Context-Aware Modelling for Multi-Robot Systems Under Uncertainty

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

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

Michal Staniaszek
Oxford Robotics Institute,
University of Oxford
Oxford, UK
michal@robots.ox.ac.uk

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

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

ABSTRACT

Formal models of multi-robot behaviour are fundamental to planning, simulation, and model checking techniques. However, existing models are invalidated by strong assumptions that fail to capture execution-time multi-robot behaviour, such as simplistic duration models or synchronisation constraints. In this paper we propose a novel multi-robot Markov automaton formulation which models asynchronous multi-robot execution in continuous time. Robot dynamics are captured using phase-type distributions over action durations. Moreover, we explicitly model the effects of robot interactions, as they are a key factor for the duration of action execution. We also present a scalable discrete-event simulator which yields realistic statistics over execution-time robot behaviour by sampling through the Markov automaton. We validate our model and simulator against a Gazebo simulation in a range of multi-robot navigation scenarios, demonstrating that our model accurately captures high-level multi-robot behaviour.

KEYWORDS

Multi-robot systems; Markov automata; Discrete-event simulation

ACM Reference Format:
Charlie Street, Bruno Lacerda, Michal Staniaszek, Manuel Mühlig, and Nick Hawes. 2022. Context-Aware Modelling for Multi-Robot Systems Under Uncertainty. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

A formal high-level model of a multi-robot system (MRS) reasons over robot decision making abstract of low-level components such as controllers. Formal models of MRSs have been used for planning [26], simulation [11], and verification [20]. However, the success of these techniques is limited by the accuracy of the multi-robot model [2]. Existing multi-robot models make strong assumptions, such as simplistic duration models, e.g. deterministic or Gaussian models [9, 12], robot independence [4], and synchronised execution [5]. Often, these assumptions do not correlate with execution-time behaviour. For example, by assuming robot independence, we ignore execution-time interactions such as congestion, which can cause significant robot delays [34]. To accurately capture high-level multi-robot behaviour, we require formal models that capture uncertain action durations and robot interactions in continuous time.

In this paper, we present a multi-robot Markov automaton (MRMA) formulation that accurately models asynchronous multi-robot execution in continuous time. Markov automata (MA) [14] extend Markov decision processes (MDPs) [30] by separating instantaneous decision making from stochastic action durations. Robot dynamics are abstracted into continuous stochastic processes over action durations. We model these processes using phase-type distributions (PTDs) [6], which can be fitted from empirical data [36]. Moreover, the MRMA explicitly handles multi-robot interactions, such as congestion, which increase uncertainty over action execution, as robots must manoeuvre around each other to resolve motion planning conflicts [34]. To support this, we define two approaches for describing the *context* in which robot actions are executed. Combining the MRMA structure with PTDs built from empirical data enables our model to accurately capture the behaviour of robots interacting to achieve tasks in a physical world. Though there exist other continuous-time models for asynchronous MRSs [1, 26, 28, 37], our MRMA formulation is the first to explicitly model execution-time robot interactions and their effects on action durations.

The MRMA has numerous applications, such as in multi-agent reinforcement learning [7], where the MRMA could act as a surrogate world model to speed up learning while maintaining solution quality [18]. Moreover, the MRMA could be used for multi-robot planning under uncertain durations and interactions [34], or as a discrete-event simulator (DES). DESs simulate high-level multi-robot behaviour while avoiding the computational overhead of more realistic physics-based simulators [22]. We present a scalable DES called CAMAS (context-aware multi-agent simulator) which samples through the MRMA under a set of robot policies to evaluate high-level multi-robot properties, such as the time to complete a set of tasks, or the number of robot failures. We use CAMAS to empirically validate the MRMA. To the best of our knowledge, CAMAS is

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

the first DES to explicitly model the effects of robot interactions on action durations.

The main contributions of this work are (i) an MRMA model for a team of robots interacting in a shared space, which allows for asynchronous execution and duration uncertainty; (ii) a DES that samples through the MRMA to compute statistics over MRS behaviour; and (iii) an empirical validation demonstrating that our model can accurately capture the behaviour of an MRS simulated in Gazebo.

