Congestion-Aware Policy Synthesis for Multirobot Systems

Charlie Street, Sebastian Pütz, Manuel Mühlig, Nick Hawes, Bruno Lacerda

2021

Preprint:

This is an accepted article published in IEEE Transactions on Robotics. The final authenticated version is available online at: https://doi.org/10.1109/TRO.2021.3071618

Congestion-Aware Policy Synthesis for Multi-Robot Systems

Charlie Street, Sebastian Pütz, Manuel Mühlig, Nick Hawes, *Member, IEEE,* and Bruno Lacerda, *Member, IEEE*

***Abstract*—Multi-robot systems must be able to maintain performance when robots get delayed during execution. For mobile robots, one source of delays is *congestion*. Congestion occurs when robots deployed in shared physical spaces interact, as robots present in the same area simultaneously must manoeuvre to avoid each other. Congestion can adversely affect navigation performance, and increase the duration of navigation actions. In this paper, we present a multi-robot planning framework which utilises learnt probabilistic models of how congestion affects navigation duration. Central to our framework is a *probabilistic reservation table* which summarises robot plans, capturing the effects of congestion. To plan, we solve a sequence of single-robot *time-varying Markov automata*, where transition probabilities and rates are obtained from the probabilistic reservation table. We also present an iterative model refinement procedure for accurately predicting execution-time robot performance. We evaluate our framework with extensive experiments on synthetic data and simulated robot behaviour.**

***Index Terms*—Multi-robot systems; Planning under uncertainty; Temporal uncertainty; Formal verification**

I. Introduction

When planning for multi-robot systems under uncertainty, we wish to obtain *resilient* robot behaviour. To attain resilience, the system must be capable of handling unexpected delays during execution. Delays may stem from various sources, such as unknown obstacles or adverse weather conditions. Whilst it is not feasible to model all such sources *a priori*, in this work we focus on *congestion*.

Congestion occurs when multiple mobile robots are trying to operate in the same part of the environment at the same time. Congestion can occur in almost any environment a multi-robot system may be deployed, such as in warehouses [1], fruit fields [2] and roads [3]. To resolve congestion, robots must manoeuvre to avoid each other. These manoeuvres are often highly dependent on the precise spatial and temporal situation, and thus are hard to predict accurately at planning time. Therefore, congestion increases uncertainty over navigation performance.

Example 1. In Fig. 1, the presence of robots in the same aisle can cause congestion. If the aisle is wide enough and there are not many robots, they can manoeuvre around each other, with this deviation incurring a cost. However, if the aisle is too narrow, or many robots are present, some robots may have to turn back or wait. In the worst case, robots may become trapped by the other robots and be unable to move. Failure to consider this congestion when planning may lead to suboptimal multi-robot behaviour that diverges significantly from our expectations of the planned behaviour.

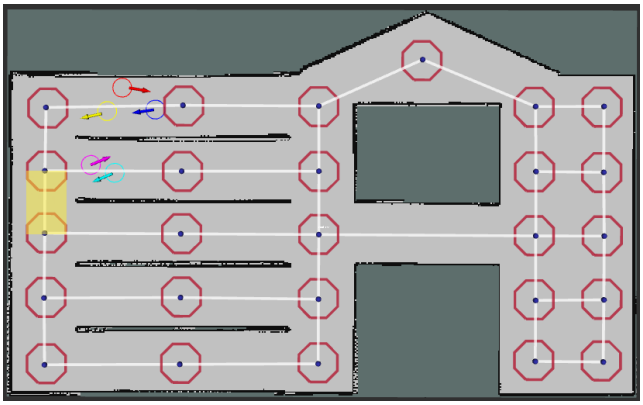


Fig. 1. A congested warehouse environment with a topological map overlaid. The circles with arrows show robots in the map. The yellow box marks the edge whose duration data and distributions are shown in Fig. 3.

An alternative to modelling the interactions between robots is to constrain the problem such that these interactions do not occur. For example, *multi-agent path finding* (MAPF) algorithms generate plans that prevent robots from being in the same location simultaneously [4]. However, since this may force robots to take conservative routes in order to stay away from each other, the resulting behaviour may be inefficient.

In this paper we present a novel framework for probabilistic planning that synthesises efficient robot behaviour by explicitly modelling congestion. We plan on a topological map, where navigation actions represent transitions between map nodes. Our framework reasons about the effects of congestion on navigation duration, allowing a robot to choose a longer route when shorter alternatives are likely to present heavy congestion; or select a slightly congested route that is likely to be quicker than longer, less congested alternatives. To plan for multi-robot behaviour under congestion, we model navigation durations as continuous stochastic processes using *phase-type distributions* (PTDs) [5]. Though we model congestion, we do not explicitly model how congestion is resolved, nor the underlying navigation uncertainty, e.g. from environmental sources. This information is implicitly represented as uncertainty over navigation duration, modelled as PTDs that are learned from observations. By including the learnt PTDs in our planning models we can

C.Street, N. Hawes and B. Lacerda are with the Oxford Robotics Institute, University of Oxford, UK, e-mail: {cstreet,nickh,bruno}@robots.ox.ac.uk.

S. Pütz is with the Knowledge Based Systems Group, Institute of Computer Science, Osnabrück University, Germany, e-mail: spuetz@uni-osnabrueck.de.

M. Mühlig is with the Honda Research Institute Europe GmbH, Offenbach, Germany, e-mail: manuel.muehlig@honda-ri.de.

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

synthesise multi-robot behaviour that is resilient to previously observed, but not explicitly modelled, sources of delay.

Given a policy describing the navigation route planned for each robot, we use the corresponding PTDs to construct a *continuous-time Markov chain* (CTMC), which allows us to compute the probabilities of encountering different numbers of robots (which we refer to as congestion levels) at different times across the map edges in the route. The PTDs, CTMCs and congestion information are managed in a structure called a *probabilistic reservation table* (PRT). For planning we use the information from the PRT to construct single-robot *time-varying Markov automata* (TVMA), a generalisation of a *Markov decision process* (MDP) that explicitly models uncertainty over action duration. Crucially, a TVMA allows the planning robot to consider the influence of the other robots on its route by encoding their presence as *congestion probabilities*. The policies we synthesise are resilient to unexpected delays as a robot's arrival time to a node is included in the policy's state space. Therefore, if a robot arrives late to a location, its policy action may change.

As well as synthesising robot behaviour, we also wish to accurately predict the performance of the multi-robot system at execution time. For example, we may wish to know the probability that a robot will arrive at its goal within a given time. This can then be used to aid execution monitoring [6], or to optimise the number of robots in a team [7]. To predict performance, we require accurate models of robot policy execution under congestion. We present an iterative procedure that refines single-robot CTMCs stored in the PRT to accurately capture the effects of the other robots. We then predict robot performance by using formal verification techniques to obtain probabilistic guarantees on the refined CTMCs.

This paper extends the work presented in [8]. We make the following contributions beyond our prior work:

- The formulation of continuous-time planning problems under time-varying dynamics as a time-varying Markov automaton (TVMA).
- An approximate solution to TVMAs which incrementally builds and solves an MDP abstraction of the TVMA.
- An application of the TVMA solution method to synthesise efficient robot behaviour for congestion-aware problems.
- An iterative algorithm that refines robot CTMCs to more accurately represent the influence of other robots.
- An approach to compute probabilistic guarantees over these CTMCs to predict execution-time robot performance.

These contributions are validated with an extensive set of experiments on synthetic data and in a robotic simulator.

II. Related Work

Sources of uncertainty can be found in almost any environment robots are deployed in. Therefore, it is important to incorporate environmental uncertainty into robot planning models. MDPs are a common formalism for this. An MDP models a system with non-deterministic action choices and stochastic action outcomes [9]. For mobile robot navigation planning, an MDP state typically includes the robot's location, and actions represent choices of (nearby) locations to attempt to move to.

In this work, we plan for *multiple* mobile robots under uncertainty. A multi-agent MDP (MMDP) is a natural extension of an MDP for multiple agents, where the agents act in a *joint* state and action space [10]. In a joint model 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 all robots having to wait for each other to finish acting before a subsequent action can be performed [11]. In this work we consider actions with continuous stochastic durations, and robots act asynchronously.

To avoid the exponential scalability of a joint model, it is possible to plan using single-robot models that are extended to include some knowledge of the other robots. In [1], [12] single-robot models aggregate the responses of the other robots assuming they are only aware of themselves. In [13] robots also plan on single-robot models using a procedure where robots consider their team mates more with every iteration. However, action execution is assumed to be deterministic, with the output being a sequence of actions, which are less robust to the probabilistic nature of the environment. In our work, we approximate multi-robot behaviour by creating single-robot models which include the effects of congestion. We generate policies for each robot *sequentially*, with each robot considering only those that planned before it. Such an approximation is common in multi-robot planning [14]–[16]. The quality of plans produced under a sequential planning assumption is sensitive to the ordering of robots [16], and determining an optimal ordering is NP-hard [17]. However, effective priority orderings have been generated using heuristics [18], [19], as well as optimisation methods [20].

Asynchronous execution can be enabled through the use of macro actions [21], which are high level behaviours represented with the options framework [22]. Macro actions are executed asynchronously, with synchronisation occurring at the level of the controllers that form the macro actions. Alternatively, action durations can be represented as continuous stochastic processes, with shifted Poisson distributions [13], exponential distributions [23]–[25], or arbitrary temporal distributions [11]. In our work, we use continuous stochastic processes to represent action durations as well as the effects of congestion on these durations.

Planning under continuous stochastic action durations requires a continuous-time planning model. In semi-MDPs [22], each state/action pair has an arbitrary duration distribution. Generalised semi-MDPs extend this to enable concurrent events [11], [26]. Continuous-time MDPs (CTMDPs) also allow concurrent events but restrict action durations to exponential distributions [27], and have been used to synthesise single-robot policies [28]. Markov automata generalise CTMDPs with immediate and exponentially timed transitions [23], and have been used for multi-robot planning [24], [25].

Robot environments are commonly assumed to be static. This does not hold for multi-robot systems, as the state of the robots at a given time affects the duration and outcome of actions. There exist Markov models capable of capturing such non-stationary transition dynamics, such as *time-dependent* MDPs [29]. However, approaches to solve these models either scale badly [30] or rely on strong simplifications [31], [32], such as deterministic models of action duration. Such simplifications are not applicable to our work, as we aim to explicitly model and reason about the uncertainty over action duration.

Recent work [33] has begun to address these issues. There, an approximate solution to time-varying semi-MDPs is presented, where non-stationary transition dynamics are learned online and planning is carried out by extending Monte Carlo tree search to consider non-stationarity and stochastic transition durations. This was done under the assumption of non-concurrent actions, which does not hold in a multi-robot scenario, where robots execute actions at the same time.

MAPF solvers are widely used for the scalable coordination of multiple robots, generating collision-free paths for robots to reach their goals in discretised environments [4]. In [16] a *reservation table* is used to store the route information of the robots, such that robots avoid those who have planned previously. In this paper, we expand the reservation table to allow actions with continuous stochastic durations. Most MAPF solvers assume that navigation actions have deterministic, discrete durations, and require synchronised execution. A few works have considered continuous-time MAPF. For example, in [34], the conflict-based search (CBS) algorithm [35] is adapted by replacing A* search for low-level path planning with safe interval path planning [36]. In [37], CBS is reformulated in terms of the satisfiability modulo theories problem to enable the use of efficient SAT solvers. Contrary to our work however, both [34] and [37] assume that the continuous action durations are not stochastic.

Though MAPF solutions commonly assume deterministic environments, uncertainty has been considered. In [38], a belief space is used, giving distributions that appear as 'tails' over the robot's location. The M* MAPF solver [39] is then adapted to plan in the belief space of each agent, with coordination occurring between robots likely to collide. In [40], when a robot attempts to navigate, it remains stationary with some probability. To handle action failure, a set of critical dependencies between robots are computed, which force some robots to wait until another robot has reached a certain location. Finally, in [41], the M* solver is adapted to have soft collision constraints, i.e. collisions are allowed to occur, at some cost. Whilst these works consider uncertainty, they do so in ad-hoc manners that consider limited forms of duration uncertainty. In contrast, we propose a principled way of reasoning about and evaluating timed properties of robot behaviour, adapting techniques from formal methods and model checking. We use continuous-time Markov chains (CTMCs) to model the timed evolution of a system, where time passes according to a sequence of exponential delays [42]. Timed properties of CTMCs, such as transient and steady-state properties, can be expressed using continuous stochastic logic (CSL) [43], with well established model checking solutions available [44]. Few works have considered model checking timed properties of multi-robot systems, and those that do consider deterministic action durations [45]–[47].

III. Preliminaries

For a set X, $|X|$ denotes the cardinality of X and $Dist(X)$ denotes the set of distributions over X.

Probability Distributions. We approximate continuous distributions of action durations with *discrete random variables*.

Definition 3.1. A *discrete random variable* D may take on values in the set $\{d_1, d_2, ..., d_K\}$. The probability that D takes on the value d_k is given by $Pr(D = d_k) = p_k$, where $p_k \in [0, 1]$ and $\sum_{k\in\{1,...,K\}} p_k = 1$.

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

Definition 3.2. 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 directional 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.

For edge $e = (v, v') \in E$, we use $\texttt{src}(e) = v$ and $\texttt{trg}(e) = v'$ to denote the source and target of that edge, respectively. Moreover, we partition the topological edges into a set of *edge groups*, which allow us to represent spatial dependencies between edges.

