Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Multi-Robot Planning Under Uncertainty with Congestion-Aware Models

Charlie Street, Bruno Lacerda, Manuel Mühlig, Nick Hawes

2020

Preprint:

This is an accepted article published in Nineteenth International Conference on Autonomous Agents and Multi-Agent Systems . The final authenticated version is available online at: https://doi.org/[DOI not available]

Multi-Robot Planning Under Uncertainty with Congestion-Aware Models

Charlie Street
Oxford Robotics Institute, University of Oxford
Oxford, UK
cstreet@robots.ox.ac.uk

Bruno Lacerda
Oxford Robotics Institute, University of Oxford
Oxford, UK
bruno@robots.ox.ac.uk

Manuel Mühlig
Honda Research Institute Europe GmbH
Offenbach, Germany
manuel.muehlig@honda-ri.de

Nick Hawes
Oxford Robotics Institute, University of Oxford
Oxford, UK
nickh@robots.ox.ac.uk

ABSTRACT

When planning for multi-robot navigation tasks under uncertainty, plans should prevent robots from colliding while still reaching their goal. Solutions achieving this fall on a spectrum. At one end are solutions which prevent robots from being in the same part of the environment simultaneously at planning time, ignoring the robots' capabilities to manoeuvre around each other, whilst at the other end are solutions that solve the problem at execution time, relying solely on online conflict resolution. Both approaches can lead to inefficient behaviour. In this paper, we present a novel framework in the middle of this spectrum that explicitly reasons over the effect the presence of multiple robots has on navigation performance. We refer to this effect as *congestion*. We present a structure, called the *probabilistic reservation table*, which summarises the plans of robots, allowing us to probabilistically model congestion. We show how this structure can be used for planning by proposing an approach that, for each robot, sequentially builds and solves a Markov decision process where the transition probabilities are obtained by querying the probabilistic reservation table. We carry out experiments on synthetic data and in simulation to show the effectiveness of our framework.

KEYWORDS

Multi-robot systems; Planning under uncertainty; Markov models; Temporal uncertainty

ACM Reference Format:
Charlie Street, Bruno Lacerda, Manuel Mühlig, and Nick Hawes. 2020. Multi-Robot Planning Under Uncertainty with Congestion-Aware Models. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

In recent years, multi-robot systems have been developed for warehouses [13], agriculture [15] and roads [20]. In these environments, the simultaneous presence of multiple robots in the same area causes an increase in uncertainty over navigation performance. We refer to the effect of the presence of other robots on the execution of navigation actions as *congestion*. For example, Figure 1 shows a warehouse where robots travelling along the same aisle simultaneously will cause congestion. If the aisle is wide enough, the robots can manoeuvre around each other, with this deviation incurring some cost. If the aisle is too narrow however, one of the robots may have to turn back or wait for an unknown amount of time. In the worst case a robot may become stuck and unable to navigate.

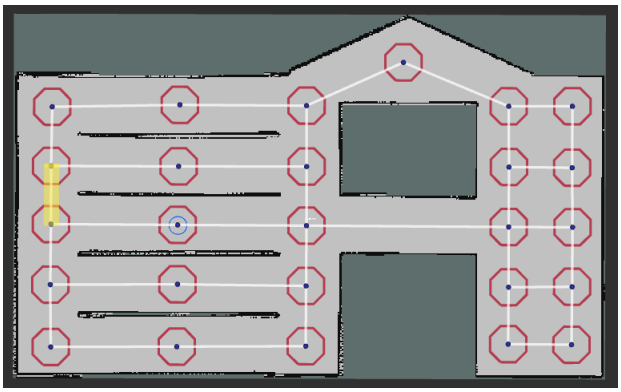


Figure 1: A warehouse environment with a topological map overlaid. The blue circle shows the size of a robot in the map.

Existing methods that deal with execution-time interactions between robots fall on a spectrum. On one end are solutions to the multi-agent path finding (MAPF) problem, which create plans that prevent robots from ever being at the same location at the same time [18]. By solving the problem entirely at planning time, the navigation capabilities of the robots, such as their ability to manoeuvre around each other, are ignored. This can result in inefficient plans which are too conservative and keep the robots far away from each other. On the other end of the multi-robot planning spectrum are motion planning solutions which solve the problem solely at execution time, by relying on the robots' ability to move around each other [3]. However, such solutions do not consider the potential effects of other robots on navigation performance – in some cases it may prove more efficient to avoid areas with higher robot density.

In this paper, we propose a method that acts in the middle of the multi-robot planning spectrum. We reason over the expected

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

effects of congestion at planning time, allowing robots to take longer but less congested routes if that proves more efficient. This assumes a motion planner that can manoeuvre the robots around each other in some cases. We model the duration of navigation actions as a continuous stochastic process through the use of phase-type distributions (PTD) [11]. Given policies defining the routes of each robot, we can use the PTDs to construct a continuous-time Markov chain (CTMC) for each robot, which allows us to compute the probabilities of different levels of congestion occurring at any location and time. The PTDs, CTMCs and congestion information are managed in a structure called the *probabilistic reservation table* (PRT). Planning is carried out on single-robot Markov decision processes (MDP) that include the influence of the other robots via the congestion probabilities obtained from the PRT.

The primary contribution of this paper is the PRT, a novel structure for modelling congestion. As a secondary contribution we present a framework for multi-robot planning under uncertainty that demonstrates the benefits of modelling the behaviour of robot teams in continuous time with the PRT.

2 RELATED WORK

The multi-agent Markov decision process (MMDP) is an exact model for multi-agent planning under uncertainty [8]. Though optimal behaviour can be synthesised from MMDPs, the state space scales exponentially with the number of robots. Furthermore, MMDPs require robots to act synchronously. Synchronous execution leads to suboptimal execution-time behaviour, due to robots having to wait unnecessarily [28]. In this work we consider actions with continuous stochastic durations, and want robots to act asynchronously.

One method for tackling the problem of scalability in multi-robot planning is to plan on single-robot models with some knowledge of the other robots. Claes et al. [13, 14] do this by aggregating the responses of the other robots via assumptions such as all other robots being self interested. Zhang et al. [40] also plan on single-robot models using an iterative procedure where robots consider their team mates more with every iteration. However, this procedure assumes no uncertainty in action execution, with the output being a sequence of actions, which are less robust to the probabilistic nature of the environment. In this paper, we approximate multi-robot behaviour in single-robot models via congestion, and generate policies for each robot *sequentially*, with each robot considering those who plan before it. Such an approximation is common in multi-robot planning [3, 17, 35]. Similar to [13, 14], our approach computes a *best-response* for each robot. The best-response is the best action a robot can take, assuming the plans of the other robots are fixed. The quality of plans produced under a sequential planning assumption is very sensitive to the ordering of agents [35], and determining an optimal ordering is an NP-hard problem [4]. However, effective priority orderings have been generated using heuristics [37], as well as optimisation methods [5], which search through the space of possible priority orderings. Determining a priority ordering is not addressed in this paper.