2 RELATED WORK

Multi-agent MDPs (MMDPs) extend MDPs to model non-deterministic multi-agent systems under uncertainty [5]. However, MMDPs enforce synchronisation, assuming all actions have the same fixed duration. In practice, robot action durations are inherently continuous and uncertain, where robot interactions contribute towards this uncertainty [34]. Therefore, to capture multi-robot behaviour we must model uncertainty over action duration, which requires a continuous-time Markov model. Semi-MDPs use arbitrary duration distributions for each state/action pair [35]. However, they cannot model concurrent behaviour, limiting their use as models for MRSs, where modelling concurrent execution is key. In contrast, continuous-time MDPs enable concurrency, but restrict durations to exponential delays [17]. Generalised semi-MDPs (GSMDPs) also handle concurrency, and have been applied to MRSs [28, 37], but are complex to define, and hard to solve. In fact, a standard solution method involves the use of PTDs, similar to our MA-based model [37]. By modelling the MRS as an MA, we avoid the complexity of GSMDPs. Our MA-based formulation is a more natural model for MRSs, separating instantaneous decisions (robot action choice) from durative events (the duration of such actions) represented as exponential delays [14]. This later enables us to detect robot interactions at the instant an action is triggered. Further, MA can be used as a semantics for generalised stochastic Petri nets [13], which have been used for modelling, analysis and planning for MRSs under uncertain action durations [1, 10, 26]. Our approach uses the notions of interactions and MA-based robot models introduced in [33] and [34] respectively. There, the effects of interactions are approximated on single-robot models using congestion distributions computed from continuous-time Markov chain models of policy execution. This is too inaccurate for simulation or model checking, and so in the MRMA we detect interactions in the joint state.

Physics-based simulators such as Gazebo [23], Webots [29], CoppeliaSim [32] and RaiSim [21] accurately reproduce system dynamics for realistic simulations. However, the computational overhead incurred when simulating complex MRSs forces the simulator to run close to real time, which is cumbersome when evaluating high-level multi-robot behaviour [19]. DESs such as CAMAS, which we propose in Section 6, mitigate this complexity by abstracting away low-level robot dynamics [3]. DESs typically use an event queue, which facilitates asynchronous robot execution by ordering events based on occurrence time [22]. However, stochastic duration models have been limited to Gaussians or exponentials [11, 12, 16]. In this paper, we use PTDs to model stochastic action durations, which approximate nonnegative distributions to an arbitrary precision [36]. In a DES, robot behaviour may be fixed [19], specified as finite state machines [12], or presented as generic software libraries [3]. Similar to [11], CAMAS receives a set of policies defined over the joint state. As the joint MRMA state models action progress, robots can make decisions based on the progress of others.

CAMAS simulates robot behaviour by sampling through an MRMA, similar to statistical model checking (SMC) techniques, which evaluate properties on formal models by sampling [24]. SMC has been applied to MRSs in [20], and MA in [8], where sample termination conditions are given for bounded and unbounded properties. SMC techniques for MA can be directly applied to CAMAS.

3 PRELIMINARIES

Markov Automata (MA). We use MA [14] to model multi-robot execution. MA extend MDPs to explicitly consider durative events through exponentially timed transitions.

Definition 3.1. An *MA* is a tuple $\mathcal{M} = \langle S, \bar{s}, A, \delta, \Delta \rangle$, where S is a finite set of states; $\bar{s}$ is the initial state; A is a finite set of actions; $\delta : S \times A \times S \to [0, 1]$ is an immediate transition function, such that $\delta(s, a, t)$ returns the probability of instantaneously transitioning to state t after executing action a in state s; and $\Delta : S \times S \to \mathbb{R}_{>0}$ is an exponential transition function, such that $\Delta(s, t)$ returns the rate between states s and t.

Similar to MDP transitions, immediate transitions represent action choices that occur instantaneously. Exponential transitions are associated with a value $\Delta(s, t)$, representing the rate parameter of an exponential distribution associated with their duration. The probability of the transition firing within time τ is given by $1 - e^{-\Delta(s,t)\cdot\tau}$. The *exit rate* is the sum of outgoing rates in a state s, i.e. $E(s) = \sum_{t \in S} \Delta(s, t)$. The probability of leaving state s within time τ is $1 - e^{-E(s)\cdot\tau}$; the probability of branching to state t from s is $\Delta(s, t)/E(s)$. When a state has multiple exponential transitions, a *race condition* occurs, i.e. there is a race over which transition fires. Race conditions can be efficiently resolved by sampling a duration using the exit rate, and sampling a successor state under the branching distribution.