Definition 3.3. A set of edge groups $G = \{g_1, ..., g_{|G|}\}$ is a partition of the edge set E, i.e $g \subseteq E$ for all $g \in G$, $g \cap g' = \emptyset$ for all $g, g' \in G$ such that $g \neq g'$, and $\bigcup_{i\in\{1,...,|G|\}} g_i = E$.

With a slight abuse of notation, we denote the edge group for e as $g(e)$. We assume edges in the same edge group g are such that robots present on an edge $e \in g$ influence the duration of traversing all other edges $e' \in g$. Thus, when counting robots on e we also consider all other edges in $g(e)$. Furthermore, for $g \in G$ and $n \in \mathbb{N}$, we assume $\rho(e, n) = \rho(e', n)$ for all $e, e' \in g$, i.e. we assume that edges in the same edge group share the same duration distributions.

In this work, for each edge $(v, v') \in E$, we assume there exists an edge in the opposite direction $(v', v) \in E$. These edges are dependent, since they occupy the same space. We define the edge groups such that for any edge (v, v'), $g((v, v')) = \{(v, v'), (v', v)\}$. Therefore, we do not distinguish between the direction a robot travels between two nodes. Though other duration models can be used, for simplicity of presentation we use this bidirectional model. Other use cases for edge groups include junctions and narrow corridors with multiple edges.

Time-Varying Markov Automata (TVMA). In this paper, we address *stochastic shortest path* problems over labelled TVMAs, denoted TVMA-SSPs. A TVMA is an extension of a Markov automata [23], which allows us to model immediate probabilistic transitions and exponentially timed transitions with time-varying dynamics [31], [32].

Definition 3.4. A *labelled TVMA* is a tuple $\mathcal{A} = \langle S, \bar{s}, A, \delta, \Delta, AP, Lab\rangle$, where S is a finite set of states, $\bar{s}$ is the initial state, and A is a finite set of actions. $\delta : S \times A \times S \times \mathbb{R}_{\geq 0} \to [0, 1]$ is a time-varying immediate transition function, such that $\delta(s, a, s', t)$ returns the probability of instantaneously transitioning to state s' after action a was taken in state s at time t. $\Delta : S \times S \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ is a time-varying exponential transition function, where $\Delta(s, s', t)$ returns the *rate* between states s and s', given that we arrived

at state s at time t. 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.

In a TVMA, action choices are made at states with immediate transitions. Without loss of generality, we assume no state has both immediate and exponential transitions: immediate transitions fire instantaneously, and so exponential transitions will never fire in states with both transitions.

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

A *time invariant* transition function does not vary with time, e.g. for states s, s', $\forall t, t'$, $\Delta(s, s', t) = \Delta(s, s', t')$. We omit argument t if a transition function is time invariant.

Definition 3.5. A *TVMA-SSP* is an extension of an SSP [48], defined as a TVMA $\mathcal{A} = \langle S, \bar{s}, A, \delta, \Delta, AP, Lab\rangle$, a time-varying cost structure $c : S \times A \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{>0}$ and a set of goal states $G \subseteq S$. The goal of a TVMA-SSP is to find a time-varying policy $\pi : S \times \mathbb{R}_{\geq 0} \to A$ that minimises the expected cost to reach a state in G.

Policies define behaviour by informing the robot which action to take in a given state at a given time, i.e. $\pi(s, t)$ represents the action the robot should take given that it arrived at state s at time t. As in SSPs over MDPs [49], in a TVMA-SSP we require the existence of at least one proper policy, i.e. a policy that, starting in $\bar{s}$, reaches a goal state with probability 1; and that for any improper policy, the value of all states under that policy is ∞. Further, all goal states must be *absorbing*, i.e. there must be no transitions from goal states.

Markov Decision Processes (MDPs). In this paper, we use MDPs to solve an approximated version of the congestion-aware multi-robot planning problem.

Definition 3.6. An *MDP* [9] is a TVMA without exponential transitions and with a time invariant transition function, i.e. an MDP is a tuple $\mathcal{M} = \langle S, \bar{s}, A, \delta\rangle$, where all elements are as in a TVMA, with δ being time invariant.

A time invariant policy $\pi : S \to A$ over an MDP $\mathcal{M}$ induces a *discrete-time Markov chain* (DTMC) $\mathcal{M}^\pi$, where, at each state, the only action available is the policy action.

Continuous-Time Markov Chains (CTMCs). A CTMC describes the continuous-time evolution of a system. In this work, we use labelled CTMCs to model the continuous-time execution of robot policies.

Definition 3.7. A *labelled CTMC* [42] is a tuple $\mathcal{Q} = \langle S, init, \Delta, AP, Lab\rangle$, where S, Δ, AP and Lab are as in a TVMA, except Δ is time invariant, and $init : S \to [0, 1]$ gives the probability of a state being the initial state.

Phase-Type Distributions (PTDs). PTDs approximate non-negative continuous distributions using the time taken to reach an absorbing state in a CTMC. We use PTDs to model action durations under congestion.

Definition 3.8. A *PTD* [50] is a tuple $\mathcal{P} = \langle S, init, \Delta, s^a\rangle$, where S, $init$ and Δ are as in a CTMC, and $s^a \in S$ is the single absorbing state such that the probability of reaching s^a is 1. Further, $init(s^a) = 0$.

We denote the expected time for $\mathcal{P}$ to reach the absorbing state as $\mathbb{E}[\mathcal{P}]$. Further, the set of all PTDs is denoted $\mathbb{P}$.

To avoid ambiguity, we use a subscript to denote 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 $\mathcal{M}$.

IV. Problem Formulation

In this section, we formulate the problems of planning (i.e. policy synthesis, Problem 1) and policy evaluation (Problem 2) under congestion. We start by defining local timed policies over a topological map, the space of policies our policy synthesis algorithm searches over.

Definition 4.1. A navigation policy for a robot r_i acting on a topological map $\mathcal{T} = \langle V, E, \rho\rangle$ is defined as $\pi_i : V \times \mathbb{R}_{\geq 0} \to E \cup \{wait\}$. $\pi_i(v, t)$ represents the action a robot should take, given it is at location v at time t. If $\pi_i(v, t) = e \in E$, then edge e should be traversed. If $\pi_i(v, t) = wait$ then robot r_i should wait at node v.

Note that the definition above only assumes *local* information for robot r_i, namely its location and the current time. This means the policy can be executed without communicating with other robots, making our proposed solution resilient to communication failures. We can now define the multi-robot policy synthesis problem we tackle in this paper.

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 $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$, where π_i is a policy for robot r_i, such that Π minimises the *makespan* of the multi-robot navigation problem, i.e. Π minimises the time until all robots have reached their goals.

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

We also consider the problem of accurately model checking properties of navigation policies:

Problem 2. Let π_i be a navigation policy for robot r_i, and φ_i a local property for r_i. Evaluate φ_i against the continuous-time behaviour induced by π_i under the effects of congestion.

We motivate Problem 2 as a means to predict the execution-time performance of the synthesised navigation policies. In this paper, we evaluate the expected time for a robot to arrive at its goal, and the probability that a robot arrives at its goal within time τ. However, the method we propose can be used for any property that can be model-checked against a CTMC, e.g. continuous stochastic logic (CSL) or linear temporal logic (LTL) properties.

V. Framework

In this section we outline our approach to Problems 1 and 2. A high-level overview of the *congestion-aware framework* is given in Fig. 2. There is one global entity, the probabilistic reservation table (PRT). The PRT keeps track of robot policies, and represents robot behaviour as a set of *congestion distributions* over the topological map (see Section VI-C). To manage complexity, each robot plans to minimise the expected time to reach its goal on a single-robot TVMA which approximates the effects of the other robots. To consider the other robots, we assume robots plan sequentially, following a pre-defined priority order. For clarity of notation, a robot's priority is defined by its index, i.e. robot r_1 plans first, then r_2 etc. To minimise the makespan, we define the priority order using the heuristic in [18], such that robots with longer routes have a higher priority. Robot r_i begins by querying the PRT to obtain congestion probabilities summarising previous robot behaviour (see Section VI-C). These probabilities are used to construct a TVMA $\mathcal{A}_i$, from which we synthesise a policy π_i (see Section VII-A). From π_i, we construct a *route CTMC* $\mathcal{Q}_i$ and insert it into the PRT (see Section VII-C). The route CTMC captures the continuous-time behaviour induced by π_i, and is used to compute future congestion probabilities. Therefore, robot r_{i+1} uses the PRT to reason over the probabilistic effects of robots $r_1, ..., r_i$ in its TVMA. The PRT is initially empty. Therefore, the first robot who plans has no information about the other robots and so acts optimistically, as it believes it can follow the shortest path to the goal without interference. The final robot to plan takes into account all robots. Therefore, robots who plan earlier have less accurate planning models.

To accurately analyse properties of the synthesised policies after planning (Problem 2), we need a model that describes the asynchronous execution of the robot team. To do so, one could construct a joint CTMC that describes the asynchronous execution of the full robot team. However, building and analysing a joint model is intractable for most realistic problems. Therefore, we use the single-robot route CTMCs stored in the PRT to approximate the joint model. Before analysis, we iteratively refine these CTMCs to take into account updated models of the other robots, particularly those that planned later (see Section VIII). Refining a robot's CTMC does not modify the corresponding policy. The refined CTMC captures the continuous-time execution of the policy under the probabilistic effect of all other robots' policies. This procedure is independent of the property being evaluated. Any single-robot property, such as expected time for task completion or probability of reaching the goal within a time bound, can then be evaluated on the refined CTMCs using formal verification techniques. The unrefined CTMCs can be evaluated in the same way, however these models are less accurate.

We proceed by presenting our probabilistic approach to modelling congestion in Section VI. In Section VII we describe how we plan for each robot using the congestion models, and in Section VIII we present the iterative CTMC refinement procedure.

VI. Modelling Congestion

To synthesise effective robot behaviour, we must capture the effects of congestion between robots. In this section, we introduce the probabilistic reservation table (PRT) as a model of congestion that considers actions with continuous and stochastic durations.

A. 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. We use PTDs to approximate these duration distributions, because they allow for the interpretation of robot policies as CTMCs (see Section VI-C), and to exploit efficient fitting algorithms [51].

Example 2. Consider Fig. 3, which shows congestion data observed on the highlighted edge in Fig. 1, as well as PTDs fitted to that data. This data demonstrates that the navigation duration for a given congestion level is stochastic, i.e. the number of robots on an edge are not enough to deterministically predict the time taken for a robot to traverse the edge. This is due to other unmodelled disturbances that have an effect on the resolution of congestion, such as the relative locations of robots on an edge. This motivates our use of probabilistic duration models to increase the system's resilience.

As well as navigating along edges, robots may wait at a node until congestion is reduced. Due to the stochastic nature of navigation, waiting is also treated as stochastic, using a waiting PTD $\mathcal{P}_\omega$, which we choose to be a two-state PTD with a single transition from the first to the second state with rate λ_ω.

B. Congestion Bands

We describe levels of congestion in terms of *congestion bands*. We use congestion bands because there are cases where we do not observe a statistical difference in the effects of congestion between similar numbers of robots (cf. 3 and 4 robots in Fig. 3). A larger number of congestion bands gives a more accurate measure of congestion while increasing the complexity of planning. Congestion bands help when we lack data, as data for a range of numbers of robots is aggregated to fit a model.

Definition 6.1. 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 integer interval on the number of robots, represented by a lower and upper bound. If $k \in \mathbb{N}$ robots are present on e, where $lb_e^j \leq k \leq ub_e^j$ then c_e^j is the congestion band 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 number of robots fits into exactly one band.

We assume that edges in the same edge group share congestion bands, i.e. given $g \in G$, $C_e = C_{e'}$ for all $e, e' \in g$. A robot is considered present on an edge $e = (v, v')$ if they are traversing it, i.e. they are travelling between v and v'. Using

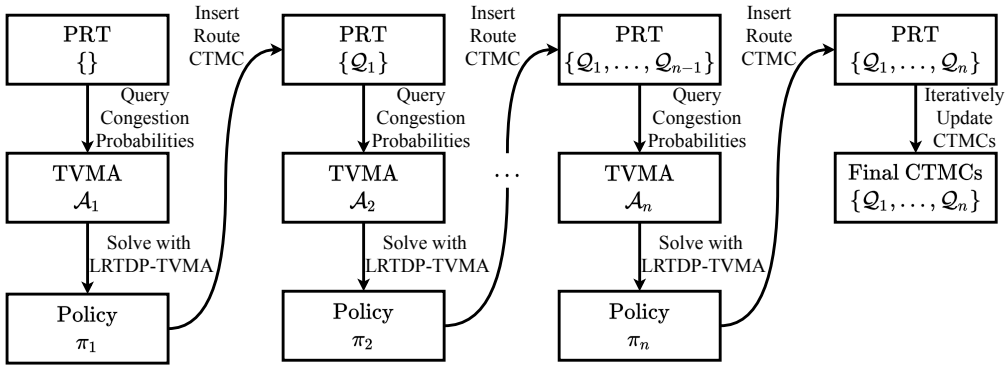
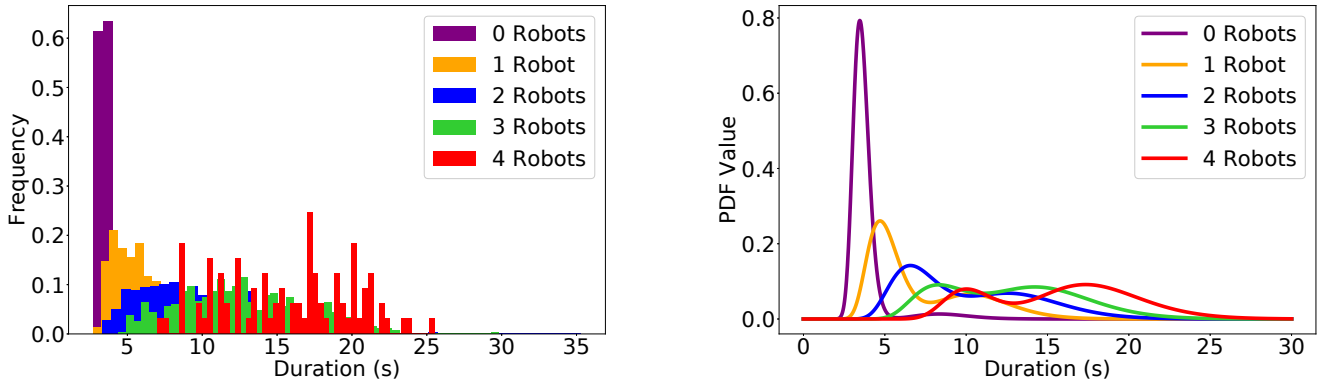


Fig. 2. An overview of the congestion-aware framework. Robots plan sequentially. Each robot r_i queries the PRT to build its TVMA $\mathcal{A}_i$, which is solved to obtain a policy π_i. A route CTMC $\mathcal{Q}_i$ representing this policy is inserted into the PRT. After planning, all CTMCs can be updated iteratively to estimate execution-time robot performance, if required.