The requirement of synchronisation in MMDPs can be removed through the use of macro actions [1], which are high level behaviours executed asynchronously, with synchronisation occurring at the level of the controllers that form the macro actions. Alternatively, action durations can be modelled as continuous stochastic processes, with shifted Poisson distributions [40], exponential distributions [16, 26], or arbitrary temporal distributions [28].

There exist Markov models which consider continuous time, such as *time-dependent* MDPs [9], however approaches to solve them either scale badly [32] or rely on strong simplifications [24] that are unsuitable for the problems presented in this paper, such as deterministic models of action duration.

Solutions to the MAPF problem generate paths for robots to reach their goals in a discretised environment while not colliding with each other [18]. Silver [35] solves this problem by using a structure called a *reservation table* to store the route information of the robots, such that subsequent robots can avoid those who have planned previously. In this paper, we expand upon the reservation table to allow actions with continuous stochastic durations. MAPF solutions commonly assume deterministic environments, though uncertainty has been considered. In [39], a belief space is used, giving distributions that appear as 'tails' over the robot's location. The M* MAPF solver [38] is then adapted to plan in the belief space of each agent, with coordination occurring between robots likely to collide. In [25], when a robot attempts to navigate, it fails with some probability, remaining stationary. To handle this, a set of critical dependencies between robots is computed, which force some robots to wait until another robot has reached a certain location. Continuous time has been considered for MAPF in [2]. The conflict-based search (CBS) algorithm [33] is adapted by replacing A* search with safe interval path planning [30] for low-level path planning. Contrary to our work however, the continuous action durations are assumed to be fixed with no stochasticity. Finally, in [34], the M* solver is adapted to have soft collision constraints, i.e. collisions are allowed to occur, at some cost.

3 PRELIMINARIES

We represent the set of distributions over set X by $Dist(X)$.

Topological Map. We represent the environment using a *topological map* with probability distributions over navigation duration.

Definition 3.1. A topological map is a tuple $\mathcal{T} = \langle V, E, \rho \rangle$, where: V is a finite set of nodes representing locations in the environment; $E \subseteq V \times V$ is a set of bidirectional edges which robots can travel on; and $\rho : E \times \mathbb{N} \to Dist(\mathbb{R}_{\geq 0})$ is a function that takes an edge and the number of robots present on that edge, and returns a distribution over the duration for an additional robot to traverse that edge.

Markov Decision Processes (MDPs). In this paper, we address *stochastic shortest path* (SSP) problems, which are typically represented as MDPs.

Definition 3.2. An MDP [31] is a tuple $\mathcal{M} = \langle S, \bar{s}, A, \delta \rangle$, where S is a finite set of states, $\bar{s} \in S$ is the initial state, A is a finite set of actions and $\delta : S \times A \times S \to [0, 1]$ returns the probability of arriving at state s' after taking action a in state s.

If there is at most one action available in each state, the MDP becomes a *discrete-time Markov chain* (DTMC).

Definition 3.3. An SSP [27] is defined as an MDP $\mathcal{M} = \langle S, \bar{s}, A, \delta \rangle$, a cost structure $c : S \times A \to \mathbb{R}_{>0}$ and a set of

goal states $G \subseteq S$. The goal of an SSP is to find a policy $\pi : S \to A$ that minimises the expected cost to reach a state in G.

Note that a policy π over an MDP $\mathcal{M}$ induces a DTMC $\mathcal{M}^\pi$, where, at each state, the only action available is the policy action.

Continuous-Time Markov Chains (CTMCs). A CTMC models the continuous-time evolution of a system.

Definition 3.4. A (labelled) CTMC [22] is a tuple $\mathcal{Q} = \langle S, init, \delta_E, AP, Lab \rangle$, where S is a finite set of states, $init : S \to [0, 1]$ gives the probability of a state being the initial state, $\delta_E : S \times S \to \mathbb{R}_{\geq 0}$ is a rate transition function, AP is a finite set of atomic propositions, and $Lab : S \to 2^{AP}$ is a labelling function that maps states to atomic propositions that hold in that state.

The value of $\delta_E(s, s')$ is the *rate* parameter of an exponential distribution associated with the transition. Thus, the probability that the transition fires within time t is $1 - e^{-\delta_E(s,s') \cdot t}$. The *exit rate* of a state s in $\mathcal{Q}$ is the sum of all outgoing rates from s, $E(s) = \sum_{s' \in S} \delta_E(s, s')$. The probability of leaving state s within time t is then $1 - e^{-E(s) \cdot t}$.

Phase-Type Distributions (PTDs). PTDs approximate non-negative continuous distributions using the time taken to reach an absorbing state in a CTMC.

Definition 3.5. A PTD [29] is a tuple $\mathcal{P} = \langle S, init, \delta_E, s^a \rangle$, where S, $init$ and δ_E are as in a CTMC, and $s^a \in S$ is the single absorbing state, i.e. there are no transitions from s^a. Further, $init(s^a) = 0$.

Note that in a PTD, the absorbing state cannot be an initial state. We denote the expected time for $\mathcal{P}$ to reach the absorbing state as $\mathbb{E}[\mathcal{P}]$, and the set of non-absorbing states of $\mathcal{P}$ as $NA(\mathcal{P})$ (the same applies for a CTMC). Furthermore, the set of all PTDs is denoted $\mathbb{P}$.

To avoid ambiguity, we use a subscript to notate an element belonging to a model, e.g. $S_{\mathcal{P}}$ to represent the state space of PTD $\mathcal{P}$, or $\delta_{\mathcal{M}}$ to represent the transition function of MDP $\mathcal{M}$.

4 OVERVIEW OF APPROACH

In this section, we formulate the congestion-aware planning problem and introduce the framework we propose to address it.

Problem 1. *Let $R = \{r_1, ..., r_n\}$ be a set of robots acting on a topological map $\mathcal{T} = \langle V, E, \rho \rangle$, where r_i has initial and goal locations $v_i^{init}, v_i^{goal} \in V$. Find policies $\{\pi_1, ..., \pi_n\}$ that minimise the makespan, i.e. the time until the last robot reaches its goal.*

We assume navigation actions always succeed and lead the robot to the desired location. Therefore, the uncertainty arises from the temporal uncertainty over the duration of navigation actions.