Phase-Type Distributions (PTDs). PTDs approximate non-negative continuous distributions using the time taken to reach an absorbing state in a continuous-time Markov chain [6]. We use PTDs to model action durations.

Definition 3.2. A *PTD* is a tuple $\mathcal{P} = \langle S, init, \Delta, s^f \rangle$, where S and Δ are as in an MA. Function $init : S \to [0, 1]$ gives the probability of a state being the initial state; and $s^f \in S$ is the single absorbing state such that the probability of reaching s^f is 1. Further, $init(s^f) = 0$.

We denote the set of all PTDs with $\mathbb{P}$. Moreover, subscripts denote an element belonging to a model, e.g. $S_{\mathcal{P}}$ for the state space of PTD $\mathcal{P}$, or $A_{\mathcal{M}}$ for the action set of MA $\mathcal{M}$. Given a set $\mathbb{P}$ of PTDs, we define the disjoint union of all their state spaces as $S_{\mathbb{P}} = \bigsqcup_{\mathcal{P} \in \mathbb{P}} S_{\mathcal{P}}$.

4 CONTEXT-AWARE TOPOLOGICAL MAP

We represent the environment as a *context-aware topological map*, where *contexts* describe interactions at nodes and edges, and PTDs model context-dependent edge durations [34]. Topological maps simplify the environment by considering the relevant locations for tasks, assuming continuous navigation.

Definition 4.1. A *topological map* is a tuple $\mathcal{T} = \langle V, E \rangle$ where V is a finite set of nodes representing locations in the environment,

and $E \subseteq V \times V$ is a set of directed edges which robots can travel on. We refer to nodes and edges as *map resources*, the set of which is given by $X = E \cup V$.

The outcome and duration of robot actions are dependent on the precise spatiotemporal situation in which they are executed. We refer to this as the context.

Definition 4.2. The set of *contexts* is given by $C = \bigcup_{x \in X} C_x$, where C_x denotes the contexts observable at map resource $x \in X$.

Contexts provide a framework for representing phenomena that affect action durations. For example, the time for a mobile robot to navigate through an office increases during the day, due to the increased presence of people [15], and so the context should consider an action's start time. In this paper, we describe the context in terms of *robot presence at map resources*, which causes robot interactions that affect action durations. An example interaction mode for mobile robots is congestion, where duration uncertainty increases as more robots traverse an edge simultaneously [34]. We capture robot presence by *counting* the robots at a map resource when an action is executed. In Section 5, we present two methods for describing the context using these robot counts. Finally, we define a context-aware topological map:

Definition 4.3. A *context-aware topological map* is a tuple $\mathcal{T}_C = \langle V, E, C, \rho_E \rangle$, where C is the set of contexts, and $\rho_E : E \times C \to \mathbb{P}$ maps an edge and context for that edge to a PTD over the duration for a robot to traverse that edge.

5 CONTEXT-AWARE MULTI-ROBOT MARKOV AUTOMATA

In this section, we present an MRMA that models robot interactions with contexts, and stochastic action durations with PTDs.

5.1 MRS Assumptions

Prior to building the MRMA, we first define our assumptions over the joint MRS state space and robot action execution.

Definition 5.1. Let $R = \{r_1, ..., r_n\}$ be a team of n robots. We represent robot r_i's local state space as $S_i^l = S_i^{l,1} \times ... \times S_i^{l,k_i}$. $S_i^{l,j}$ is a local state feature, such as the robot's battery level. We assume $S_i^{l,1} = V$, which we shorten to S_i^v, i.e. a robot's local state contains its location. We also consider a possibly empty set of global state features shared among robots, such as whether doors are open, which we denote as $S^g = S^{g,1} \times ... \times S^{g,m}$. The *joint system state space* is composed of the local and global state features, $S_J = S_1^l \times ... \times S_n^l \times S^g$.