(a) (b)

Fig. 3. The effects of congestion on navigation duration for the highlighted edge in Fig. 1. Data was collected using the method outlined in Section IX. (a) Histograms of the collected data. (b) PTDs fitted from the data in Fig. 3a using the method in [51].

the notion of congestion bands, we introduce a topological map under congestion.

Definition 6.2. 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.

If we consider one congestion band per possible number of robots, we recover Definition 3.2, but using PTDs to represent the duration distributions. The connectivity of $\mathcal{T}_C$ defines how robots move through the environment. For waiting we add a self loop, referred to as a *wait edge*, around each node in $\mathcal{T}_C$, i.e. $\forall v \in V,\ (v, v) \in E$. Wait edges have a single wait PTD, as congestion can not be experienced when waiting, i.e. $\rho_C((v, v), c^0_{(v,v)}) = \mathcal{P}_\omega$, for all $v \in V$.

To simplify notation, we omit subscript e from congestion band c^j_e if e can be inferred from the context.

C. Computing Congestion Probabilities

To determine the minimum cost policy during planning, we require the congestion band observed on edge e at time t. Fig. 3 demonstrates that action durations are stochastic. When propagated over sequences of actions, this stochasticity prevents us from knowing where a robot will be at a given time. Thus, we consider the *probability* of each congestion band occurring on edge e at time t. The process for computing congestion probabilities is shown in Fig. 4.

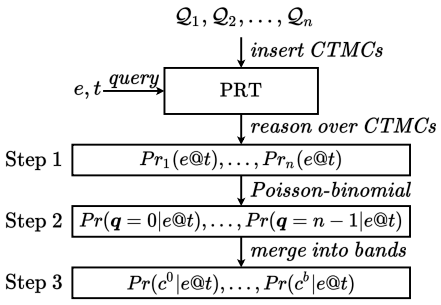


Fig. 4. The pipeline for computing the congestion probabilities on edge e at time t using the PRT. Note, there may not be a CTMC for all robots at the time of use.

To compute the congestion probabilities, we require information about the robots' routes. We use a CTMC $\mathcal{Q}_i$ to represent the route of robot r_i. CTMCs are a natural representation of continuous-time robot behaviour, since we can concatenate the PTDs modelling navigation durations to construct a route CTMC. Further, the transient properties of this CTMC can

be used to compute the congestion probabilities [42]. In this section, we define the assumptions a route CTMC must satisfy, and in Section VII describe how to construct such a model.

Definition 6.3. Let $\mathcal{T}_C$ be a topological map under congestion. A *route CTMC* is a tuple $\mathcal{Q} = \langle S_\mathcal{Q}, init_\mathcal{Q}, \Delta_\mathcal{Q}, AP_\mathcal{Q}, Lab_\mathcal{Q} \rangle$, where $AP_\mathcal{Q} = E \cup \{goal\}$, i.e. the set of atomic propositions is comprised of the edges of $\mathcal{T}_C$ and a $goal$ proposition, and $Lab_\mathcal{Q}$ is defined such that all non-absorbing states in $\mathcal{Q}$ are labelled with a single topological edge, and absorbing states are labelled with $goal$, i.e. for all $s \in S_\mathcal{Q}$, $|Lab_\mathcal{Q}(s)| = 1$ and $Lab_\mathcal{Q}(s) = \{goal\}$ if and only if s is absorbing.

We require the CTMC labelling function to represent the robot's route through the topological map. States labelled with edge $e \in E$ in a route CTMC $\mathcal{Q}_i$ represent states for which r_i is present on e. The route CTMC states 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 modelled by the PTD it comes from.

An absorbing state of the route CTMC represents the completion of a robot's route, and is labelled with proposition $goal$. The $goal$ proposition is useful for analysing policies, e.g. computing the expected time for a robot to arrive at its goal, as required by Problem 2. Policy analysis is discussed further in Section VIII.

When a robot plans, it builds a route CTMC for its policy and stores it in the PRT. As depicted in Fig. 4, the first step to compute the congestion probabilities is to consider the probability that each robot r_i is present on an edge in $g(e)$ at time t, denoted $Pr_i(e@t)$. Recall that we must consider all edges in $g(e)$ as robots present on any edge in $g(e)$ influence the congestion on e.

To compute $Pr_i(e@t)$, we compute the *transient probabilities* over the route CTMC $\mathcal{Q}_i$, which returns the probability of being in each state $s \in S_{\mathcal{Q}_i}$ at time t, denoted $Pr(s@t)$. From the transient probabilities, we can calculate the probability $Pr_i(e@t)$ for robot r_i by summing across all states labelled with e:

$$Pr_i(e@t) = \sum_{\{s \in S_{\mathcal{Q}_i} \mid g(e) \cap Lab_{\mathcal{Q}_i}(s) \neq \emptyset\}} Pr(s@t) \quad (1)$$

In practice, the transient probabilities of a CTMC can be calculated using standard techniques [42].

We compute the probability a robot is present on an edge using only its route CTMC. This assumes all necessary congestion information for r_i is contained in $\mathcal{Q}_i$. However, the true congestion probabilities change as new route CTMCs are added to the PRT. Under the sequential planning assumption, we treat a robot's route as fixed at the point of insertion into the PRT. In Section IX, we show empirically how this assumption still allows us to effectively reason over congestion. Further, in Section VIII we address how the true congestion probabilities may be approximated using an iterative procedure that incrementally updates the PRT.

Step 2 in computing the congestion probabilities for a new robot is to use the probabilities $Pr_i(e@t)$ to compute a discrete distribution over how many *other* robots are on an edge in $g(e)$ at time t, i.e. how congested the edge will be. For this we use the Poisson-binomial distribution, which describes the outcome of n trials, each distributed by a different Bernoulli random variable [52]. The Bernoulli random variables model the presence of each robot on edges in $g(e)$ at time t, distributed by $Pr_i(e@t)$. We denote the probability of q other robots being on an edge in $g(e)$ at time t by $Pr(q \mid e@t)$.

In step 3 we group the probabilities for each number of robots on an edge in $g(e)$ at time t into congestion bands. The probability $Pr(c^j \mid e@t)$ of experiencing congestion band c^j on edge e at time t, considering all edges that influence the congestion on e, is given by:

$$Pr(c^j \mid e@t) = \sum_{q=lb_e^j}^{ub_e^j} Pr(q \mid e@t) \quad (2)$$

D. Defining a PRT

A PRT is initialised with the set of robots R and the topological map $\mathcal{T}_C$. A PRT is a table with an entry for each robot $r_i \in R$. The PRT entry for robot r_i, PRT(r_i), stores its route CTMC $\mathcal{Q}_i$. Two functions are provided by the PRT: $update$ and $cong$. The function $update(r_i, \mathcal{Q}_i)$ updates PRT(r_i) by inserting route CTMC $\mathcal{Q}_i$. The function $cong(r_i, e, j, t)$ returns the probability of r_i observing congestion band c^j on edge e at time t, computed using the method in Fig. 4.

E. Using the PRT Online

In this paper, we focus on offline planning problems. However, robots often have to react to tasks arriving *online*, e.g. when packing warehouse orders. This can be achieved with two PRT modifications. The PRT can be accessed by only one robot at a time. For offline problems, this is handled by the priority ordering. Online, the priority ordering must be replaced with an access rule, such as first come first served, for robots to query the PRT and insert route CTMCs. Moreover, robots will start tasks at different times. To handle this, the PRT must log the start time of each robot's route. When computing congestion probabilities, the probability of a robot being on an edge before their start time is zero.

VII. Planning Under Congestion

As described in Problem 1, we wish to synthesise a set of policies that takes the robots to their respective goals and minimises the makespan. In this section, we outline how to plan in a scalable way using single-robot models, where the effects of congestion are used to summarise the team behaviour. We also detail how to construct route CTMCs that describe a robot's continuous-time behaviour.

A. Single-Robot Planning Models

The congestion probabilities for an edge vary with time. Further, action durations are represented as PTDs, which are CTMCs. Therefore, we formulate the planning model for robot r_i as a TVMA. Let $S_\mathbb{P} = \bigcup_{\mathcal{P} \in \mathbb{P}} S_\mathcal{P}$ be the union of all PTD state spaces, assuming they are disjoint. For $s \in S_\mathbb{P}$, we write $\texttt{PTD}(s)$, $e(s)$ and $b(s)$ to denote the PTD, edge, and congestion

band a state s belongs to respectively. For notational simplicity, for $s \in S_{\mathbb{P}}$, let $init_s = init_{\mathtt{PTD}(s)}(s)$ be the initial probability of state s in its respective PTD.

Definition 7.1. Given topological map $\mathcal{T}_C = \langle V, E, C, \rho_C \rangle$ and the PRT, we define robot r_i's TVMA as $\mathcal{A}_i = \langle S_{\mathcal{A}_i}, \bar{s}_{\mathcal{A}_i}, A_{\mathcal{A}_i}, \delta_{\mathcal{A}_i}, \Delta_{\mathcal{A}_i}, AP_{\mathcal{A}_i}, Lab_{\mathcal{A}_i} \rangle$, where:

- $S_{\mathcal{A}_i} = V \cup S_{\mathbb{P}}$. A state is a topological node or a PTD state.
- $\bar{s}_{\mathcal{A}_i} = v_i^{init}$. The initial state is the robot's initial location.
- $A_{\mathcal{A}_i} = E \cup \{wait\}$. The actions correspond to navigating the topological map edges, plus waiting.
- $\delta_{\mathcal{A}_i}$ models immediate transitions corresponding to action choices that capture uncertainty over congestion. From a topological node state s, we choose an outgoing edge e. For each congestion band c^j on edge e there is a transition from s to the initial states s' of PTD $\rho_C(e, c^j)$. The transition probability is the congestion probability for c_e^j multiplied with the initial probability of s' in $\rho_C(e, c^j)$. Formally:

$$\delta_{\mathcal{A}_i}(s, e, s', t) = \begin{cases} cong(r_i, e, j, t) \cdot init_{s'} & s \in V, \\ & s = \mathtt{src}(e), \\ & c^j = b(s') \text{ and} \\ & s' \in S_{\rho_C(e, c^j)} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

- $\Delta_{\mathcal{A}_i}$ captures the effects of the PTDs. For a state s in PTD $\mathcal{P}$, the rate to a non-absorbing state s' in $\mathcal{P}$ is defined according to $\Delta_{\mathcal{P}}$. Transitions to the absorbing state in $\mathcal{P}$ now arrive at the target of the edge the robot was traversing. Formally:

$$\Delta_{\mathcal{A}_i}(s, s', t) = \begin{cases} \Delta_{\mathtt{PTD}(s)}(s, s') & s, s' \in S_{\mathtt{PTD}(s)} \text{ and} \\ & s' \neq s^a_{\mathtt{PTD}(s)} \\ \Delta_{\mathtt{PTD}(s)}(s, s^a_{\mathtt{PTD}(s)}) & s \in S_{\mathbb{P}}, \\ & s' \in V \text{ and} \\ & s' = \mathtt{trg}(e(s)) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

- $AP_{\mathcal{A}_i} = E \cup \{wait, goal\}$. The atomic propositions represent the action currently being executed, or the robot reaching its goal.
- $Lab_{\mathcal{A}_i}$ labels PTD states with the edge they model (or waiting), and topological nodes at the goal with $\{goal\}$. Formally:

$$Lab_{\mathcal{A}_i}(s) = \begin{cases} \{wait\} & s \in S_{\mathcal{P}_\omega} \\ \{e(s)\} & s \in S_{\mathbb{P}} \setminus S_{\mathcal{P}_\omega} \\ \{goal\} & s \in V \text{ and } s = v_i^{goal} \\ \emptyset & \text{otherwise} \end{cases} \tag{5}$$

In the TVMA, the robot arrives at a location and chooses the next edge. There is a time-varying congestion distribution on this edge. For each band, there is then a sequence of exponential transitions that describe the respective PTD. All PTDs finish at the location the robot arrives at.

Existing solution methods for Markov automata are computationally expensive, due to the combination of immediate and exponential transitions. For timed objectives, exponential transitions are often 'digitised' to be of the same duration, which makes solution methods slow to converge [53]. To improve scalability, we pose the planning problem for r_i as a TVMA-SSP, where the objective is to minimise the expected time to reach the goal. As such, the TVMA-SSP cost function considers only the expected duration of actions; this mitigates the complexity of digitisation based methods. In Section VIII, we model-check the synthesised policies on timed properties to estimate execution-time robot performance.