We now describe our general approach to approximate solutions for Problem 1. There is one global entity, the probabilistic reservation table (PRT). Planning takes place on single-robot models that take into account probabilistic information about the other robots. In particular, we take an approximate approach to planning by minimising the expected time to reach the goal on a *time-dependent* MDP which uses expected values for the duration of navigation actions. In order to plan on single-robot models while not treating robots independently, we assume robots plan sequentially, following a pre-defined priority order, which we assume, w.l.o.g., to be according to the robots' indices. Thus, robot r_i waits until previous robots r_1 to r_{i-1} have updated the PRT with their policies. Next, r_i takes this information from the PRT, constructs a planning model, and solves this to obtain a policy π_i which minimises the expected time to reach v_i^{goal}, given the information about the routes of the robots that have planned previously. Information about π_i is then inserted into the PRT, so r_{i+1} can start planning. Once all robots have planned, the policies are executed concurrently.

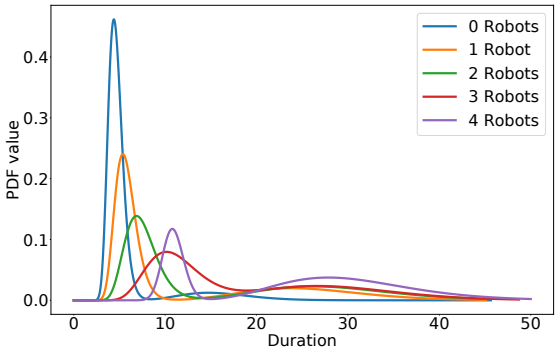


Figure 2: A set of distributions obtained from the highlighted edge of the map in Figure 1. The method for obtaining this data is described in Section 7.

5 MODELLING CONGESTION

In this section, we introduce the PRT as a model of congestion that considers actions with continuous and stochastic durations.

5.1 Modelling Action Durations

Recall that we model the environment as a topological map $\mathcal{T} = \langle V, E, \rho \rangle$, where $\rho(e, m)$ represents the continuous distribution over the duration of traversing $e \in E$ with m other robots on the edge. In this work, we use PTDs to approximate these duration distributions. Figure 2 shows a set of PTDs for an edge in Figure 1.

5.2 Congestion Bands

We model levels of congestion in terms of *congestion bands*.

Definition 5.1. Let $e \in E$ and n be the total number of robots. A set of congestion bands $C_e = \{c_e^0, c_e^1, ..., c_e^b\}$ is such that:

- $c_e^j = [lb_e^j, ub_e^j]$. A congestion band is an interval on the number of robots, represented by a lower and an upper bound. If k robots are present on e, where $lb_e^j \leq k \leq ub_e^j$ then c_e^j is the congestion level on that edge.
- $c_e^0 = [0, 0]$, i.e. a congestion band considering 0 other robots is always present.
- $c_e^b = [lb_e^b, n - 1]$. The last congestion band has an upper bound of $n - 1$, since at most $n - 1$ other robots can be present on an edge.
- $lb_e^{j+1} = ub_e^j + 1$. Congestion bands do not intersect and each possible number of robots fits into exactly one band.

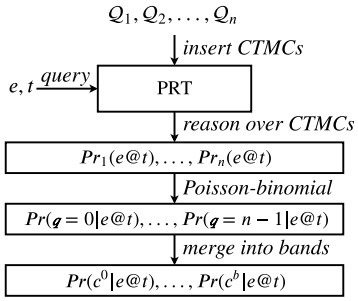


Figure 3: The pipeline for computing the probabilities of each congestion band on edge e at time t using the PRT. Note, there may not be a CTMC for all robots at the time of use.

We use congestion bands because there are cases where we do not observe a significant difference in the effects of congestion between similar numbers of robots (cf. Figure 2). Using the notion of congestion bands, we introduce a topological map under congestion.

Definition 5.2. A topological map under congestion is a tuple $\mathcal{T}_C = \langle V, E, C, \rho_C \rangle$, where: $C = \bigcup_{e \in E} C_e$; and $\rho_C : E \times C \to \mathbb{P}$ is a function that maps an edge and congestion band (for that edge) to a PTD over the duration of traversing that edge.

Given enough data, a larger number of congestion bands gives a more accurate measure of congestion while increasing the complexity of planning. Furthermore, note that if we consider one congestion band per possible number of robots, we recover Definition 3.1, but now using PTDs to approximate the duration distributions. This allows us to exploit efficient algorithms for fitting PTDs [36], whilst also allowing for the interpretation of robot policies as CTMCs, as we will describe next. To simplify notation, we will omit subscript e from congestion band c_e^j if e can be inferred from the context.

5.3 Computing Congestion Probabilities

To support planning, we must compute the probability of each congestion band occurring on an edge e at time t. The process for this is shown in Figure 3.

To compute the congestion probabilities, we require information about the routes of the robots. We use a CTMC $\mathcal{Q}_i$ to model the route of robot r_i through the topological map. CTMCs are a natural representation for the continuous-time behaviour of the robots. By modelling the edge navigation durations as PTDs, we can construct a CTMC modelling a robot's route by connecting the PTDs. We can then exploit methods to analyse transient properties in CTMCs [22] to compute the probability of observing a congestion band on an edge at a specific time. We describe the PTD connection process and subsequent planning procedure in Section 6.

We require the CTMC to be constructed with a labelling function that represents the robot's route through the topological map.

Definition 5.3. Let $\mathcal{T}_C$ be a topological map under congestion. A route CTMC is a tuple $\mathcal{Q} = \langle S, init, \delta_E, AP, Lab \rangle$, where $AP = E \cup \{goal\}$, i.e. the atomic propositions are the edges of $\mathcal{T}_C$ as well as a *goal* proposition; and *Lab* is defined such that all non-absorbing states in $\mathcal{Q}$, denoted $NA(\mathcal{Q})$, are labelled with a single edge of the topological map, i.e. the following condition holds:

$$\forall s \in NA(\mathcal{Q}),\ |Lab(s)| = 1 \wedge goal \notin Lab(s) \tag{1}$$

States labelled with edge $e \in E$ in a route CTMC $\mathcal{Q}_i$ represent states for which r_i is present on e. This is because states in the route CTMC are the union of the states of a set of PTDS, where each PTD represents an edge in r_i's route. Thus, each state can be mapped to the edge the PTD it comes from is modelling.

If an absorbing state of the route CTMC represents the successful completion of a robot's route it can be labelled with proposition *goal*. Although left to future work, this opens up the possibility of formally verifying policies, e.g. computing the probability of a robot reaching its goal within a time bound.

With a set of route CTMCs stored in the PRT, the first step to computing the congestion probabilities is to consider the probability that each robot r_i is present on edge e at time t, denoted $Pr_i(e@t)$. This is shown in the second box of Figure 3.