Each robot r_i can navigate on the topological map and execute a set of non-navigation actions N_i at topological nodes, such as grasping an object or opening a door. We denote the actions r_i can execute with $A_i = (E \cup N_i)$. The complete set of non-navigation actions is denoted $N = \bigcup_i N_i$. Non-navigation action durations are defined by $\rho_N : N \times C \to \mathbb{P}$, where $\rho_N(a, c)$ returns a PTD over the duration of executing action a under context c. Our MRMA formalism allows for stochastic outcomes, as demonstrated in Section 8. However, to simplify notation we assume actions are deterministic. We assume actions can only change the robot's local state and the global state features. We define the effect(s) of an action with

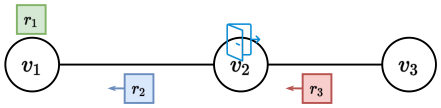


Figure 1: An MRS on a topological map, where squares represent robots and there is a door at v_2. Robot r_1 is stationary and about to navigate along topological edge (v_1, v_2).

$eff_i : S_i^l \times S^g \times A_i \to S_i^l \times S^g$, where $eff_i(s_i^l, s^g, a) = (t_i^l, t^g)$ returns the new local state for robot r_i and the updated global state features after r_i executes action a.

Example 1. Consider Fig. 1, where robot r_i's local state contains its latest topological node and battery level b_i. There is a closed door at v_2, represented in S^g as `open` or `closed`. The current joint state is $((v_1, b_1), (v_2, b_2), (v_3, b_3), \texttt{closed}) \in S_J$. Upon reaching v_2, robot r_3 opens the door. The effects of opening the door are $eff_3((v_2, b_3), \texttt{closed}, \texttt{open\_door}) = ((v_2, b_3 - \gamma), \texttt{open})$, i.e. r_3's battery is decreased by γ, and the door is now open.

5.2 Single-Robot MA

In this subsection we build a single-robot MA $\mathcal{M}_i$ for each robot r_i, which we later compose to build the MRMA. In $\mathcal{M}_i$, robot r_i chooses an action and follows exponential transitions corresponding to the PTD for that action and a context that assumes no other robot is present.

Definition 5.2. A *single-robot MA* is a tuple $\mathcal{M}_i = \langle S_{\mathcal{M}_i}, \bar{s}_{\mathcal{M}_i}, A_{\mathcal{M}_i}, \delta_{\mathcal{M}_i}, \Delta_{\mathcal{M}_i} \rangle$, where $S_{\mathcal{M}_i} = S_i^l \times S_{\mathbb{P}}^{\perp} \times S^g$, with $S_{\mathbb{P}}^{\perp} = S_{\mathbb{P}} \cup \{\perp\}$. A state $s = (s_i^l, s_i^p, s^g) \in S_{\mathcal{M}_i}$ comprises the robot's local state s_i^l, a PTD state s_i^p, and the global state feature values s^g. The PTD state tracks progress through the robot's current PTD, where $s_i^p = \perp$ means the robot is not executing an action. The initial state $\bar{s}_{\mathcal{M}_i} = (\bar{s}_i^l, \perp, \bar{s}^g)$ specifies the initial local and global state feature values, where the robot is idle. $A_{\mathcal{M}_i} = A_i$, i.e. robots can navigate along edges and execute non-navigation actions. We write $A_i(s)$ to denote the enabled actions in state $s \in S_{\mathcal{M}_i}$.

Before defining the transition functions $\delta_{\mathcal{M}_i}$ and $\Delta_{\mathcal{M}_i}$, we require additional notation. For each action and context, there is a corresponding PTD, and each PTD maps to a single action (cf. Fig. 2a). Thus, for PTD state s_i^p, we write $\mathcal{P}(s_i^p)$ to denote the PTD s_i^p belongs to, and $a(s_i^p)$ to denote the action modelled by $\mathcal{P}(s_i^p)$; we define $a(s_i^p) = \perp$ when $s_i^p = \perp$. Further, we define a function $\texttt{PTD}_i : S_{\mathcal{M}_i} \times A_{\mathcal{M}_i} \to \mathbb{P}$, which returns the PTD followed by robot r_i when executing action a in state s:

$$\texttt{PTD}_i(s, a) = \begin{cases} \rho_E(a, c_i(s, a)) & \text{if } a \in A_i(s) \cap E \\ \rho_N(a, c_i(s, a)) & \text{if } a \in A_i(s) \cap N_i, \end{cases} \tag{1}$$

where $c_i(s, a)$ is the context observed by r_i immediately before executing action a in state s. For $\mathcal{M}_i$, $c_i(s, a) = c^0, \forall s \in S_{\mathcal{M}_i}$ and $\forall a \in A_i(s)$, where c^0 denotes the absence of other robots. When composing the single-robot models into the MRMA, we will modify c_i to count the robots in locations that affect an action's duration.