Definition 7.2. The *TVMA-SSP* for robot r_i is defined as:

- A TVMA $\mathcal{A}_i$ according to Definition 7.1.
- A cost function $c_i : S_{\mathcal{A}_i} \times A_{\mathcal{A}_i} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ that gives the expected duration over all congestion bands:

$$c_i(s, e, t) = \begin{cases} \sum_j cong(r_i, e, j, t) \cdot \mathbb{E}[\rho_C(e, c^j)] & s \in V \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

  where $j \in \{0, ..., |C_e| - 1\}$.
- A set of goal states G_i defined as:

$$G_i = \{v \in V \mid v = v_i^{goal}\} \tag{7}$$

Time-varying transition functions increase the model complexity [31], [32], making an exact solution to the TVMA-SSP impractical. Therefore, we present an approximate solution, which adapts LRTDP [54] to incrementally build and solve an MDP abstraction of the TVMA.

B. LRTDP for TVMAs

To efficiently solve the TVMA-SSP for robot r_i, we adapt labelled real time dynamic programming (LRTDP) to TVMAs [54]. We refer to this method as ***LRTDP-TVMA***. LRTDP is a trial-based heuristic search method that uses an admissible heuristic h_i to initialise the value of states, so states unlikely to contribute to the optimal policy are not explored.

LRTDP-TVMA abstracts the TVMA behaviour into an MDP $\mathcal{M}_i$. $\mathcal{M}_i$ can be built *incrementally*, as Bellman backups are performed only on states visited during a trial. An MDP state is a topological node paired with the time of arrival to that node. Including time in the state allows us to consider the time-varying nature of the transitions [33]. In the TVMA, continuous-time robot behaviour is captured by timed transitions. We approximate the continuous-time behaviour in an MDP by discretising the PTDs. We apply the method in [55], which formulates discretisation as a constrained non-linear optimisation problem. The objective function is chosen to preserve the mean and variance of the original distribution. By preserving more than the mean, we can accurately capture features of the original distribution.

Definition 7.3. For an edge $e \in E$ and congestion band $c^j \in C_e$, the *discretisation* of PTD $\rho_C(e, c^j)$ is a discrete random variable $D_e^j \in \{d_{e,1}^j, ..., d_{e,K}^j\}$, where $Pr(D_e^j = d_{e,k}^j) = p_{e,k}^j$, $\forall k \in \{1, ..., K\}$, and K is a user-defined parameter.

Algorithm 1 LRTDP-TVMA (adapted from [48])

Input: $\mathcal{A}_i$, v_i^{init}, convergence threshold θ, time bound T
Output: Greedy policy π_i, value function V

1: **function** LRTDP-TVMA($\mathcal{A}_i, v_i^{init}, \theta, T$)
2: **while** $(v_i^{init}, 0)$ not labelled solved and time left **do**
3: LRTDP-TVMA-TRIAL($\mathcal{A}_i, v_i^{init}, \theta, T$)
4: **end while**
5: **end function**
6:
7: **function** LRTDP-TVMA-TRIAL($\mathcal{A}_i, v_i^{init}, \theta, T$)
8: $visited \leftarrow$ empty stack
9: $v \leftarrow v_i^{init}$, $t \leftarrow 0$
10: **while** (v, t) not labelled solved **do**
11: push (v, t) onto $visited$
12: **// Check for termination**
13: **if** $(v, t) \in G_i$ or $t > T$ **then**
14: break
15: **end if**
16: **// Perform Bellman backup & update policy**
17: $\pi_i(v, t) \leftarrow \text{argmin}_{a \in A_{\mathcal{A}_i}} Q((v, t), a)$
18: $V((v, t)) \leftarrow Q((v, t), \pi_i(v, t))$
19: **// Sample successor MDP state**
20: $c^j \leftarrow$ sample according to $cong(r_i, \pi_i(v, t), j, t)$
21: $d^j_{\pi_i(v,t),k} \leftarrow$ sample according to $D^j_{\pi_i(v,t)}$
22: $v \leftarrow \texttt{trg}(\pi_i(v, t))$, $t \leftarrow t + d^j_{\pi_i(v,t),k}$
23: **end while**
24: **// Update solved label for visited states**
25: **while** $visited \neq$ empty stack **do**
26: $(v, t) \leftarrow$ pop the top of $visited$
27: **if** $\neg$check_solved$((v, t), \theta)$ **then**
28: break
29: **end if**
30: **end while**
31: **end function**

In MDP $\mathcal{M}_i$, the robot arrives at node v at time t, and chooses an edge e. As in the TVMA, there is a congestion distribution over edge e, dependent on t. The duration of e under congestion band c^j is determined from a value $d^j_{e,k} \in D^j_e$, with the robot completing edge e at time $t + d^j_{e,k}$. Each state/action pair in the MDP has K successors per available congestion band. The transition probabilities combine the congestion probabilities and discretised PTDs.

In Algorithm 1, we present LRTDP-TVMA, which incrementally builds the MDP abstraction of the TVMA. The main loop of LRTDP-TVMA (lines 2-4) runs trials until convergence is reached, or a fixed computational budget is exceeded.

For each LRTDP-TVMA trial (lines 7-31), we begin by setting the MDP initial state (line 9), which is the robot's initial location at time 0. A trial samples through $\mathcal{M}_i$ until the termination condition in line 13 is met. Including time in the MDP state raises the possibility of a trial never terminating. To counteract this possibility, we introduce a time bound T, where any state with a time greater than T is made a *dead end*, terminating the trial. Trials are also terminated if the goal location is reached. The goal set G_i is modified to ensure the goal is reached within time T:

$$G_i = \{(v, t) \in V \times \mathbb{R}_{\geq 0} \mid v = v_i^{goal} \text{ and } t < T\} \quad (8)$$

Assuming navigation always succeeds, a robot can reach its goal location with probability 1. Under a finite horizon, exceeding time bound T is the only way to not reach the goal. If T is above any realistic time to reach the goal, the planning model is a finite-horizon SSP with *avoidable* dead ends. Goal states are defined as in Eq. 8; a dead end is any state with time greater than T. A policy will reach a goal state or a dead end, not both. Since dead ends are avoidable, there exists a reachable goal state. Therefore, there is guaranteed to exist a policy that reaches a goal state. As trials terminate at dead ends, all trials will terminate. This allows LRTDP to synthesise optimal policies, since policies that reach the goal will do so in less time than those that reach a dead end [56].

In lines 17 and 18, we perform a Bellman backup on the current state. The value function V gives the expected cost to the goal from a state. The policy π_i informs which action to choose in a state, where the actions in $\mathcal{M}_i$ are identical to the TVMA. The Q value for a state/action pair is the immediate cost of the action, plus the value of subsequently following π_i to the goal:

$$\begin{aligned} Q((v, t), e) = c_i(v, e, t) + \\ \sum_j cong(r_i, e, j, t) \cdot \\ \sum_k p^j_{e,k} \cdot V((\texttt{trg}(e), t + d^j_{e,k})) \end{aligned} \quad (9)$$

where $j \in \{0, ..., |C_e| - 1\}$ and $k \in \{1, ..., |D^j_e|\}$. Eq. 9 implicitly defines the MDP transition function. For example, the probability of reaching state $(v', t + d^j_{e,k})$ after choosing edge $e = (v, v')$ in state (v, t) is $cong(r_i, e, j, t) \cdot p^j_{e,k}$.

In lines 20-22, we sample a successor MDP state. First, we sample a congestion band given the policy edge and current time (line 20). Next, the edge duration is sampled from the discretised PTD (line 21). The successor state is the destination of the policy edge after adding the sampled duration to the previous time.

In lines 25-30 we update the 'solved' label for states visited during the trial. A solved state does not need to be explored any further. Visiting a solved state terminates the trial (line 10). If the initial MDP state $(v_i^{init}, 0)$ is solved, LRTDP-TVMA has converged. The check_solved function is implemented as in [54]. LRTDP-TVMA maintains a search tree, which forms the explored portion of the MDP abstraction of the TVMA. The check_solved function explores the sub-tree rooted at a state (v, t). If the value of (v, t) has changed below a threshold θ since its last Bellman backup, and all of its descendants in the sub-tree are solved, (v, t) is considered solved. If (v, t) is not solved, Bellman backups are performed on all unsolved states of the sub-tree.

The value $V((v, t))$ of an MDP state (v, t) is initialised with a heuristic $h_i : S_{\mathcal{M}_i} \rightarrow \mathbb{R}_{\geq 0}$, which we define as:

$$h_i((v, t)) = \textit{min\_cost}(v, v_i^{goal}), \quad (10)$$

where *min_cost* returns the minimum cost to travel from v to v_i^{goal}, which we compute using the Floyd-Warshall algorithm [57]. 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.

Increasing the discretised PTD size K improves the granularity of $\mathcal{M}_i$. As $K \to \infty$, we obtain a continuous-space MDP that is semantically equivalent to the TVMA in Definition 7.1.

To define the policy over the TVMA, we adapt the LRTDP policy to be piecewise constant [58], where for node v at time t we return the action for the MDP state at node v with time closest to t. This is necessary for execution, as a robot will never arrive at a node at precisely a time in one of the MDP states.

C. Constructing a Robot's Route CTMC

In the congestion-aware framework, a robot plans by taking the best action considering the robots who planned before, whose effects are encapsulated by the congestion probabilities computed by the PRT. To compute these probabilities, the PRT requires a route CTMC for each robot that has planned.

Route CTMCs are constructed using a robot's MDP produced by Algorithm 1, its policy, and the PTDs. Recall that the LRTDP-TVMA search tree represents the explored portion of the MDP abstraction of the TVMA. By applying the policy at each state in the tree, we obtain the induced DTMC, which captures uncertainty over congestion and action duration. DTMC transitions first branch according to the congestion probabilities, and then the discretised PTDs, according to lines 20 and 21 of Algorithm 1.

Example 3. Consider Fig. 5a, which shows an induced DTMC annotated with actions, congestion bands and values of the discretised PTDs. The transitions combine the congestion probabilities and discretised PTDs, e.g. $\delta_{\mathcal{M}_i^{\pi_i}}(s_1, s_3) = cong(r_i, e_2, 0, t_{s_1}) \cdot Pr(D^0 = d_1^0) = 0.6 \cdot 0.9 = 0.54$.

To construct the route CTMC we first combine the DTMC and PTDs to produce a time-invariant Markov automata without action choice $\mathcal{D}_i$, i.e. where states with immediate transitions have at most one action available [23]. There is a PTD for each congestion band a robot may observe on an edge. After branching by congestion band c_e^j in a DTMC state (v, t), we reach a *pseudo-state* (v, t, c_e^j) before branching by the discretised PTD. To build $\mathcal{D}_i$, pseudo-states are replaced with the corresponding PTD. Immediate transitions to the pseudo-state are connected to the PTD initial states as in Eq. 3.

Example 4. Consider Fig. 5a and Fig. 5b, where we assume that the PTDs for the congestion bands in Fig. 5a are as depicted in Fig. 5b. Absorbing PTD states are represented by concentric circles. The smaller circles in Fig. 5a represent pseudo-states. To build $\mathcal{D}_i$, the pseudo-state for band c^0 between states s_1 and s_3 in Fig. 5a would be replaced with the PTD for band $c_{e_2}^0$. From the absorbing PTD state, we then branch according to d_1^0 or d_2^0. We do not show $\mathcal{D}_i$ due to space constraints.

To build a route CTMC from a time-invariant Markov automata without action choice, we collapse the immediate transitions into the timed transitions. The collapsing process is adapted from [59]. A timed transition followed by a sequence of immediate transitions is collapsed into a single timed transition via multiplication. The new timed transitions capture uncertainty over congestion and action duration. Immediate transitions from the initial state are collapsed into the CTMC initial state distribution. After collapsing, we label the route CTMC states as in the robot's TVMA.

Example 5. Consider Fig. 6, which illustrates the collapsing process. For example, $\Delta_{\mathcal{Q}_i}(s_0, s_4) = \Delta_{\mathcal{D}_i}(s_0, s_1) \cdot \delta_{\mathcal{D}_i}(s_1, s_2) \cdot \delta_{\mathcal{D}_i}(s_2, s_4) = \lambda \cdot 0.6 \cdot 0.8 = 0.48\lambda$. If we apply this to the time-invariant Markov automata without action choice produced from Fig. 5a and Fig. 5b as explained in Example 4, we construct the route CTMC in Fig. 5c.

D. Complexity Analysis

Our framework avoids the exponential complexity of planning on joint models. For each robot, we obtain a policy with LRTDP-TVMA, and construct a route CTMC. A robot's MDP, built by LRTDP-TVMA, is a directed acyclic graph, where up to $|A_{\mathcal{M}_i}|$ actions are enabled at each state. For each action, there are successor states for each discretised PTD value for each congestion band. Each MDP transition advances time by at least $\min_{e,j} \mathbb{E}[\rho_C(e, c^j)]$, and so the maximum MDP depth is $d = \left\lceil \frac{T}{\min_{e,j} \mathbb{E}[\rho_C(e, c^j)]} \right\rceil$, i.e. the maximum number of transitions before reaching the goal or a dead end. Therefore, the worst case state space size for a robot's MDP is $\mathcal{O}((|A_{\mathcal{M}_i}| \cdot |C_e^{\max}| \cdot K)^d)$, where $C_e^{\max}$ is the largest congestion band set. A route CTMC is built from an induced DTMC, where only one action is enabled at each state. Each DTMC transition is replaced with a PTD. Thus, the worst case route CTMC size is $\mathcal{O}(|S_{\mathcal{P}}^{\max}| \cdot (|C_e^{\max}| \cdot K)^d)$, where $S_{\mathcal{P}}^{\max}$ is the largest PTD state space. The time complexity for route CTMC construction is equal to its size. Neither of these models depend on the number of robots.