To compute $Pr_i(e@t)$, we analyse the route CTMC $\mathcal{Q}_i$ for robot r_i by computing the *transient probabilities* over the CTMC, which returns the probability of being in each state $s \in S_{\mathcal{Q}_i}$ at time t, denoted $Pr(s@t)$. We can use this information to calculate the probability $Pr_i(e@t)$ of robot r_i being present on edge e at time t:

$$Pr_i(e@t) = \sum_{\{s \in S_{\mathcal{Q}_i} \mid e \in Lab_{\mathcal{Q}_i}(s)\}} Pr(s@t) \tag{2}$$

In practice, existing software can be used to obtain the transient probabilities. In this work, we use the PRISM model checker [23].

We compute the edge presence probabilities using only the route CTMCs of individual robots. This assumes that all necessary congestion information for r_i is contained in $\mathcal{Q}_i$. However, when new route CTMCs are added to the PRT, the true probabilities of congestion will change. By making the assumption that the robots can be considered sequentially, we treat a robot's route as fixed at the point of insertion into the PRT. Whilst this does not hold in practice, this is part of the modelling assumption we make. To reason over the true congestion probabilities would require a joint model, which we are explicitly avoiding in favour of scalability. Furthermore, as we show in the experiments in Section 7, this assumption still allows our planning framework to effectively reason over congestion.

The next step in computing the congestion probabilities is to use the presence probabilities to compute a discrete distribution over how many *other* robots are on edge e at time t, i.e. a distribution over how congested an edge will be. This can be seen in the third box in Figure 3. To compute this, we use the Poisson-binomial distribution, which models the outcome of n Bernoulli trials, with the random variable differing in each trial [6]. Here, the Bernoulli random variables are over the presence of each robot on edge e at time t, and are distributed by the probabilities $Pr_i(e@t)$. We denote the probability given by the Poisson-binomial distribution of q other robots being on edge e at time t by $Pr(q \mid e@t)$.

The final step in computing the congestion probabilities is to merge the probabilities over the number of robots on edge e at time t into congestion bands, as seen in the bottom box of Figure 3. The probability $Pr(c^j \mid e@t)$ of experiencing congestion band c^j

on edge e at time t is given by:

$$Pr(c^j \mid e@t) = \sum_{q=lb_e^j}^{ub_e^j} Pr(q \mid e@t) \tag{3}$$

In some cases, very small congestion probabilities are observed that bear no practical gain, yet increase the complexity of planning. Therefore, we define a *pruning threshold* ε and set all congestion probabilities below ε to 0. The distribution is then normalised.

5.4 Defining the PRT

A PRT is initialised with the set of robots R, the topological map $\mathcal{T}_C$, and pruning threshold ε. A PRT is a table with an entry for each robot $r_i \in R$. An entry in the table, PRT(r_i), stores a route CTMC $\boldsymbol{Q}_i$ representing r_i's route through the topological map. Two functions are provided by the PRT: *insert* and *cong*. The function $insert(r_i, \boldsymbol{Q}_i)$ inserts route CTMC $\boldsymbol{Q}_i$ into the entry for r_i in the PRT. The function $cong(r_i, e, j, t)$ returns the probability that r_i experiences congestion band c^j on edge e at time t, calculated according to the method described in Figure 3. Note that *cong* returns the congestion probability for r_i given the entries of all other robots in the PRT, i.e., *cong* reflects the current state of the PRT rather than the congestion information fixed in $\boldsymbol{Q}_i$.

6 PLANNING UNDER CONGESTION

6.1 Obtaining Robot Policies

The planning problem for each robot r_i is to synthesise a policy that minimises the expected time to reach v_i^{goal}, taking into account the congestion probabilities and the continuous stochastic models of action duration. In order to tackle this time dependency whilst still being able to scale to reasonable sized models, we build an approximate MDP per robot where each edge traversal has an outcome per congestion band, and the duration of this outcome is the expected duration of the congestion band.

Definition 6.1. Given topological map $\mathcal{T}_C = \langle V, E, C, \rho_C \rangle$, time bound $T \in \mathbb{R}_{\geq 0}$, and the PRT, we define robot r_i's MDP as $\mathcal{M}_i = \langle S, \bar{s}, A, \delta \rangle$, where:

- $S = V \times \mathbb{R}_{\geq 0}$. States are a product of a topological map node and expected time of arrival to that node.
- $\bar{s} = (v_i^{init}, 0)$, i.e. the robot's initial location at time 0.
- $A = E$. The actions are the edges of the topological map.
- The transition function δ is defined such that for each congestion band on an edge, there is a successor state defined by the location reached from that edge, and time value equal to the previous state's time plus the expected duration of the edge under that congestion band. Robots are assumed to start the next edge immediately after finishing the previous one. The probability of reaching this successor is the probability of that congestion band being experienced at the time of arrival to that edge. Formally, for states $s = (v, t)$, $s' = (v', t')$, and edge e:

$$\delta(s, e, s') = \begin{cases} cong(r_i, e, j, t) & \text{if } e = (v, v'),\ t < T \text{ and} \\ & t' = t + \mathbb{E}[\rho_C(e, c^j)] \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where $j \in \{0, ..., |C_e| - 1\}$.

By representing time in the state, the state space becomes infinite. However, the state space is also discrete as the feasible time values for an MDP state are based on sums of expected values of the PTDs. Therefore, to keep the state space finite, we introduce a bound $T \in \mathbb{R}_{\geq 0}$, where any MDP state with a time value greater than T is made a dead end, i.e. have no successors. This makes the state space finite as any path that has not reached the goal by T, such as one looping around two edges indefinitely, is cut off. We now define the planning problem for r_i as an SSP over $\mathcal{M}_i$.

Definition 6.2. The SSP for robot r_i is defined as:

- An MDP $\mathcal{M}_i$ according to Definition 6.1.
- A cost function $c_i : (V \times \mathbb{R}_{\geq 0}) \times E \to \mathbb{R}_{>0}$ that gives the expected duration over all congestion bands:

$$c_i((v, t), e) = \sum_{j \in \{0, ..., |C_e|-1\}} cong(r_i, e, j, t) \cdot \mathbb{E}[\rho_C(e, c^j)] \tag{5}$$

- A set of goal states G_i defined as:

$$G_i = \{(v, t) \in V \times \mathbb{R}_{\geq 0} \mid v = v_i^{goal} \text{ and } t < T\} \tag{6}$$

To solve the SSP efficiently, we use labelled real time dynamic programming (LRTDP) [7]. LRTDP is a trial-based heuristic search method for MDPs that uses an admissible heuristic $h_i : S \to \mathbb{R}_{\geq 0}$ to initialise the value of states, so states unlikely to contribute to the optimal policy are not explored. We define this heuristic as:

$$h_i((v, t)) = min_cost(v, v_i^{goal}) \tag{7}$$

In Equation 7, *min_cost* returns the minimal cost to travel from v to v_i^{goal}, which can be computed using the Floyd-Warshall algorithm [19]. The cost of each edge e in *min_cost* is set to $\mathbb{E}[\rho_C(e, c^0)]$, the expected duration assuming no congestion.