$\delta_{\mathcal{M}_i}$ models instantaneous action choice. When action a is chosen in state s, the PTD state is updated to an initial state of

$\mathsf{PTD}_i(s,a)$, where the transition probability is the initial PTD state probability. Formally, for $s = (s_i^l, s_i^p, s^g)$ and $t = (t_i^l, t_i^p, t^g)$:

$$\delta_{\mathcal{M}_i}(s,a,t) = \begin{cases} init_{\mathsf{PTD}_i(s,a)}(t_i^p) & \text{if } s_i^p = \perp \text{ and} \\ & (s_i^l, s^g) = (t_i^l, t^g) \text{ and} \\ & a \in A_i(s) \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

$\Delta_{\mathcal{M}_i}$ captures the effects of the PTDs. If the robot is in PTD state s_i^p, the rate to a non-absorbing state $t_i^p \in S_{\mathcal{P}(s_i^p)}$ is defined according to $\Delta_{\mathcal{P}(s_i^p)}$. When the robot finishes a PTD, its PTD state is set to $\perp$ and its local state, as well as the global state features, are updated. The rate of this transition is the rate to the absorbing state according to $\Delta_{\mathcal{P}(s_i^p)}$. Formally, for $s = (s_i^l, s_i^p, s^g)$ and $t = (t_i^l, t_i^p, t^g)$:

$$\Delta_{\mathcal{M}_i}(s,t) = \begin{cases} \Delta_{\mathcal{P}(s_i^p)}(s_i^p, t_i^p) & \text{if } \mathcal{P}(s_i^p) = \mathcal{P}(t_i^p) \text{ and} \\ & (s_i^l, s^g) = (t_i^l, t^g) \text{ and} \\ & t_i^p \neq s^f_{\mathcal{P}(t_i^p)} \\ \Delta_{\mathcal{P}(s_i^p)}(s_i^p, s^f_{\mathcal{P}(s_i^p)}) & \text{if } t_i^p = \perp \text{ and} \\ & \mathit{eff}_i(s_i^l, s^g, a(s_i^p)) = (t_i^l, t^g) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The transition functions can be extended to handle stochastic outcomes by using a PTD for each outcome, where upon action choice there are immediate transitions to the initial states of these PTDs, weighted by the outcome probability. Upon finishing a PTD, the state update is outcome dependent. We demonstrate this in Section 8, where we integrate robot navigation failure into the MRMA.

Example 2. Consider robot r_1 in Fig. 1, where the current state in $\mathcal{M}_1$ is $s = ((v_1, b_1), \perp, \texttt{closed})$. The only enabled action for r_1 is $A_1(s) = \{(v_1, v_2)\}$, i.e. navigate along (v_1, v_2). Upon choosing action (v_1, v_2), r_1's state is updated to $((v_1, b_1), s_1^p, \texttt{closed})$, where $s_1^p \in S_{\rho_E((v_1,v_2),c^0)}$, as r_1 has begun to navigate. Robot r_1 then follows a sequence of exponential transitions until it arrives at v_2.

5.3 Multi-Robot MA

To model asynchronous multi-robot execution and interactions, we compose the single-robot MA $\mathcal{M}_i$, $i \in [\![1, n]\!]$, where $[\![i, j]\!]$ denotes the discrete interval $\{i, ..., j\}$, $j \geq i$, into an MRMA $\mathcal{M}_J$ that models the state features in S_J and each robot's PTD state. Each transition models a robot's action choice, or PTD progression. Upon action choice, we observe interactions by *counting* the robots at relevant locations in the joint state. From this, we determine the context.

Definition 5.3. A *context-aware MRMA* is a tuple $\mathcal{M}_J = \langle S_{\mathcal{M}_J}, \bar{s}_{\mathcal{M}_J}, A_{\mathcal{M}_J}, \delta_{\mathcal{M}_J}, \Delta_{\mathcal{M}_J} \rangle$, where $S_{\mathcal{M}_J} = (\bigtimes_{i \in [\![1,n]\!]} (S_i^l \times S_{\mathbb{P}}^{\perp})) \times S^g$, i.e. each robot's local and PTD state, and the global state features. The initial state is $\bar{s}_{\mathcal{M}_J} = (\bar{s}_i^l, \perp, ..., \bar{s}_n^l, \perp, \bar{s}^g)$, i.e. each robot's initial local and PTD states, and the initial global state feature values. The action set $A_{\mathcal{M}_J} = \bigcup_i A_{\mathcal{M}_i}$ is the union of each robot's action set. Let $[\cdot]_i$ project states in $S_{\mathcal{M}_J}$ onto $S_{\mathcal{M}_i}$, i.e. $[(s_1^l, s_1^p, ..., s_n^l, s_n^p, s^g)]_i = (s_i^l, s_i^p, s^g)$. Immediate transitions model a single robot's action choice, which changes only their local state:

$$\delta_{\mathcal{M}_J}(s,a,t) = \begin{cases} \delta_{\mathcal{M}_i}([s]_i, a, [t]_i) & \text{if } a \in A_{\mathcal{M}_i} \text{ and} \\ & \forall j \neq i,\ [s]_j = [t]_j \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Exponential transitions model a single robot's PTD progress, which may change only their local state and the global state features:

$$\Delta_{\mathcal{M}_J}(s,t) = \begin{cases} \Delta_{\mathcal{M}_i}([s]_i, [t]_i) & \text{if } \forall j \neq i,\ [s]_j^l = [t]_j^l \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

Recall that $c_i(s,a)$ is the context observed by robot r_i when executing action a in state s. In Equation (1), c_i ignores robot interactions. We now present two methods for redefining $c_i : S_{\mathcal{M}_J} \times A_{\mathcal{M}_J} \to C$ that consider interactions in MRMA states. Each method defines the context set C differently. However, both methods require *counting* how many other robots are at map resource x in joint state s:

$$\mathsf{cnt}_i(s,x) = \sum_{k \in [\![1,n]\!] \setminus \{i\}} \mathbb{1}[L(s_k^v, a(s_k^p)) = x]. \tag{6}$$

In the above, $\mathbb{1}$ is the indicator function and $L : V \times (E \cup N \cup \{\perp\}) \to X$ represents where an action is taking place, defined as $L(v,a) = a$ if $a = (v, v') \in E$, and $L(v,a) = v$ otherwise. In words, navigation actions take place on the edge being navigated, and non-navigation actions occur at the node the action is executed at. Robots that are idle in a node are also considered: if r_k is in a state where s_k^p is such that $a(s_k^p) = \perp$, then $L(s_k^v, a(s_k^p)) = s_k^v$.

To support the context computation, we also model spatial dependencies between map resources using *map resource groups*.

Definition 5.4. A *map resource group function* $g : X \to 2^X$ maps each map resource x to the set of map resources that affect robot action execution at x.

Example 3. Consider Fig. 1, where robot r_1 is about to navigate along edge (v_1, v_2). The presence of robot r_2 on (v_2, v_1) affects the duration of navigating on (v_1, v_2), and so $(v_2, v_1) \in g((v_1, v_2))$.

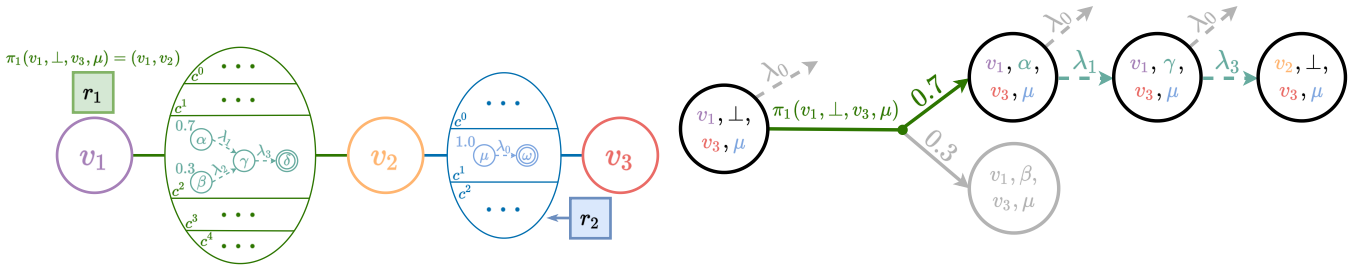
Scalar Contexts. Scalar contexts describe the context with a single number of robots, $C_x = [\![0, n-1]\!],\ \forall x \in X$. To observe the context for an action, we count the robots present anywhere in the action location's map resource group. Formally:

$$c_i(s,a) = \sum_{x \in g(L(s_i^v, a))} \mathsf{cnt}_i(s,x). \tag{7}$$

In practice, we fit PTDs for each context from empirical data (cf. Section 8). Scalar contexts provide data efficiency by aggregating multiple interaction scenarios together. However, this lacks precision, as the context is computed as if all robots are at the action location. To improve precision, we consider vectorising the context.