The team size only impacts the time complexity of Bellman backups: let $S_{\mathcal{Q}}^{\max}$ be the largest route CTMC state space, and Λ be the maximum exit rate across all PTDs. The worst case time complexity for CTMC transient analysis is $\mathcal{O}(|S_{\mathcal{Q}}^{\max}|^3 \cdot \log(\Lambda T))$ [42], [60]. The Bellman backup complexity is $\mathcal{O}(|S_{\mathcal{M}_i}| \cdot |A_{\mathcal{M}_i}| \cdot (n-1) \cdot |S_{\mathcal{Q}}^{\max}|^3 \cdot \log(\Lambda T))$; for each state/action pair, we may need to analyse $n-1$ CTMCs to compute the congestion probabilities. We abstract a robot's TVMA into a finite-horizon SSP MDP, which can be solved by performing a single Bellman backup per state [48]. This gives a time complexity of $\mathcal{O}(|S_{\mathcal{M}_i}|^2 \cdot |A_{\mathcal{M}_i}| \cdot (n-1) \cdot |S_{\mathcal{Q}}^{\max}|^3 \cdot \log(\Lambda T))$. LRTDP improves upon this in practice. Therefore, single-robot planning scales polynomially with the team size. This result extends to planning for the whole team.

VIII. Policy Execution Model Refinement

In this section, we detail our approach to Problem 2. Our goal is to evaluate a property on the policy obtained by a robot, which can be used to predict execution-time performance. Our approach leverages our solution to Problem 1, namely the use of route CTMCs to capture robot policy execution under congestion.

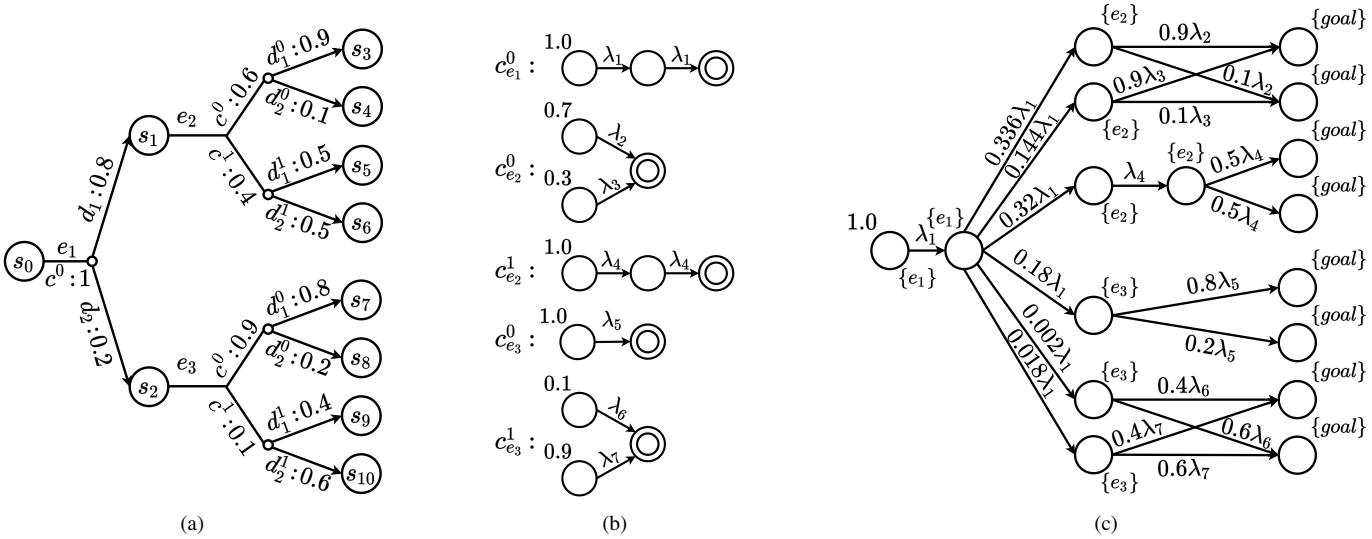


Fig. 5. The process of generating a route CTMC from a robot's policy. (a) An induced DTMC $\mathcal{M}_i^{\pi_i}$. (b) PTDs experienced in the MDP/DTMC. (c) The route CTMC $\mathcal{Q}_i$ produced by applying our method to Fig. 5a and Fig. 5b.

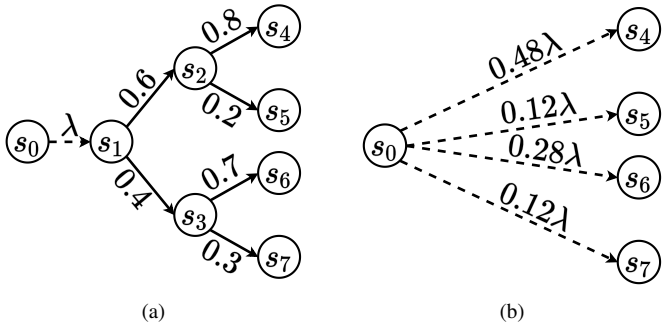


Fig. 6. The process of collapsing immediate transitions into timed transitions. Solid lines are immediate transitions; dashed lines are timed. (a) A time-invariant Markov automata $\mathcal{D}_i$ without action choice. (b) The CTMC $\mathcal{Q}_i$ obtained from collapsing the immediate transitions.

A. The Full Joint Model

To evaluate properties on robot policies requires model checking a joint CTMC which describes the asynchronous, continuous-time execution of the team. The joint CTMC combines the single-robot CTMCs and accurately captures the effects of congestion by observing congestion levels directly. The joint CTMC acts in the joint state space of the robots, i.e. $S_{joint} = S_{\mathcal{Q}_1} \times ... \times S_{\mathcal{Q}_n}$, and so the joint state of the team is observable. Since the edge currently being traversed by each robot can be observed, the exact congestion levels across the environment are computed by counting the number of robots on each edge. The number of robots is then mapped to the corresponding congestion band.

In a joint state, we can observe each robot's progress through their current PTD. In the single-robot CTMCs, exponential transitions either progress the robot along its current edge PTD, or switch it to a new PTD if the edge is finished. Joint CTMC states consider the outgoing transitions for *each robot*. Each joint CTMC transition is associated with one robot, and changes only the local state for that robot. When a transition fires that finishes a robot's current PTD, their next PTD is determined by the policy edge and congestion level on that edge.

By tracking each robot's progress, the joint CTMC accurately captures the effects of congestion on policy execution. However, the size of the state space increases exponentially with the number of robots. Further, the size of the joint state space increases as PTDs become larger; many PTD states may be required to accurately model edge durations. Empirically, we observe that it is impractical to construct and model check the joint CTMC for more than 2 robot problems. Statistical model checking techniques [61] analyse the joint CTMC without enumerating the full state space. However, these methods become cumbersome for large numbers of robots. Therefore, we desire a more compact, scalable representation of joint robot behaviour.

B. Approximating the Joint Model with Single-Robot Models

As an alternative to model checking the joint CTMC, we consider model checking the single-robot CTMCs stored in the PRT. However, the *initial* route CTMC for r_i only considers robots r_1 to r_{i-1}, providing inaccurate congestion information. We require more accurate representations of congestion, and so in Algorithm 2 we propose an iterative procedure that incrementally updates the route CTMCs after *all* robots have planned. In each iteration, a route CTMC is selected and refined using the current information in the PRT. The updated route CTMC is then inserted back into the PRT, ready for the next iteration.

In Line 2 of Algorithm 2, we choose the next robot to update. In Line 3, we refine the CTMC for the chosen robot using the method in Section VII-C and the current congestion probabilities obtained from the PRT. The refined route CTMC differs only in its congestion probabilities; its structure remains unchanged. Refining a robot's route CTMC does not change its policy. We then update the robot's CTMC in the PRT in Line 4. Lines 2-4 are repeated until all models converge.

Choosing the Next Robot: The `choose_next_robot` function in Algorithm 2 may be defined arbitrarily under a

Algorithm 2 The Iterative CTMC Refinement Procedure

Input: PRT, set of policies $\Pi = \{\pi_i, \pi_2, ..., \pi_n\}$
Output: Refined set of CTMCs $\{\mathcal{Q}_1, \mathcal{Q}_2, ..., \mathcal{Q}_n\}$

```
while not converged do
    r_i ← choose_next_robot()
    Q_i ← refine_CTMC(r_i)
    update(r_i, Q_i)
end while
return {Q_1, Q_2, ..., Q_n}
```

fairness assumption, i.e. each robot is selected infinitely often. However, this choice affects the rate at which Algorithm 2 converges. In Section IX, we analyse a number of heuristics for choosing the next robot.

Testing for Convergence: To test convergence of Algorithm 2, we consider how a robot's route CTMC changes between updates. We introduce a distance function over CTMCs with the same state space, defined as the maximum absolute difference in the rate of a transition between two CTMCs with the same state space $S_{\mathcal{Q}}$:

$$d(\mathcal{Q}, \mathcal{Q}') = \begin{cases} \max\limits_{s,s'} |\Delta_{\mathcal{Q}}(s,s') - \Delta_{\mathcal{Q}'}(s,s')| & S_{\mathcal{Q}} = S_{\mathcal{Q}'} \\ \infty & \text{otherwise} \end{cases} \tag{11}$$

Defining $\mathcal{Q}_i$ as the CTMC for r_i prior to its last refinement and $\mathcal{Q}_i'$ as the current CTMC for r_i, the test for convergence is then defined as:

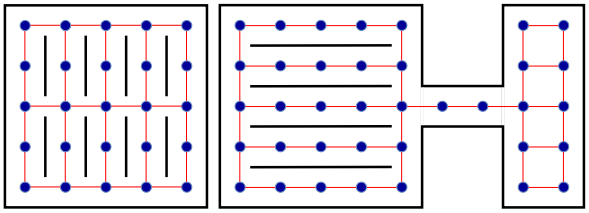
$$\max_{r_i \in R} d(\mathcal{Q}_i, \mathcal{Q}_i') < \xi, \tag{12}$$

where ξ is a small convergence threshold.

The CTMCs produced by Algorithm 2 are much smaller than the joint CTMC and therefore more practical to model check. However, these CTMCs are approximate. The single-robot CTMCs only reproduce the joint CTMC exactly in uncongested problems, where the joint CTMC is the parallel composition of the single-robot CTMCs. In all other cases, the single-robot CTMCs contain less information than the joint CTMC, as the congestion probabilities summarise the effects of congestion using only information local to r_i, whereas the joint CTMC keeps track of the global team state. In the next section we will show that the refined single-robot CTMCs provide a more accurate model of congestion than the initial CTMCs stored in the PRT after planning.

IX. Experiments

In this section we analyse the congestion-aware framework in terms of its scalability, execution-time performance, and predictions of execution-time behaviour. All experiments are run on Ubuntu 16.04, with an Intel Core i7-8700 CPU@3.2GHz, and 64GB of RAM. We use the PRISM model checker to compute the transient probabilities of CTMCs, and for model checking [62]. All software is written in Python, except for PRISM, which is written in Java/C++. All



(a) (b)

Fig. 7. The maps used for the synthetic experiments. (a) A 5×5 warehouse. (b) Warehouse with tunnel.

experimental problems follow Problem 1, i.e. find a set of policies that take each robot to its goal location.

In the PRT, very small congestion probabilities are sometimes observed that bear no practical gain. Therefore, we define a *pruning threshold* ε and set all congestion probabilities below ε to 0. The congestion distribution is then normalised to adjust for the removed probability mass.

For all experiments, we set the discretised PTD size to $K = 1$, i.e. we consider the expected value of the original PTD with probability 1. Increasing K increases the size of the route CTMCs. Reading larger CTMCs into PRISM becomes a bottleneck, decreasing scalability. Further, for the problems in this paper, increasing K did not improve the policies.

We compare the congestion-aware planner against two baselines. The first baseline treats all robots independently, i.e. robots ignore each other during planning. Under this baseline robots follow their shortest path to the goal, assuming no congestion. The second baseline acts similarly to our framework: robots plan sequentially using the priority order heuristic in [18], and insert a route CTMC into the PRT after planning. However, under this baseline we only consider an edge during planning if the probability of one or more robots on the edge is less than a threshold $\epsilon = 0.1$. If an edge is considered, the TVMA states that the robot deterministically traverses the edge under no congestion. The threshold ϵ is set as low as possible while still allowing plans to be synthesised. This baseline is similar to a deterministic MAPF approach, as robot interactions are almost entirely avoided during planning. If the threshold is set to 0, the environmental uncertainty would result in no plans being found, and so avoidance cannot be guaranteed during execution. These baselines provide two orthogonal approaches for handling execution-time robot interactions during planning: the independent baseline ignores interactions, whereas the MAPF baseline handles interactions conservatively. The high-level planner does not resolve execution-time interactions between robots; this is handled by a motion planner (see Section IX-B).

A. Validation in Synthetic Environments

We created 3 maps using synthetic data. These maps were a 5×5 and 15×15 warehouse map (see Fig. 7a, the 15×15 map is the 5×5 map scaled up), as well as a warehouse with a tunnel, designed to produce congestion (see Fig. 7b). The map sizes refer to the number of topological nodes along each dimension.

