasks according to a queue and allocates AMRs in a first-come-first-serve manner.

For comparison, we use two greedy approaches for the station and waypoint planning. The greedy station plan always moves towards the nearest working station relative to the current position in its current task. For the greedy waypoint planning, we use Dijkstra's Shortest Path algorithm, where the shortest path is based on edges' euclidean distances. Both greedy planners are deterministic. Our baseline for comparison will then be an AMR planner that uses the greedy station planner and greedy waypoint planner. For a fixed queue of tasks, this yields a deterministic simulation.

We compare the approaches by the total completion time, makespan, for all tasks. In an experiment, all the AMRs on the map will have the same combination of planners, such as UPPAAL STRATEGO-based station planning and greedy waypoint planner. We study changes to four parameters: 1) the planner configuration, 2) the simulation budget for UPPAAL STRATEGO, 3) the number of AMRs on the map, and 4) the generated task queues. We experiment with fleets of 10, 15, and 20 AMRs and study all four waypoint and station planners combinations. If one of the planning levels uses UPPAAL STRATEGO, we repeat the experiment 100 times to assess the stability, as the planner is stochastic in nature. As a simplification, the communicated strategies are not provided in full but simplified to only the most likely sequence in which the stations or waypoints will be visited. The makespans of these experiments will be compared to the fully greedy planner, such that we report a relative improvement of the planning configuration.

It is important to note that the logical time of the simulation is paused whenever a plan is generated, emulating an infinitely fast computer. This implies that the AMR gets a plan instantly, seen from the point of view of the simulator, which allows us to evaluate the effectiveness of the synthesized plans directly. Nonetheless, we will discuss the implications of computation times. We ran the experiments on an AMD EPYC 7642 running Ubuntu 20.04 with hyperthreading and frequency-scaling disabled.

Relative Makespan of Configurations

First, we evaluate experiments by their effect on the relative makespan. In Figs. 1 to 4, the box plots are grouped by the number of AMRs in the given simulation. Each group contains simulations for greedy station planning and UPPAAL STRATEGO waypoint planning (GSUW), UPPAAL STRATEGO station planning, and greedy waypoint planning (USGW), and UPPAAL STRATEGO station planning and UPPAAL STRATEGO waypoint planning (USUW). The y-axis values are the simulations' logical completion time relative to the fully greedy approach with the same number of AMRs. The value reported is *simulated completion time/baseline*, which give us the relative makespan.

Common for all experiments, we observe that the fleet of size 10 experiences the smallest reduction of makespan while a fleet of 20 AMRs results in a much more significant improvement. The median improvement of USUW and USGW is at least 25% better than our baseline. With a task queue generated under a triangular distribution, the median even has a 27.5% reduction in the makespan.

We observe that GSUW's median performs worse for any fleet size under any generated task queue. There can be two causes for this: 1) the generated station plan makes it hard to generate a good waypoint plan that also deviates from the greedy approach, and 2) the chosen representation is not well-formed for the waypoint planning. We leave it to future work to further investigate this hypothesis.

The last aspect is that the increase of UPPAAL STRATEGO's budget only has a minor increase or even a decrease on the relative makespan. An example is that USGW's median decreases slightly from 25.0% to 24.8%, even though the budget is five times larger. It is possible that a budget of 1000 and 5000 would converge to the same median.

Computation Time

The CPU time spent by UPPAAL STRATEGO on synthesizing the individual waypoint or station plan for an episode budget of 1000 is shown in Fig. 5. We observe that the CPU time for waypoint planning grows significantly faster than the CPU time spent for station planning. This is detrimental to the USUW and GSUW approaches that rely on the frequent replanning using the UPPAAL STRATEGO waypoint planner. We also note that the best performing combination (USGW) utilizes the greedy waypoint planner and thus avoids this particular runtime overhead. However, the USGW planner still needs to compute the station plans.

Waypoint plans are computed at every node in the graph, but the CPU time makes neither USUW nor GSUW feasible in a real environment. Having computation times about 10 seconds means that an AMR would need to stop at the waypoint while computing, thus blocking other AMRs.