Vector Contexts. Vector contexts describe the context for an action as a set of (resource, #robots) pairs for each location in the action location's map resource group, $C_x = 2^{X \times [\![0, n-1]\!]},\ \forall x \in X$. Formally:

$$c_i(s,a) = \{(x, \mathsf{cnt}_i(s,x)) \mid x \in g(L(s_i^v, a))\}. \tag{8}$$



(a) An MRS which visualises the leftmost state in Fig. 2b.

(b) The simulation run. Grey transitions are enabled but not sampled.

Figure 2: A simulation run through a 2 robot MRMA $\mathcal{M}_J$. Solid arrows represent immediate transitions, dashed arrows represent exponential transitions, and concentric circles represent PTD absorbing states.

Vector contexts distinguish between different interaction scenarios more precisely than scalar contexts by counting each map resource separately. However, since there are many vector contexts per action, we require large datasets to build accurate PTDs per context. We compare scalar and vector contexts in Section 8.

Example 4. In Fig. 1, robot r_1 is about to navigate edge (v_1, v_2), and so we must observe the context. In this example, let $g((v_1, v_2)) = \{(v_1, v_2), (v_2, v_1), (v_3, v_2)\}$. Since r_2 is present on (v_2, v_1), and r_3 on (v_3, v_2), the scalar context is 2. The vector context is $(((v_1, v_2), 0), ((v_2, v_1), 1), ((v_3, v_2), 1))$.

Example 5. Consider the MRS in Fig. 2a, where each robot's local state is its current node, $S_i^l = V$, and there are no global state features. Robot r_1 is stationary, and r_2 is navigating (v_3, v_2) under context c^1. The current MRMA state is $(v_1, \perp, v_3, \mu)$ (cf. Fig. 2b), i.e. r_1 is idle at v_1, and r_2 is in PTD state $\mu \in S_{\rho_E((v_3,v_2),c^1)}$, coming from v_3.

6 CONTEXT-AWARE MULTI-ROBOT SIMULATION

In this section, we introduce CAMAS, a DES which simulates robot behaviour by sampling through an MRMA. First, we define single-robot behaviour policies, which we input to CAMAS.

Definition 6.1. A *policy* for a robot r_i is defined as $\pi_i : S_{\mathcal{M}_J} \to A_i$. $\pi_i(s)$ defines which action r_i will execute in state $s \in S_{\mathcal{M}_J}$.

Next, we describe the MRMA sampling procedure. In the MRMA, immediate transitions fire before exponential transitions. Policy π_i determines the action choice for robot r_i. In states where multiple robots need to choose an action, we apply the policy action and sample the successor state for each robot in turn. If the current state has no immediate transitions, we resolve the race condition induced by the exponential transitions. Simulation time is recorded by summing the sampled durations. The simulation termination condition is problem dependent. For example, we may terminate each simulation after a fixed time, or when a certain condition over states is satisfied.

Example 6. Consider Fig. 2, which demonstrates a run through MRMA $\mathcal{M}_J$, where each robot's local state is its current node, $S_i^l = V$. In state $(v_1, \perp, v_3, \mu)$ in Fig. 2b, robot r_1 chooses action $\pi_1(v_1, \perp, v_3, \mu) = (v_1, v_2)$. Assume that the context for (v_1, v_2) is c^2; the corresponding PTD is shown along the edge of (v_1, v_2) in Fig. 2a. The immediate transition samples the initial PTD state. From state (v_1, α, v_3, μ), r_1 wins two consecutive race conditions which progresses it along its PTD until it arrives at the new local state v_2.