The underlying duration distributions were created using lognormals, where for each successive number of robots, the

mode and variance is increased, as observed in practice. We generated 1000 samples from each lognormal, and fit PTDs using the method in [51]. For each edge we defined four congestion bands, $[0,0], [1,3], [4,5], [6,n-1]$. All edges follow the same distributions, with minor variations made on each edge to ensure a representative state space in the MDPs produced by Algorithm 1. We set the PRT pruning threshold as $\varepsilon = 10^{-4}$ and the LRTDP-TVMA time bound as $T = 200s$. LRTDP is run until convergence or 150 trials are run. $\mathbb{E}[\mathcal{P}_\omega]$ is set to the expected duration of an edge (which are all of the same length) under no congestion.

Planner Scalability. To analyse the planner's scalability, 20 random robot configurations were generated for each environment for 2-15 robot problems. Scalability is measured in terms of the time taken for all robots to obtain a policy. This includes the time to construct route CTMCs and compute congestion probabilities. The results of this experiment for the congestion-aware framework and both baselines are displayed in Fig. 8.

The congestion-aware framework mitigates the exponential state space increase observed in joint models on all maps. 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 across problem configurations. Under lower congestion, the branching factor of the MDPs produced by Algorithm 1 decrease as fewer congestion bands are experienced. This reduces planning time, as LRTDP-TVMA converges in fewer trials and fewer congestion probabilities are computed. For example, the times for the warehouse with tunnel map are typically longer than for the 5×5 warehouse despite their similar size, as the tunnel causes congestion. In the worst case, the congestion-aware framework takes a significant time to plan, as prolonged congestion drastically increases the size of the MDP abstraction of the TVMA. For example, in Fig. 8b and Fig. 8c, 1 out of the 20 problems takes much longer to plan for.

The independent baseline is significantly faster than the other methods, as planning reduces to solving A* search over the topological map. The congestion-aware method plans quicker than the MAPF baseline approximately 60% of the time. The constrained behaviour of the MAPF baseline causes robots to take longer routes, increasing the time of LRTDP-TVMA trials. However, in the worst case, the congestion-aware method takes significantly longer to plan (cf. Fig. 8b). In highly congested environments, the MDPs built by LRTDP-TVMA will have a high branching factor, increasing the state space. Computing congestion probabilities requires expensive model checking operations, increasing planning time. This state space increase is much slower under the MAPF baseline, as only one congestion band is experienced per edge. The effects of congestion also cause the MAPF baseline to frequently use wait actions, which do not require costly PRT operations.

Performance of the Obtained Policies. To evaluate performance in synthetic environments, we generated a problem for 2-10 robots on the 5×5 warehouse map. Each additional robot increases the chance of congestion. For each problem we evaluate the execution-time team performance in terms of the makespan. We measure the makespan by generating 1000 samples through the full joint CTMC for each method/number of robots. Empirically we found that 1000 samples provided an effective trade-off between computational cost and variance over the makespan. Results for this experiment are shown in Fig. 9.

Outliers in Fig. 9 occur when congestion is heavier than expected, or the sampled durations are near the PTD tails. Beyond 4 robots, the congestion-aware planner outperforms both baselines. The MAPF baseline is conservative, synthesising long routes with high makespans. The independent baseline ignores robot interactions. This causes heavy congestion during execution, increasing the makespan. Conversely, our planner effectively distributes robots across the map, reducing the makespan. For 2-4 robots, all methods perform similarly. For fewer robots there is less congestion, and so all 3 methods synthesise similar policies. In Section IX-B, we evaluate the planning methods on simulated robot behaviour.

The Effect of Congestion Bands. Next, we analyse how the number of congestion bands per edge affects scalability and makespan. Using the warehouse with tunnel map in Fig. 7b, we generate a single heavily congested problem for 10 robots. We vary the number of congestion bands between 2 and 10. We measure the total planning time over 20 repeats for each number of congestion bands. The makespan is estimated by generating 1000 samples through the full joint CTMC that results from the synthesised policies for each number of congestion bands. The joint CTMC uses a congestion band for each number of robots, the most accurate representation of congestion. Congestion bands were chosen to be as equal in size as possible, except the first congestion band is always $[0,0]$. Where this was not possible the congestion bands were smaller for lower numbers of robots. For example, for 5 congestion bands, the bands were set to $[0,0], [1,2], [3,4], [5,6], [7,9]$. In Fig. 10 and Fig. 11 we show how the scalability and makespan change with the number of congestion bands.

The complexity of planning increases with the number of congestion bands, as the branching factor of the MDPs produced by Algorithm 1 increases. For 8-10 bands, the rate of increase decreases. The newly added congestion bands are for higher numbers of robots, which are rarely experienced. The effect on the MDP branching factor is minimal, and so there is little change in planning time. For 2 and 3 congestion bands, the planning time is higher than for 4. If there are too few congestion bands, we plan on an inaccurate model of congestion. For example, if we use 2 congestion bands, the effects of 1 robot on an edge is treated the same as if there were 9. Since low levels of congestion incur a large cost in this inaccurate model, robots plan to avoid all congestion, which increases the length of routes and LRTDP-TVMA trials, increasing the planning time. Longer routes give higher makespans; in Fig. 11, the makespans for 2 and 3 congestion bands are higher on average than for 4-10. The increased makespans demonstrate the consequences of an inaccurate congestion model. It is much quicker to plan for 4 congestion bands than for 10, despite no significant change in makespan. This suggests the robots are planning near identical routes. Therefore, using more congestion bands may increase the planning time at no practical gain. With this, we see that we can synthesise

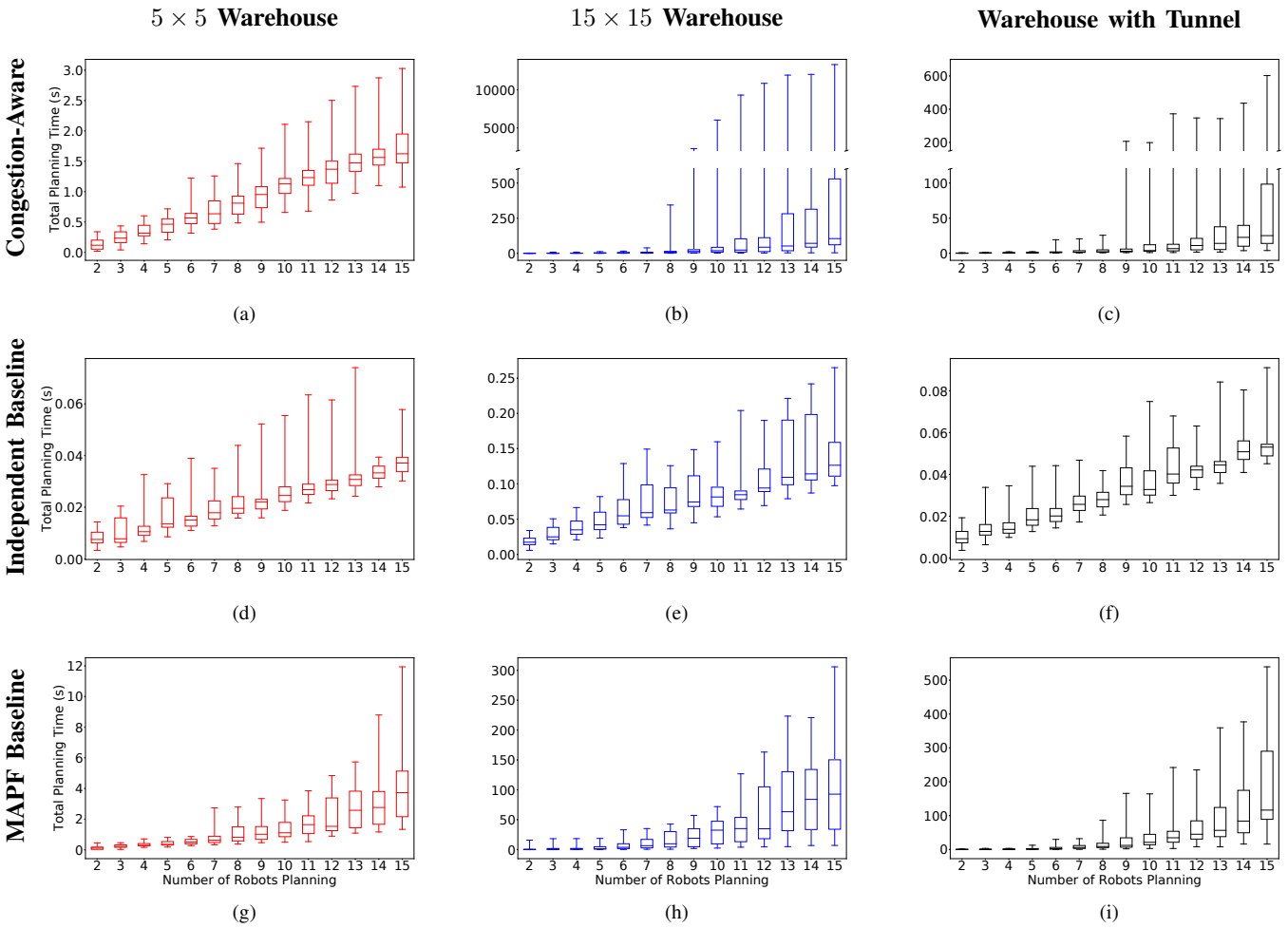


Fig. 8. The scalability across the warehouse maps. Fig. 8a - 8c show the scalability of the congestion-aware method. Fig. 8d - 8f show the scalability of the independent baseline, and Fig. 8g - 8i show the scalability of the MAPF baseline.

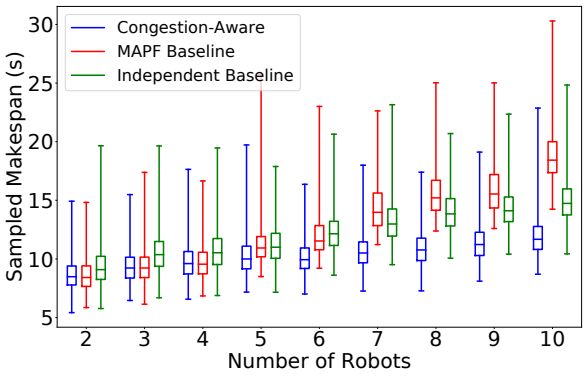


Fig. 9. The observed makespans for each of the 3 planning methods in the synthetic policy evaluation experiment.

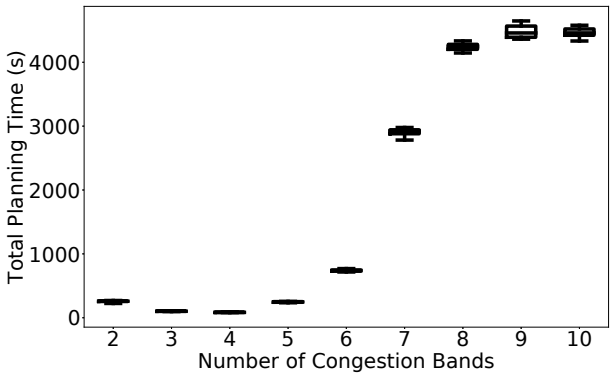


Fig. 10. The effect of the number of congestion bands on planning time.

efficient behaviour using a small number of congestion bands.

Analysing the Iterative Refinement Procedure. For each problem on the warehouse with tunnel map used to test planner scalability (cf. Fig. 7b), we ran Algorithm 2 after planning to analyse its scalability and performance. The convergence threshold ξ was set to $\xi = 10^{-6}$. The order in which CTMCs are refined in Algorithm 2 affects the convergence speed (cf. Section VIII). We compare 3 different heuristics:

- *Random*: The next robot is chosen randomly.
- *Sequential*: Robots are selected in a round-robin fashion according to the priority order.

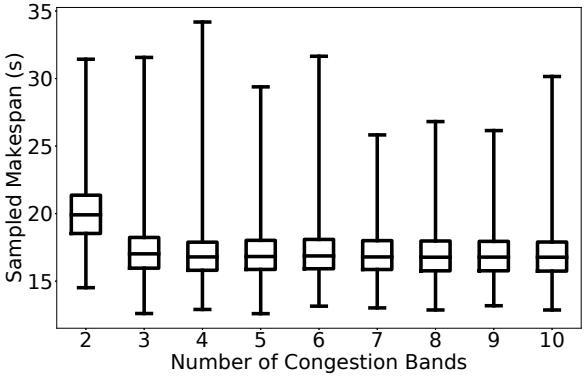


Fig. 11. The effect of the number of congestion bands on makespan.

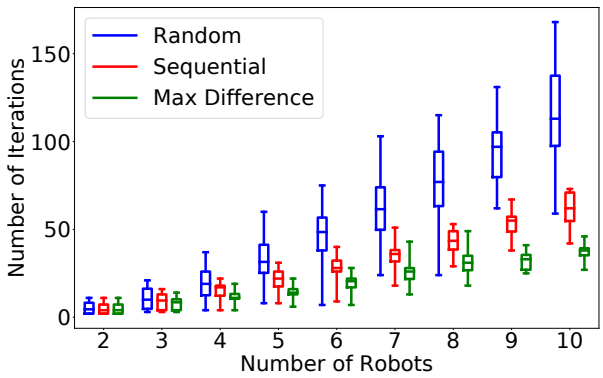


Fig. 12. The number of iterations of Algorithm 2 for each heuristic to reach convergence.

- *Maximum difference*: We choose the robot whose CTMC changed the most the last time it was refined, according to the CTMC difference metric in Eq. 11. To compute this, all robots must have their CTMC rebuilt at least once. Therefore, for the first n iterations we use the sequential heuristic. The CTMC difference acts as a proxy for how inaccurate a robot's CTMC is. This heuristic is similar to prioritised sweeping, used for ordering dynamic programming updates [63].