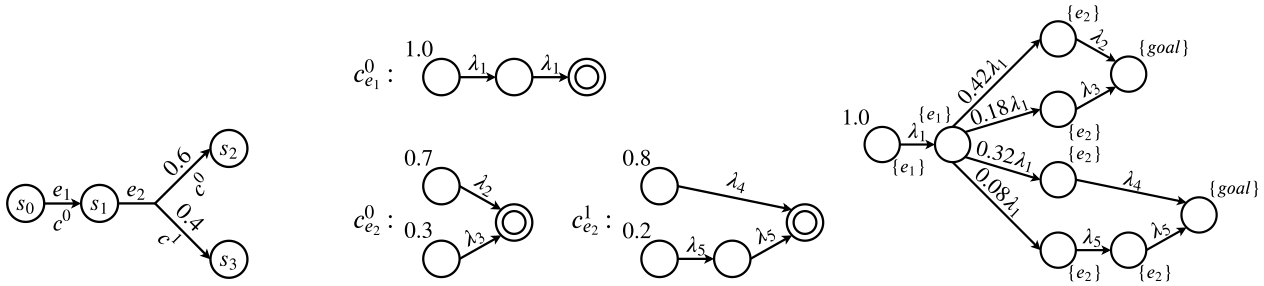
If T is set higher than any realistic cost of reaching the goal, the model used for planning is a finite-horizon SSP with *avoidable* dead ends, as there always exists a policy from the initial state that can reach a goal state while never visiting a dead end. Thus, LRTDP trials that visit a dead end will always terminate, and so it can be used to synthesise optimal policies [21].

To execute the obtained policies, the executor needs to be able to define the current state. This poses an issue when time is included in the MDP state as a robot will never arrive at a node at precisely a time in one of the states. To solve this, we choose the successor with time value closest to the true arrival time. This ensures we can always identify the current state to obtain the policy action.

6.2 Constructing a Robot's Route CTMC

Given our presented planning solution, in Algorithm 1 we detail how to construct a route CTMC $\boldsymbol{Q}_i$ for robot r_i that models the execution of policy π_i on MDP $\mathcal{M}_i$. Algorithm 1 modifies the algorithm presented in [12] for cases where non-determinism is resolved.

The first step is to compute the MDP transitions reachable under policy π_i, as seen in line 2. This is equivalent to the transitions of the induced DTMC. To build $\boldsymbol{Q}_i$, we then iterate over these transitions. Each transition is associated with a single PTD, which is added to $\boldsymbol{Q}_i$ in lines 5-7. Function `getPTD` returns the PTD for a transition, and `addStatesAndTransitions` adds all states in $NA(\mathcal{P})$ (the non-absorbing states of $\mathcal{P}$) to $\boldsymbol{Q}_i$ as well as the transitions between them. These states are then labelled with the edge $\mathcal{P}$ models.



(a) The induced DTMC. **(b) PTDs for the edges used in the MDP/DTMC.** **(c) The route CTMC.**

Figure 4: The process of generating a route CTMC from a robot's policy.

Algorithm 1: Route CTMC Construction

Input : MDP $\mathcal{M}_i$, policy π_i, PTDs $\mathbb{P}$, topological map $\mathcal{T}_C$
Output: Route CTMC $\mathcal{Q}_i$

1 **begin**
2 $Tr \leftarrow \{(s, s') \in S_{\mathcal{M}_i} \times S_{\mathcal{M}_i} \mid \delta_{\mathcal{M}_i}(s, \pi_i(s), s') > 0\}$
3 **for** $(s, s') \in Tr$ **do**
4 // **Add the PTD for the transition**
5 $\mathcal{P} \leftarrow$ `getPTD`(s, s')
6 `addStatesAndTransitions`$(NA(\mathcal{P}))$
7 $\forall_{s_\mathcal{P} \in NA(\mathcal{P})}, Lab_{\mathcal{Q}_i}(s_\mathcal{P}) = \{\pi_i(s)\}$
8 // **If** s' **is a dead-end or goal state in** $\mathcal{M}_i$
9 **if** s' *has no action in* π_i **then**
10 `addAbsorbingStateAndTransitions`$(s^a_\mathcal{P})$
11 **if** s' *is a goal state in* $\mathcal{M}_i$ **then**
12 $Lab_{\mathcal{Q}_i}(s^a_\mathcal{P}) = \{goal\}$
13 // **Set the initial distribution of** $\mathcal{Q}_i$
14 **if** $s = \bar{s}_{\mathcal{M}_i}$ **then**
15 **for** $s_\mathcal{P} \in NA(\mathcal{P})$ **do**
16 $init_{\mathcal{Q}_i}(s_\mathcal{P}) = \delta_{\mathcal{M}_i}(s, \pi_i(s), s') \cdot init_\mathcal{P}(s_\mathcal{P})$
17 // **Connect the PTDs**
18 $pre \leftarrow \{s'' \in S_{\mathcal{M}_i} \mid \delta_{\mathcal{M}_i}(s'', \pi_i(s''), s) > 0\}$
19 **for** $s'' \in pre$ **do**
20 $\mathcal{P}_{pre} \leftarrow$ `getPTD`(s'', s)
21 **for** $s_{\mathcal{P}_{pre}} \in NA(\mathcal{P}_{pre})$ **do**
22 **for** $s_\mathcal{P} \in NA(\mathcal{P})$ **do**
23 $\delta_{E,\mathcal{Q}_i}(s_{\mathcal{P}_{pre}}, s_\mathcal{P}) = \delta_{E,\mathcal{P}_{pre}}(s_{\mathcal{P}_{pre}}, s^a_{\mathcal{P}_{pre}}) \cdot \delta_{\mathcal{M}_i}(s, \pi_i(s), s') \cdot init_\mathcal{P}(s_\mathcal{P})$
24 **return** $\mathcal{Q}_i = \langle S_{\mathcal{Q}_i}, init_{\mathcal{Q}_i}, \delta_{E,\mathcal{Q}_i}, E \cup \{goal\}, Lab_{\mathcal{Q}_i} \rangle$

By default, the absorbing state of PTD $\mathcal{P}$ is not added to $\mathcal{Q}_i$, as $\mathcal{P}$ should be connected to PTDs modelling the next edge to be traversed. However, if the MDP transition under consideration reaches the goal or a dead-end, the absorbing state should be added as the robot's route is finished. This is seen in lines 9-12. Function `addAbsorbingStatesAndTransitions` adds absorbing state $s^a_\mathcal{P}$ to $\mathcal{Q}_i$ and all transitions to it. If a goal state is reached in the MDP, $s^a_\mathcal{P}$ is labelled with $\{goal\}$.