On the contrary, we see USGW as a feasible candidate for deployment. We anticipate that station plans can be computed simultaneously as background computation while working at a station if the AMR can guess a global state, i.e., 10 seconds from now. This strategy does not extend to the waypoint planning as no movement idle-time can be expected between two adjacent waypoints, and the non-movement of an AMR could block other AMRs.

8 Conclusion

In this paper, we studied the problem of dynamic route planning for a fleet of Autonomous Mobile Robots (AMRs) performing pickup and delivery tasks in the stochastic environment in which AMRs must visit several stations before delivery. This captures some real-world challenges that are not handled by traditional MAPF and MAPD approaches. We formalize the problem as a Scheduling Problem for Stochastic Work Graphs (SPSWG).

We demonstrate how such planning problems can be translated into Euclidean Markov Decision Processes (EMDPs) and (approximately) solved by the tool UPPAAL STRATEGO. To achieve scalability, we propose a distributed architecture and further subdivide the planning problem into two parts: 1) the sequence of workstations to visit, and 2) the sequence of waypoints to visit between two workstations.

To establish a baseline, we introduce two naïve planners (one for each sub-planning problem) based on shortest path algorithms and the Nearest Neighbor approach, which we outperform with up to to 27.5%. Lastly, we argue for the feasibility of deploying our method with the USGW configuration by studying the computation time.

We leave for further work the study of this approach in the context of periodic human interference (e.g. by humans co-inhabiting the working area of the AMRs). We also propose as further work the study of our approach in the context of battery-limitations, where charging cycles must be included as part of the planning.

Acknowledgments

The work was partly funded by Independent Research Fund Denmark under grant number DFF-7017-00348, the Villum Investigator Grant S4OS held by Kim G. Larsen, and by the Danish national research center DIREC (www.direc.dk). We also thank Napalys Kličius for assisting with the development of the Argos3 to Uppaal bindings.

References

Brihaye, T.; Bruyère, V.; and Raskin, J.-F. 2005. On Optimal Timed Strategies. In Pettersson, P.; and Yi, W., eds., *FORMATS*, 49–64. Springer.

Bøgh, S.; Schou, C.; Rühr, T.; Kogan, Y.; Dömel, A.; Brucker, M.; Eberst, C.; Tornese, R.; Sprunk, C.; Tipaldi, G. D.; et al. 2014. Integration and assessment of multiple mobile manipulators in a real-world industrial production facility. In *ISR/Robotik*, 1–8. VDE.

Čáp, M.; Novák, P.; Kleiner, A.; and Seleckỳ, M. 2015. Prioritized planning algorithms for trajectory coordination of multiple mobile robots. *IEEE transactions on automation science and engineering*, 12(3): 835–849.

David, A.; Jensen, P. G.; Larsen, K. G.; Mikučionis, M.; and Taankvist, J. H. 2015. Uppaal Stratego. In Baier, C.; and Tinelli, C., eds., *TACAS*, 206–211. Springer.

De Bruin, T.; Kober, J.; Tuyls, K.; and Babuška, R. 2015. The importance of experience replay database composition in deep reinforcement learning. In *Deep reinforcement learning workshop, NIPS*.

Eriksen, A. B.; Lahrmann, H.; Larsen, K. G.; and Taankvist, J. H. 2019. Controlling Signalized Intersections using Machine Learning. *Transportation Research Procedia*, 48: 987–997. WCTR.

Fazlollahtabar, H.; and Saidi-Mehrabad, M. 2015. Methodologies to optimize automated guided vehicle scheduling and routing problems: a review study. *Journal of Intelligent & Robotic Systems*, 77(3): 525–545.

Garey, M. R.; Johnson, D. S.; and Sethi, R. 1976. The Complexity of Flowshop and Jobshop Scheduling. *Mathematics of Operations Research*, 1(2): 117–129.

Geeraerts, G.; Guha, S.; and Raskin, J.-F. 2018. Safe and Optimal Scheduling for Hard and Soft Tasks. In Ganguly, S.; and Pandya, P., eds., *FSTTCS*, LIPIcs, 36:1–36:22. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Goorden, M. A.; Larsen, K. G.; Nielsen, J. E.; Nielsen, T. D.; Rasmussen, M. R.; and Srba, J. 2021. Learning Safe and Optimal Control Strategies for Storm Water Detention Ponds. In Jungers, R. M.; Ozay, N.; and Abate, A., eds., *ADHS*, IFAC-PapersOnLine, 13–18. Elsevier.