The above heuristics operate under a fairness assumption. However, the use of a convergence threshold breaks this assumption, as each CTMC is refined only a finite number of times. This may affect the results of Algorithm 2 under different heuristics.

In Fig. 12, we detail the number of iterations/CTMC refinements before convergence in Algorithm 2. In Table I we show the mean times with standard deviation. The random heuristic scales poorly, as it often rebuilds the CTMCs of robots who have no new information to incorporate, e.g. if a robot's route is clear of all other robots. The maximum difference heuristic performs best in terms of iterations and time, as it rebuilds only the CTMCs likely to be improved. The sequential heuristic performs worse than the maximum difference heuristic, as the planning order forces us to rebuild the CTMCs of robots who may have already converged. The worst case convergence times highlight the difference between heuristics. For 10 robots, the worst case time for the sequential and random heuristic were approximately 2.6 and 4.6 times longer than the maximum difference heuristic respectively.

Though the full joint CTMC cannot be replicated using single-robot CTMCs, Algorithm 2 is designed to provide a close, tractable approximation. Therefore, the final CTMCs produced by Algorithm 2 should provide a closer approximation of the joint model than the initial CTMCs stored in the PRT immediately after planning. To demonstrate this improvement, we compare the evaluation of expected time and time-bounded reachability properties between the initial and final CTMCs. We consider the mean error of the single-robot CTMCs from the true joint CTMC values across the team. To evaluate the joint model, we generated 1000 samples through it. Sampling is similar to statistical model checking techniques [61]. The

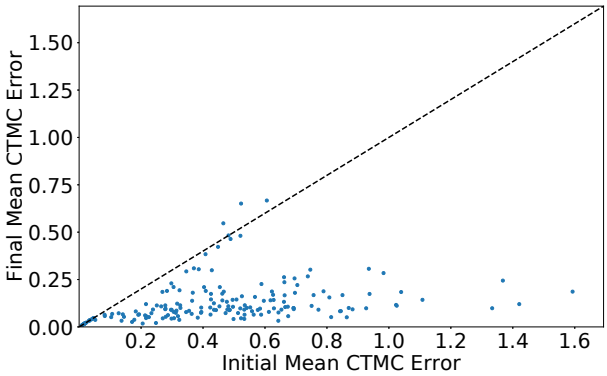


Fig. 13. The difference in expected time error between the initial and final CTMCs. Each point gives the mean error across the entire team.

advantage of using Algorithm 2 over statistical model checking on the joint model is that the single-robot CTMCs are property independent. Using statistical model checking requires a new set of samples for each property to be evaluated, which is cumbersome, particularly for larger problems. For time-bounded reachability, the time bound is selected as the expected time to traverse the longest straight line of nodes in the map, under no congestion. This is computed using the PTDs. In Fig. 13 and Fig. 14 we show the results for expected time and time-bounded reachability respectively.

For both properties, the final CTMCs give a consistently small error against the joint model. In the majority of cases this error is lower than for the initial CTMCs. Cases where the initial and final CTMCs give the same error are attributed to uncongested problems, i.e. the robots never interact. In a few instances the initial CTMCs produce a smaller error. These instances occur in problems where 2 or more robots follow the same set of edges simultaneously. The first robot to plan underestimates the time to reach the goal in its initial CTMC. This underestimate is closer to the true value of the property than the joint CTMC, resulting in a lower mean error. These results show that evaluating timed properties on the CTMCs produced by Algorithm 2 provides a close approximation to the full joint model, providing an accurate estimate of execution-time robot behaviour.

TABLE I
THE MEAN TIMES IN SECONDS ± STANDARD DEVIATION TO COMPLETE ALGORITHM 2 FOR EACH OF THE 3 HEURISTICS.

Number of Robots	2	3	4	5	6	7	8	9	10
Random Heuristic	1.34 ± 2.05	3.78 ± 5.2	8.65 ± 11.9	38.1 ± 89.5	152 ± 270	236 ± 451	706 ± 1582	3018 ± 6967	7747 ± 18807
Sequential Heuristic	$\mathbf{1.29 \pm 1.99}$	2.98 ± 3.85	6.3 ± 7.96	20.7 ± 48.6	79.7 ± 132	161 ± 343	390 ± 903	1776 ± 4187	3948 ± 10370
Max Difference Heuristic	$\mathbf{1.29 \pm 1.98}$	$\mathbf{2.94 \pm 3.83}$	$\mathbf{5.75 \pm 6.96}$	$\mathbf{16.3 \pm 33.7}$	$\mathbf{73.3 \pm 126}$	$\mathbf{151 \pm 321}$	$\mathbf{352 \pm 874}$	$\mathbf{877 \pm 2029}$	$\mathbf{1693 \pm 4067}$

Cells in bold have the lowest mean time for that number of robots.

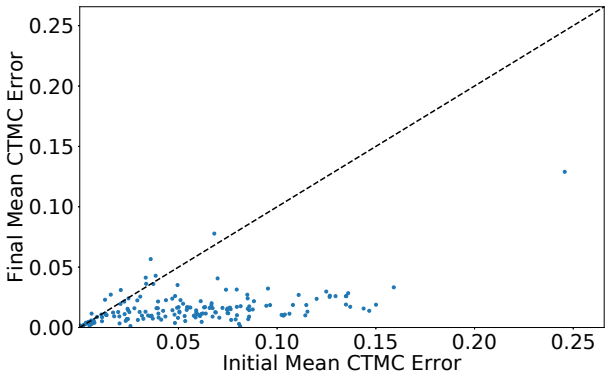


Fig. 14. The difference in time-bounded reachability error between the initial and final CTMCs. Each point gives the mean error across the entire team.

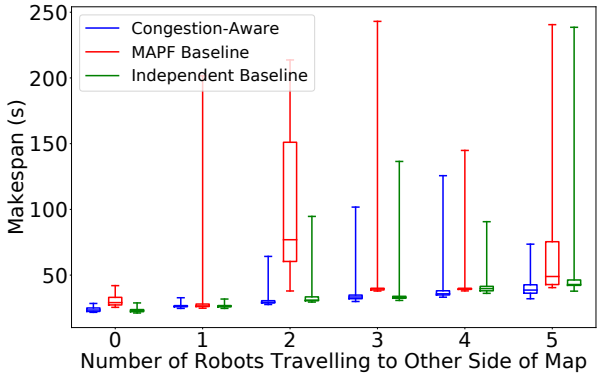


Fig. 15. The observed makespans for each of the 3 planning methods in the simulation experiment.

B. Validation in a Simulated Environment

To test execution-time performance, we simulate 5 robots in ROS using the Stage simulator with the map seen in Fig. 1. All robots use Smach state machines to coordinate navigation using Move Base Flex tailored to multi-robot systems [64]. The state machines constrain robots to travel via topological edges and maintain collision avoidance. A centralised node detects intersecting robot paths and forces some robots to wait to avoid collisions. When a robot executes a wait action, a waiting time is sampled from $\mathcal{P}_\omega$. Having two tunnels on the map allows robots to take the longer tunnel if the shorter one is too congested. PTDs were fit from simulated data, by repeatedly sending robots through targeted edges on the map. Whenever 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. Fig. 3 shows an example set of PTDs from this map, alongside the data collected. We use 5 congestion bands, one for each number of robots.

Performance of the Obtained Policies. We evaluated 6 problem configurations on the congestion-aware planner and 2 baselines. For each problem, each method was run 40 times. The value of $\mathbb{E}[\mathcal{P}_\omega]$ was set to the expected duration of the shortest edge, under no congestion. Across all problems, 3 robots begin on the larger side of the map and 2 on the smaller side. In problem 0, each robot's goal is on the same side of the map as their initial location. For problems 1-5, the problem number states how many robots must travel to the other side of the map. Congestion increases as more robots switch sides of the map. In this experiment we measure the *makespan*, i.e. the time until the last robot reaches its goal.

The makespans for each problem and method are presented in Fig. 15. In Table II, we show the results of a Mann-Whitney U test comparing the congestion-aware method to both baselines. The null hypothesis is rejected if $p < 0.05$. Outliers occur when the motion planner takes longer to resolve congestion. The stochastic nature of the environment means congestion may still occur under the MAPF baseline. This significantly affects the makespan, since congestion was unexpected during planning. The congestion-aware method outperforms the MAPF baseline in all problems, as the MAPF baseline avoids congestion, even if it would only incur a small cost.

The congestion-aware method and independent baseline produce similar policies for less congested problems. For problems 0, 1 and 3, there is no statistical difference between their makespans, as the cost of congestion is less than that of alternate routes. However, for problem 2 the congestion-aware framework produces lower makespans. This is likely due to the independent baseline choosing randomly between two identical length routes, where one causes congestion. The congestion-aware method will choose the uncongested route. For problems 4 and 5, the congestion-aware method produces statistically lower results than the independent baseline, as it routes robots through the top tunnel in Fig. 1, minimising congestion. The independent baseline routes all robots through the central tunnel, causing heavy congestion. This demonstrates how the independent baseline fails to handle robot interactions.

Accuracy of Policy Evaluation. The refined single-robot CTMCs (and the full joint CTMC) rely on the environment model being accurate, i.e. the PTDs accurately model action durations and the effects of congestion. To test whether this holds in practice, we ran Algorithm 2 using the congestion-aware policies obtained in the previous experiment. We then evaluated the expected time property on each robot's CTMC, as well as

TABLE II
THE p VALUES COMPUTED FROM A ONE-SIDED MANN-WHITNEY U TEST COMPARING THE SIMULATION TIMES FOR THE CONGESTION-AWARE METHOD AGAINST THE 2 BASELINES.

Problem Number	0	1	2	3	4	5
Against Independent Baseline	8.92×10^{-1}	3.48×10^{-1}	$\mathbf{2.41 \times 10^{-6}}$	6.59×10^{-1}	$\mathbf{1.26 \times 10^{-5}}$	$\mathbf{4.42 \times 10^{-5}}$
Against MAPF Baseline	$\mathbf{5.04 \times 10^{-12}}$	$\mathbf{3.27 \times 10^{-2}}$	$\mathbf{1.42 \times 10^{-12}}$	$\mathbf{2.42 \times 10^{-10}}$	$\mathbf{1.29 \times 10^{-6}}$	$\mathbf{2.73 \times 10^{-8}}$

Cells in bold are where $p < 0.05$ and the mean rank is lower for the congestion-aware method, i.e. the congestion-aware method gives statistically significantly lower makespans.

the joint CTMC. We obtain joint CTMC values by averaging 1000 samples for each problem. We then compare these values against the times recorded for each robot over the 40 simulated runs of each problem. These results can be seen in Fig. 16.

Both the single-robot and joint CTMC times typically underestimate the simulation times, suggesting some congestion effects are unmodelled. The congestion-aware framework makes a number of modelling assumptions, such as the use of congestion bands and PTDs to model action durations. During planning we consider the congestion level at the point a robot joins an edge. However, other robots may join/leave an edge after this point during execution. Further, we do not distinguish between directions of congestion, or consider the proximity of other robots on an edge. The PTDs may also be erroneous, as distributions are generalised across edges of the same length. Despite this, in many cases the predicted expected times are close approximations of the simulation times. This shows how the PTDs and PRT provide an effective approximation of congestion.

X. CONCLUSION

In this paper, we have presented a congestion-aware framework for multi-robot planning under uncertainty. We synthesise policies using LRTDP-TVMA on single-robot TVMAs that explicitly capture the effects of congestion. By modelling navigation durations probabilistically, the policies we synthesise are resilient to previously observed, but not explicitly modelled, sources of delay. Further, route CTMCs can be used to predict execution-time robot performance. Though we have focused on multi-robot navigation tasks under uncertain travel times, our framework can be adapted to solve any problem where multiple robots must coordinate under temporal uncertainty. In future work, we will consider planning for collaborative meeting tasks under congestion, and how we can relax the sequential planning assumption to build more accurate planning models. Moreover, we will enable robots to re-plan online if something unexpected occurs during execution.

REFERENCES

[1] D. Claes, F. Oliehoek, H. Baier, and K. Tuyls, "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, 2017, pp. 492–500.

[2] G. Das, G. Cielniak, P. From, and M. Hanheide, "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)*, 2018.

[3] J. Karlsson, C.-I. Vasile, J. Tumova, S. Karaman, and D. Rus, "Multi-vehicle motion planning for social optimal mobility-on-demand," in *Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7298–7305.

[4] A. Felner, R. Stern, S. E. Shimony, E. Boyarski, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek, "Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges," in *Proceedings of the Tenth Annual Symposium on Combinatorial Search (SoCS)*, 2017, pp. 29–37.

[5] P. Buchholz, J. Kriege, and I. Felko, *Input Modeling with Phase-Type Distributions and Markov Models: Theory and Applications*. Springer, 2014.

[6] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[7] B. Golden, A. Assad, L. Levy, and F. Gheysens, "The fleet size and mix vehicle routing problem," *Computers & Operations Research*, vol. 11, no. 1, pp. 49–66, 1984.

[8] C. Street, B. Lacerda, M. Mühlig, and N. Hawes, "Multi-robot planning under uncertainty with congestion-aware models," in *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2020, pp. 1314–1322.

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

[10] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK '96)*, 1996, pp. 195–210.

[11] J. V. Messias, M. Spaan, and P. Lima, "Gsmdps for multi-robot sequential decision-making," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, pp. 1408–1414.

[12] D. Claes, P. Robbel, F. A. Oliehoek, K. Tuyls, D. Hennes, and W. Van der Hoek, "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, 2015, pp. 881–890.

[13] S. Zhang, Y. Jiang, G. Sharon, and P. Stone, "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, 2017, pp. 501–510.