To compute the initial state distribution for $\mathcal{Q}_i$, we consider the PTDs for all MDP transitions outgoing from $\bar{s}_{\mathcal{M}_i}$. For a state in one of these PTDs $\mathcal{P}$, its initial state probability in $\mathcal{Q}_i$ is its initial probability in $\mathcal{P}$ weighted by the probability of the associated MDP transition. This is shown in lines 14-16.

The final component of Algorithm 1 is to connect the PTDs to model the probabilistic outcomes of the MDP, and to sequence them to represent the robot's route. This is shown in lines 18-23. If a PTD $\mathcal{P}$ in the route CTMC models MDP transition (s, s'), $\mathcal{P}$ needs to be connected to the successor PTDs modelling the transitions outgoing from s'. For state $s_\mathcal{P}$ in $\mathcal{P}$, a transition from $s_\mathcal{P}$ to $s^a_\mathcal{P}$ is now split into multiple transitions. These go from $s_\mathcal{P}$ to the initial states of the successor PTDs. For a state $s_\mathcal{P}$ in $\mathcal{P}$, and $s_{\mathcal{P}'}$ in successor PTD $\mathcal{P}'$, the rate between the two states is the original rate from $s_\mathcal{P}$ to $s^a_\mathcal{P}$ in $\mathcal{P}$ multiplied by the initial probability of $s_{\mathcal{P}'}$ in $\mathcal{P}'$, weighted by the probability of the MDP transition that $\mathcal{P}'$ models.

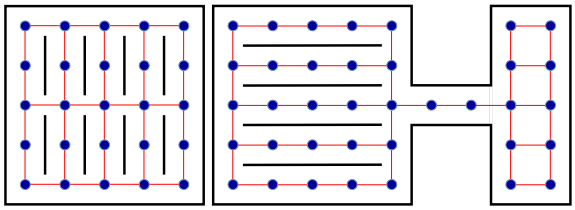
In Figure 4, we provide an example to demonstrate Algorithm 1. Figure 4a shows the DTMC induced by a policy on an MDP, annotated with actions and congestion bands. Figure 4b shows the PTDs, with absorbing states represented by concentric circles. Figure 4c shows the resulting route CTMC. Note how, in this example the λ_1 transition to the absorbing state in the PTD for $c^0_{e_1}$ is split into 4 transitions in the route CTMC. However, as the exit rate from that state is still λ_1, we still leave that state in the same amount of time.

7 EXPERIMENTS

In this section we analyse the scalability and execution-time performance of the presented planning framework. All experiments in this section are run on Ubuntu 16.04, with an Intel Core i7-8700 CPU@3.2GHz, and 16GB of RAM. All software is written in Python, except for PRISM [23], which is written in Java/C++.

Scalability. To test the scalability of the framework, we ran planning on three maps using synthetic data. The three maps were a 5×5 and 15×15 warehouse map (see Figure 5a, the 15×15 map is identical but scaled up), as well as a warehouse with a tunnel, designed to produce congestion (see Figure 5b).

For these experiments, the underlying duration distributions were created using lognormals, where for each successive number of robots, the mode of the distribution increases and the distribution



(a) A 5×5 warehouse. (b) Warehouse with tunnel.

Figure 5: The maps used for synthetic experiments.

has a higher variance, as we observe in practice. 1000 samples were taken from each, and PTDs were fit to these using the method outlined by Thummler et al. [36]. For each edge we defined four congestion bands, set to $[0, 0]$, $[1, 3]$, $[4, 5]$, $[6, n-1]$. All edges follow the same distributions, with minor variations made on each edge to ensure a representative state space. In the planning framework, the PRT pruning threshold is $\varepsilon = 10^{-4}$, the MDP time bound $T = 200$ and LRTDP is run until it converges or 100 trials are run. The scalability is measured in terms of total planning time, which is the time taken from the first robot starting to plan to the time at which the last robot finishes planning. This includes the time for CTMC insertion into the PRT, as well as the computation of congestion probabilities to define the transition functions of the MDPs.

To analyse the scalability of the framework, 40 random configurations of robots were generated for each of the 3 environments for 2-15 robot problems, by starting with 40 random 2 robot configurations for each map, and then adding one randomly configured robot to each to obtain the 3 robot configurations etc. The results of these experiments are displayed in Figure 6. The results show there to be a sub-exponential increase in the total planning time on all maps, and so this framework mitigates the exponential state space increase seen when using joint models. Larger maps give longer planning times, as the routes are typically longer, increasing the problem size. The variance in the results is due to the variance in congestion experienced across the problem configurations used. If there is less congestion, the branching factor of the MDPs will decrease as the variance over congestion is reduced. With a lower branching factor, LRTDP requires fewer trials to converge and fewer computations in the PRT will be required, reducing planning time. For example, though the 5×5 warehouse and warehouse with tunnel maps are of a similar size, the times for the latter are generally longer, as the tunnel causes congestion. Across all maps and problem configurations, approximately half of the planning time was spent in the PRT (i.e. computing congestion probabilities), with a median of 51%.

Execution Time Performance. To test the performance of the obtained policies at execution time, we simulate a 5 robot setup in ROS using the Stage simulator, using the map seen in Figure 1. All robots use the ROS `move_base` motion planner. Having two tunnels on the map allows robots to take the longer tunnel if the shorter one is too congested. PTDs were fit from data collected in simulation, by repeatedly sending robots to varying goal locations on the map. Every time a robot traversed an edge, the duration and number of robots on the edge was recorded. To ensure enough data was collected to be representative, we focused collection on a small subset of edges, and reused this data for edges of the same length. Figure 2 shows an example set of PTDs from this map. We use 5 congestion bands, one for each number of robots.