Gu, R.; Enoiu, E.; Seceleanu, C.; and Lundqvist, K. 2020. Probabilistic Mission Planning and Analysis for Multi-agent Systems. In Margaria, T.; and Steffen, B., eds., *ISoLA*, 350–367. Springer.

Hönig, W.; Kiesel, S.; Tinka, A.; Durham, J. W.; and Ayanian, N. 2019. Persistent and robust execution of MAPF schedules in warehouses. *IEEE Robotics and Automation Letters*, 1125–1131.

Hönig, W.; Kumar, T. S.; Cohen, L.; Ma, H.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Multi-agent path finding with kinematic constraints. In *ICAPS*, 477–485.

Jaeger, M.; Bacci, G.; Bacci, G.; Larsen, K. G.; and Jensen, P. G. 2020. Approximating Euclidean by Imprecise Markov Decision Processes. In Margaria, T.; and Steffen, B., eds., *ISoLA*, 275–289. Springer.

Jaeger, M.; Jensen, P. G.; Larsen, K. G.; Legay, A.; Sedwards, S.; and Taankvist, J. H. 2019. Teaching stratego to play ball: Optimal synthesis for continuous space MDPs. In *ATVA*, 81–97. Springer.

Kempf, J.-F.; Bozga, M.; and Maler, O. 2013. As Soon as Probable: Optimal Scheduling under Stochastic Uncertainty. In Piterman, N.; and Smolka, S. A., eds., *TACAS*, 385–400. Springer.

Larsen, K. G.; Mikučionis, M.; Muñiz, M.; Srba, J.; and Taankvist, J. H. 2016. Online and Compositional Learning of Controllers with Application to Floor Heating. In Chechik, M.; and Raskin, J.-F., eds., *TACAS*, 244–259. Springer.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. S.; and Koenig, S. 2020. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *AAMAS*, 1898–1900.

Lin, K.; Zhao, R.; Xu, Z.; and Zhou, J. 2018. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *ACM SIGKDD*, 1774–1783.

Ling, J.; Gupta, T.; and Kumar, A. 2020. Reinforcement Learning for Zone Based Multiagent Pathfinding under Uncertainty. In *ICAPS*, 551–559.

Liu, H.; Hyodo, A.; Akai, A.; Sakaniwa, H.; and Suzuki, S. 2021. Action-limited, Multimodal Deep Q Learning for AGV Fleet Route Planning. In *CCEAI*, 57–62.

Ma, H.; Li, J.; Kumar, T.; and Koenig, S. 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv preprint arXiv:1705.10868*.

Oda, T.; and Joe-Wong, C. 2018. Movi: A model-free approach to dynamic fleet management. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, 2708–2716. IEEE.

Pinciroli, C.; Trianni, V.; O'Grady, R.; Pini, G.; Brutschy, A.; Brambilla, M.; Mathews, N.; Ferrante, E.; Di Caro, G.; Ducatelle, F.; Birattari, M.; Gambardella, L. M.; and Dorigo, M. 2012. ARGoS: a Modular, Parallel, Multi-Engine Simulator for Multi-Robot Systems. *Swarm Intelligence*, 271–295.

Salzman, O.; and Stern, R. 2021. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *AAMAS*, 1711–1715.

Schiøler, H.; Totu, L.; Dimon, J.; Larsen, K. G.; and Taankvist, J. H. 2018. Time Optimal Robust Fleet Management of Micro UAV Through Timed Games Formulation. In *IEEE CCTA*, 146–152. IEEE.

Staal, A. S.; Salvatierra, C. G.; Albertsen, D. D.; Mahendran, M.; Ravichandran, R.; Thomsen, R. F.; Hansen, E. B.; and Bøgh, S. 2020. Towards a Collaborative Omnidirectional Mobile Robot in a Smart Cyber-Physical Environment. *Procedia Manufacturing*, 193–200.