[14] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "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, 2019, pp. 936–943.

[15] F. Faruq, D. Parker, B. Lacerda, and N. Hawes, "Simultaneous task allocation and planning under uncertainty," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3559–3564.

[16] D. Silver, "Cooperative pathfinding," in *Proceedings of the first AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*. AAAI Press, 2005, pp. 117–122.

[17] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2001, pp. 271–276.

[18] J. P. Van Den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2005, pp. 430–435.

[19] W. Wu, S. Bhattacharya, and A. Prorok, "Multi-robot path deconfliction through prioritization by path prospects," in *Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9809–9815.

[20] M. Bennewitz, W. Burgard, and S. Thrun, "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots," *Robotics and Autonomous Systems*, vol. 41, no. 2-3, pp. 89–99, 2002.

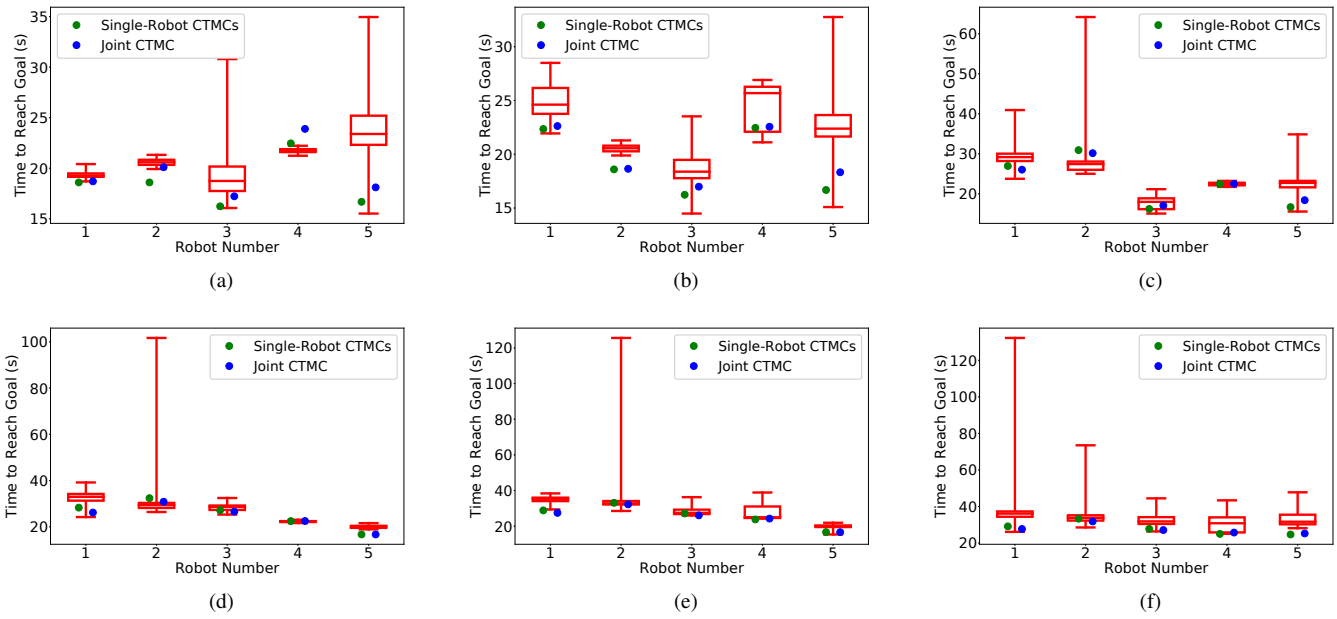


Fig. 16. Box plots showing the time taken for each robot to reach its goal in simulation. Green circles show the expected time value computed on the CTMCs produced by Algorithm 2. Blue circles show the expected time value computed by sampling through the joint CTMC. Fig. 16a - 16f show results for problems 0-5 respectively, where the problem number indicates how many robots were forced to travel through one of the tunnels in the map in Fig. 1.

[21] C. Amato, G. D. Konidaris, and L. P. Kaelbling, "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, 2014, pp. 1273–1280.

[22] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[23] C. Eisentraut, H. Hermanns, and L. Zhang, "On probabilistic automata in continuous time," in *Proceedings of the 2010 25th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2010, pp. 342–351.

[24] M. Mansouri, B. Lacerda, N. Hawes, and F. Pecora, "Multi-robot planning under uncertain travel times and safety constraints," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 478–484.

[25] C. Azevedo, B. Lacerda, N. Hawes, and P. Lima, "Long-run multi-robot planning for persistent tasks," in *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 4323–4328.

[26] H. L. Younes and R. G. Simmons, "Solving generalized semi-markov decision processes using continuous phase-type distributions," in *Proceedings of the Nineteenth AAAI Conference on Artificial Intelligence*, 2004, pp. 742–747.

[27] X. Guo and O. Hernández-Lerma, *Continuous-time Markov decision processes - Theory and Applications*. Springer-Verlag Berlin Heidelberg, 2009.

[28] A. M. Ayala, S. B. Andersson, and C. Belta, "Probabilistic control from time-bounded temporal logic specifications in dynamic environments," in *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 4705–4710.

[29] J. A. Boyan and M. L. Littman, "Exact solutions to time-dependent mdps," in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2000, pp. 1026–1032.

[30] E. Rachelson, P. Fabiani, and F. Garcia, "Timdppoly: An improved method for solving time-dependent MDPs," in *Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2009, pp. 796–799.

[31] L. Liu and G. S. Sukhatme, "A solution to time-varying markov decision processes," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1631–1638, 2018.

[32] J. Xu, K. Yin, and L. Liu, "Reachable space characterization of markov decision processes with time variability," in *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[33] P. Duckworth, B. Lacerda, and N. Hawes, "Time-bounded mission planning in time-varying domains with semi-mdps and gaussian processes," in *Proceedings of the 4th Annual Conference on Robot Learning (CoRL)*, 2020.

[34] A. Andreychuk, K. Yakovlev, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 39–45.

[35] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[36] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 5628–5635.

[37] P. Surynek, "Multi-agent path finding with continuous time and geometric agents viewed through satisfiability modulo theories (smt)," in *Proceedings of the Twelfth Annual Symposium on Combinatorial Search (SoCS)*, 2019, pp. 200–201.

[38] G. Wagner and H. Choset, "Path planning for multiple agents under uncertainty," in *Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS)*, 2017, pp. 577–585.

[39] ——, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1–24, 2015.

[40] H. Ma, T. S. Kumar, and S. Koenig, "Multi-agent path finding with delay probabilities," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017, pp. 3605–3612.

[41] R. Shi, P. Steenkiste, and M. M. Veloso, "Sc-m*: A multi-agent path planning algorithm with soft-collision constraint on allocation of common resources," *Applied Sciences*, vol. 9, no. 19, 2019.

[42] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic model checking," in *International School on Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*. Springer, 2007, pp. 220–270.

[43] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton, "Model-checking continuous-time markov chains," *ACM Transactions on Computational Logic (TOCL)*, vol. 1, no. 1, pp. 162–170, 2000.

[44] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen, "Model checking continuous-time markov chains by transient analysis," in *Proceedings of the International Conference on Computer Aided Verification (CAV)*. Springer, 2000, pp. 358–372.

[45] A. Nikou, D. Boskos, J. Tumova, and D. V. Dimarogonas, "On the timed temporal logic planning of coupled multi-agent systems," *Automatica*, vol. 97, pp. 339–345, 2018.

[46] S. Karaman and E. Frazzoli, "Vehicle routing problem with metric temporal logic specifications," in *Proceedings of the 2008 47th IEEE Conference on Decision and Control (CDC)*. IEEE, 2008, pp. 3953–3958.
[47] Z. Liu, J. Dai, B. Wu, and H. Lin, "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications," in *Proceedings of the 2017 American Control Conference (ACC)*. IEEE, 2017, pp. 2516–2521.
[48] Mausam and A. Kolobov, *Planning with Markov Decision Processes: An AI Perspective*. Morgan & Claypool Publishers, 2012.
[49] D. P. Bertsekas and J. N. Tsitsiklis, "An analysis of stochastic shortest path problems," *Mathematics of Operations Research*, vol. 16, no. 3, pp. 580–595, 1991.
[50] H. Okamura, R. Watanabe, and T. Dohi, "Variational bayes for phase-type distribution," *Communications in Statistics – Simulation and Computation*, vol. 43, no. 8, pp. 2031–2044, 2014.
[51] A. Thummler, P. Buchholz, and M. Telek, "A novel approach for phase-type fitting with the em algorithm," *IEEE Transactions on Dependable and Secure Computing*, vol. 3, no. 3, pp. 245–258, 2006.
[52] W. Biscarri, S. D. Zhao, and R. J. Brunner, "A simple and fast method for computing the poisson binomial distribution function," *Computational Statistics & Data Analysis*, vol. 122, pp. 92–100, 2018.
[53] H. Hatefi and H. Hermanns, "Model checking algorithms for markov automata," *Electronic Communications of the EASST*, vol. 53, 2012.
[54] B. Bonet and H. Geffner, "Labeled rtdp: Improving the convergence of real-time dynamic programming," in *Proceedings of the Thirteenth Interanational Conference on Automated Planning and Scheduling (ICAPS)*, 2003, pp. 12–21.
[55] Z. Drezner and D. Zerom, "A simple and effective discretization of a continuous random variable," *Communications in Statistics-Simulation and Computation*, vol. 45, no. 10, pp. 3798–3810, 2016.
[56] A. Kolobov, Mausam, and D. S. Weld, "A theory of goal-oriented mdps with dead ends," in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence (UAI'12)*, 2012, pp. 438–447.
[57] R. W. Floyd, "Algorithm 97: Shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.
[58] P. Buchholz and I. Schulz, "Numerical analysis of continuous time markov decision processes over finite horizons," *Computers & Operations Research*, vol. 38, no. 3, pp. 651–659, 2011.
[59] Y. Butkova, R. Wimmer, and H. Hermanns, "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, 2017, pp. 188–203.
[60] R. Pulungan and H. Hermanns, "Transient analysis of ctmcs: Uniformization or matrix exponential?" *IAENG International Journal of Computer Science*, vol. 45, no. 2, pp. 267–274, 2018.
[61] A. Legay, B. Delahaye, and S. Bensalem, "Statistical model checking: An overview," in *Proceedings of the International Conference on Runtime Verification (RV)*, 2010, pp. 122–135.
[62] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *Proceedings of the 23rd International Conference on Computer Aided Verification (CAV'11)*. Springer, 2011, pp. 585–591.
[63] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103–130, 1993.
[64] S. Pütz, J. S. Simón, and J. Hertzberg, "Move base flex: A highly flexible navigation framework for mobile robots," in *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3416–3421.

Charlie Street received the MSci degree in computer science from the School of Computer Science at the University of Birmingham, UK in 2018. He is currently working towards the DPhil degree in engineering science within the GOALS group at the Oxford Robotics Institute, University of Oxford, UK.

His research interests include multi-robot planning under temporal uncertainty, time-varying Markov models, and formal methods.

Sebastian Pütz studied physics and computer science at Osnabrück University, Germany. From 2016, he was a Ph.D. student in the Knowledge-Based Systems group at Osnabrück University, and from 2017 to 2020 was a Ph.D. scholar of the Friedrich Ebert Foundation.

He is currently a researcher at the German Research Center for Artificial Intelligence (DFKI) in the Plan-based Robot Control group. His research focuses on flexible, multimodal and modular autonomous robot navigation, and mapping.

Manuel Mühlig received his Diploma in computer science from the Ilmenau University of Technology, Germany, in 2008. Subsequently, he was a Ph.D. student at the Research Institute for Cognition and Robotics (CoR-Lab) at Bielefeld University and received his Ph.D. degree in September 2011.

Since 2011, he has been working as a Senior Scientist at the Honda Research Institute Europe in Germany. His research focuses on interactive, goal-directed imitation learning approaches, robot motion generation and long-term autonomous robot systems.

Nick Hawes (M'17) received a BSc in artificial intelligence and computer science in 1999, and a PhD in artificial intelligence in 2003. Both degrees were obtained from the School of Computer Science at the University of Birmingham, UK.

In 2003 he moved to a postdoctoral position at the Media Lab Europe in Dublin, before returning to the University of Birmingham to work as a postdoc from 2004 to 2009. In 2009 he started as a Lecturer at the same institution, before promotion to Senior Lecturer (2013) and Reader (2015). In 2017 he moved to the University of Oxford. He is currently an Associate Professor in the Oxford Robotics Institute, Department of Engineering Science, and a Tutorial Fellow at Pembroke College. His research interests all involve the use of AI to control robot behaviour, particularly mission planning and long-term autonomy. He is an Associate Editor of the Journal of AI Research, and a Group Leader for AI/Robotics at the Turing Institute.

Prof. Hawes gave the Lord Kelvin Award Lecture for the British Science Association in 2013, and was awarded Senior Member status of AAAI in 2020.

Bruno Lacerda (M '19) received the B.Sc. degree in applied mathematics and computation and the M.Sc. degree in mathematics and applications in 2007, and the Ph.D degree in electrical and computing engineering in 2013, all from the Instituto Superior Técnico, University of Lisbon, Portugal.

Since 2017, he is a Senior Researcher at the Oxford Robotics Institute, University of Oxford, UK. Between 2013 and 2017, he was a Research Fellow at the School of Computer Science, University of Birmingham, UK. His research interests are in the use of formal approaches to specify and synthesise high-level robot controllers. To achieve this goal, he is particularly interested in the use of planning under uncertainty and probabilistic model checking techniques.