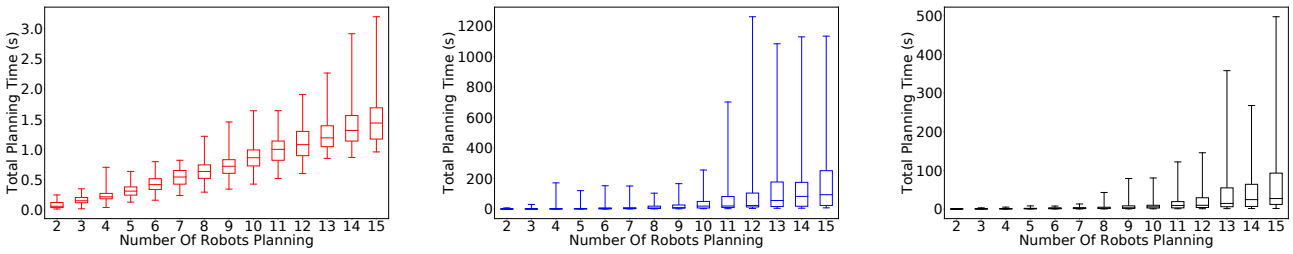
	0	1	2	3	4	5
Congestion-Aware	1.0	1.0	0.95	0.95	0.8	0.8
MAPF Baseline	1.0	0.95	0.95	1.0	0.95	1.0
Independent Baseline	1.0	1.0	0.95	0.95	0.8	0.25

Table 1: The success rate of each method as the number of robots sent to the other side of the map increases.

To judge the execution-time performance, we compare our framework against two baselines. The first baseline treats all robots independently, i.e. the robots ignore each other at planning time. When using this baseline all robots will follow their shortest path to the goal. This baseline relies only on the motion planner, as the interactions between robots are ignored at planning time. The second baseline acts similarly to our framework, but will only consider an edge at planning time if the probability of any congestion is less than a threshold ϵ, taken to be 0.1 in this experiment. If this is the case, the MDP states that the robot deterministically traverses the edge with no congestion. This baseline is similar to a MAPF approach as it almost entirely avoids interactions between robots at planning time. The threshold ϵ is chosen to be as low as possible while still allowing plans to be generated. For traditional MAPF solvers this threshold would be 0, but given the uncertainty in our models this would result in no plans being found. These baselines mirror the two ends of the multi-robot planning spectrum.

We evaluated 6 problem configurations on each of the 3 methods. For each problem, each method was run 20 times. Across all problems, 3 robots begin on the larger side of the map and 2 on the smaller side. Problem 0 sets the goal location of each robot on the same side of the map as their initial location. For problems 1-5, the problem number states how many robots have to travel to the other side of the map. If there are more robots switching sides of the map, there will be a higher level of congestion. A run is considered a failure if either the motion planner of one of the robots fails, or the robots do not reach their goals within 5 minutes. In either case this suggests robots are unable to navigate due to the presence of other robots. In this experiment we measure the *makespan*, i.e. the time until the last robot reaches its goal.

In Table 1, we show the success rates for the 3 methods over the 6 problem configurations. The low congestion problems see the robots always succeed as there is very little congestion in the environment. In the later problems, robots begin to fail as `move_base` is incapable of consistently dealing with multi-robot interactions when there is heavy congestion. Since the MAPF baseline is designed to avoid congestion it succeeds almost all of the time, with any failures being accountable to the small chance that heavy congestion can still occur with this baseline, as it is still a stochastic model. The independent framework fails frequently as congestion increases, as it plans for all robots to travel through the same tunnel. The congestion-aware framework shows a much smaller decrease in success rate, as it routes robots through separate tunnels, reducing the chance of creating scenarios `move_base` cannot handle.



(a) 5×5 **warehouse.** **(b)** 15×15 **warehouse.** **(c) Warehouse with tunnel.**

Figure 6: The scalability across the warehouse maps.

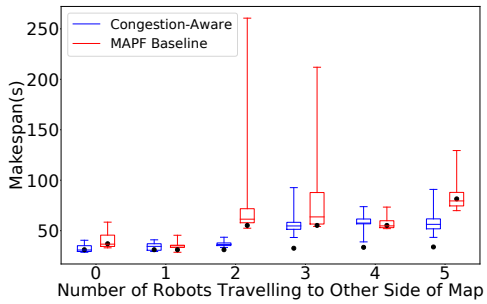


Figure 7: The makespans and expected makespans (circles) for the congestion-aware method and the MAPF baseline.

The success rates can be seen to describe how each of the 3 methods handle congestion. The independent baseline fails often as it does not handle congestion at all. The MAPF baseline almost never fails, as it does not exploit the fact that robots can navigate through certain levels of congestion. The success rate of the congestion-aware method shows how it will route robots through congested edges if appropriate. The failures caused by `move_base` are not represented in our planning models, though failures were recorded during data collection. Incorporating the failure of the continuous motion planner into our planning models is left to future work.

Figure 7 shows the makespans for the successful runs across all 6 problems for the congestion-aware method and MAPF baseline, as well as the expected makespans computed at planning time. For the low congestion problems, both methods route the robots through their shortest paths. When 2 and 5 robots are required to travel through the tunnels the congestion-aware framework outperforms the MAPF baseline, which sends robots through unnecessarily long routes to avoid congestion. When 3 and 4 robots travel through the tunnels, both methods perform similarly. This shows the congestion-aware framework can take longer routes if deemed most efficient. There is little change in the makespan for the congestion-aware framework between 3 and 5 robots travelling through the tunnel, displaying how this method can effectively distribute robots across the environment. In congested environments there is a significant difference between the expected and actual makespans for the congestion-aware framework. This shows the effect of `move_base` being unable to deal with multi-robot interactions. The congestion-aware method assumes a motion planner that can deal with such interactions and always succeed, whereas `move_base` often fails. Though the PTDs were obtained using `move_base`, the recorded durations of edge traversals do not take into account the frequent occurrence of `move_base` sending robots back to previous edges to avoid the congestion. These failures increase the actual makespan from the expected value obtained at planning time.

In summary, the congestion-aware framework outperforms both ends of the multi-robot planning spectrum, represented by the two baselines. It outperforms the independent baseline as the robots reach their goals successfully significantly more often with the congestion-aware framework. The congestion-aware framework outperforms the MAPF baseline as it plans such that the robots reach their goals in the same time or less than the baseline, by exploiting the motion planner's ability to handle congestion.

8 CONCLUSION

In this paper, we have presented a framework for multi-robot planning under uncertainty that allows robots to reason over continuous stochastic action durations and congestion. Though we have focused on robot navigation, this framework is applicable to general actions and shared resources. In this work, we have approximated a solution for uncertain action durations in a time-dependent MDP. This approach allows us to highlight the benefit of using the PRT in planning. In future work we will investigate more accurate planning approaches and include congestion models from the context of traffic assignment, such as link capacity functions [10], into our planning models. We will also integrate our framework with a motion planner better suited to handle congestion, and explore how to provide formal guarantees over the performance of the robots.

ACKNOWLEDGMENTS

Street is funded by the Honda Research Institute Europe GmbH. Lacerda is funded by UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) research hub [EP/R026084/1].

REFERENCES

[1] Christopher Amato, George D Konidaris, and Leslie P Kaelbling. 2014. Planning with macro-actions in decentralized POMDPs. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 1273–1280.

[2] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Stern. 2019. Multi-agent Pathfinding with Continuous Time. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 39–45.

[3] Andrea Bajcsy, Sylvia L Herbert, David Fridovich-Keil, Jaime F Fisac, Sampada Deglurkar, Anca D Dragan, and Claire J Tomlin. 2019. A Scalable Framework For Real-Time Multi-Robot, Multi-Human Collision Avoidance. In *Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 936–943.

[4] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. 2001. Optimizing schedules for prioritized path planning of multi-robot systems. In *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 271–276.

[5] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. 2002. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robotics and Autonomous Systems* 41, 2-3 (2002), 89–99.

[6] William Biscarri, Sihai Dave Zhao, and Robert J Brunner. 2018. A simple and fast method for computing the Poisson binomial distribution function. *Computational Statistics & Data Analysis* 122 (2018), 92–100.

[7] Blai Bonet and Hector Geffner. 2003. Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming. In *Proceedings of the Thirteenth Interanational Conference on Automated Planning and Scheduling (ICAPS)*. 12–21.

[8] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96)*. Morgan Kaufmann Publishers Inc., 195–210.

[9] Justin A Boyan and Michael L Littman. 2001. Exact solutions to Time-Dependent MDPs. In *Proccedings of the Advances in Neural Information Processing Systems (NIPS)*. 1026–1032.

[10] David Branston. 1976. Link capacity functions: A review. *Transportation Research* 10, 4 (1976), 223–236.

[11] Peter Buchholz, Jan Kriege, and Iryna Felko. 2014. *Input Modeling with Phase-Type Distributions and Markov Models: Theory and Applications*. Springer.

[12] Yuliya Butkova, Ralf Wimmer, and Holger Hermanns. 2017. Long-Run Rewards for Markov Automata. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*. Springer, 188–203.

[13] Daniel Claes, Frans Oliehoek, Hendrik Baier, and Karl Tuyls. 2017. Decentralised Online Planning for Multi-Robot Warehouse Commissioning. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 492–500.

[14] Daniel Claes, Philipp Robbel, Frans A Oliehoek, Karl Tuyls, Daniel Hennes, and Wiebe Van der Hoek. 2015. Effective Approximations for Multi-Robot Coordination in Spatially Distributed Tasks. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 881–890.

[15] Gautham Das, Grzegorz Cielniak, Pal From, and Marc Hanheide. 2018. Discrete Event Simulations for Scalability Analysis of Robotic In-Field Logistics in Agriculture–A Case Study. In *Proceedings of the IEEE International Conference on Robotics and Automation, Workshop on Robotic Vision and Action in Agriculture (WRVAA)*.

[16] Christian Eisentraut, Holger Hermanns, and Lijun Zhang. 2010. On Probabilistic Automata in Continuous Time. In *Proceedings of the 2010 25th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 342–351.

[17] Fatma Faruq, David Parker, Bruno Lacerda, and Nick Hawes. 2018. Simultaneous Task Allocation and Planning Under Uncertainty. In *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3559–3564.

[18] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *Proceedings of the Tenth Annual Symposium on Combinatorial Search (SoCS)*. 29–37.

[19] Robert W Floyd. 1962. Algorithm 97: Shortest Path. *Commun. ACM* 5, 6 (1962), 345.

[20] Jesper Karlsson, Cristian-Ioan Vasile, Jana Tumova, Sertac Karaman, and Daniela Rus. 2018. Multi-Vehicle Motion Planning for Social Optimal Mobility-on-Demand. In *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 7298–7305.

[21] Andrey Kolobov, Mausam, and Daniel S. Weld. 2012. A Theory of Goal-Oriented MDPs with Dead Ends. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI'12)*. 438–447.

[22] Marta Kwiatkowska, Gethin Norman, and David Parker. 2007. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*. Springer, 220–270.

[23] M. Kwiatkowska, G. Norman, and D. Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*. Springer, 585–591.

[24] Lantao Liu and Gaurav S Sukhatme. 2018. A Solution to Time-Varying Markov Decision Processes. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1631–1638.

[25] Hang Ma, TK Satish Kumar, and Sven Koenig. 2017. Multi-Agent Path Finding with Delay Probabilities. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*. 3605–3612.

[26] Masoumeh Mansouri, Bruno Lacerda, Nick Hawes, and Federico Pecora. 2019. Multi-Robot Planning Under Uncertain Travel Times and Safety Constraints. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*. 478–484.

[27] Mausam and Andrey Kolobov. 2012. *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers.

[28] João Vicente Messias, Matthijs Spaan, and Pedro Lima. 2013. GSMDPs for Multi-Robot Sequential Decision-Making. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*. 1408–1414.

[29] Hiroyuki Okamura, Ryo Watanabe, and Tadashi Dohi. 2014. Variational Bayes for Phase-Type Distribution. *Communications in Statistics – Simulation and Computation* 43, 8 (2014), 2031–2044.

[30] Mike Phillips and Maxim Likhachev. 2011. Sipp: Safe Interval Path Planning for Dynamic Environments. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5628–5635.

[31] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[32] Emmanuel Rachelson, Patrick Fabiani, and Frédérick Garcia. 2009. Timdppoly: An Improved Method for Solving Time-dependent MDPs. In *Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 796–799.

[33] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219 (2015), 40–66.

[34] Rongye Shi, Peter Steenkiste, and Manuela M Veloso. 2019. SC-M*: A Multi-Agent Path Planning Algorithm with Soft-Collision Constraint on Allocation of Common Resources. *Applied Sciences* 9, 19 (2019), 4037.

[35] David Silver. 2005. Cooperative Pathfinding. In *Proceedings of the first AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 117–122.

[36] Axel Thummler, Peter Buchholz, and Miklos Telek. 2006. A Novel Approach for Phase-Type Fitting with the EM Algorithm. *IEEE Transactions on Dependable and Secure Computing* 3, 3 (2006), 245–258.

[37] Jur P Van Den Berg and Mark H Overmars. 2005. Prioritized Motion Planning for Multiple Robots. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 430–435.

[38] Glenn Wagner and Howie Choset. 2015. Subdimensional expansion for multirobot path planning. *Artificial Intelligence* 219 (2015), 1–24.

[39] Glenn Wagner and Howie Choset. 2017. Path Planning for Multiple Agents Under Uncertainty. In *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS)*. 577–585.

[40] Shiqi Zhang, Yuqian Jiang, Guni Sharon, and Peter Stone. 2017. Multirobot symbolic planning under temporal uncertainty. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. International Foundation for Autonomous Agents and Multiagent Systems, 501–510.