7 COMPLEXITY ANALYSIS

MRMA Space Complexity. First, we consider the size of the joint system state space S_J, given by $O((\max_{i \in [\![1,n]\!]} |S_i^l|)^n \cdot |S^g|)$, which is exponential in the number of robots. Next, we consider the PTD states. Let $\mathcal{P}_{max}$ be the largest PTD across ρ_E and ρ_N. For each action enabled in a state, we need at most $|S_{\mathcal{P}_{max}}|$ states for that action. In any MRMA state, there are at most $|A_{max}| \cdot n$ enabled actions, i.e. $|A_{max}|$ actions for each robot, where A_{max} is the largest robot action set. PTD evolution in the MRMA does not affect the local or global state features, and so from each $s \in S_J$ there are at most $O(|S_{\mathcal{P}_{max}}| \cdot |A_{max}| \cdot n)$ states in the MRMA. Therefore, the MRMA size is $O((\max_{i \in [\![1,n]\!]} |S_i^l|)^n \cdot |S^g| \cdot |S_{\mathcal{P}_{max}}| \cdot |A_{max}| \cdot n)$, which remains exponential in the number of robots.

CAMAS Complexity. CAMAS employs lazy MRMA construction, only expanding states reached during simulation runs. We consider the complexity of a single MRMA simulation run. Let a_{max} be the maximum number of actions executed by a robot in a single simulation. For each action, we visit a number of PTD states upper bounded by $|S_{\mathcal{P}_{max}}|$. Therefore, a robot visits at most $O(a_{max} \cdot |S_{\mathcal{P}_{max}}|)$ states. The number of states visited per robot is unaffected by the team size, and so we visit at most $O(n \cdot a_{max} \cdot |S_{\mathcal{P}_{max}}|)$ states per simulation. At each state, we sample from a branching distribution of size n, which has worst-case complexity $O(n)$ [25]. Therefore, the complexity of CAMAS is $O(n^2 \cdot a_{max} \cdot |S_{\mathcal{P}_{max}}|)$, which is quadratic in the number of robots.

8 EXPERIMENTS

In this section, we validate the MRMA by using CAMAS to predict the behaviour of an MRS simulated in the Gazebo physics simulator [23]. If CAMAS provides accurate predictions, then the MRMA is accurately capturing high-level multi-robot behaviour. We do this for scalar and vector contexts, and compare against two baselines. Further, we compare two duration data collection methods for building the MRMA duration models in practice. Gazebo simulations are run on AWS RoboMaker. All other simulations are run

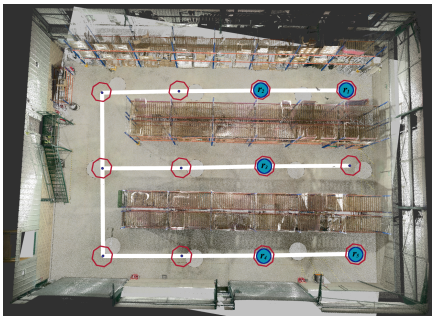


Figure 3: The warehouse environment with topological map overlaid. Blue circles show initial robot locations.

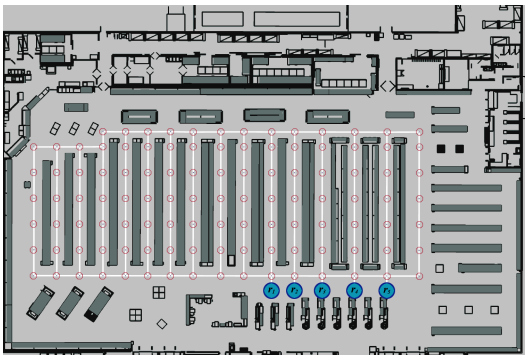


Figure 4: The supermarket environment with topological map overlaid. Blue circles show initial robot locations.

on Ubuntu 18.04, with an Intel Core i9-10900K CPU@3.7GHz and 32GB of RAM. All software is written in Python.

Experimental Environments. In our experiments, we model robot interactions caused by congestion [34]. We consider two environments: a warehouse (Fig. 3), and a supermarket (Fig. 4). The warehouse is small, forcing congestion. The larger supermarket allows us to evaluate the MRMA and CAMAS over longer horizons. Both environments are rendered in Gazebo [23], and a topological map is created for each. We observe that robots travelling on different edges to the same node often cause temporary blockages, increasing navigation durations. Therefore, we define the map resource group at an edge $(v_1, v_2) \in E$ as $g((v_1, v_2)) = \{(v_1, v_2), (v_2, v_1), v_2\} \cup \{(v', v'') \in E \mid v'' = v_2\}$, i.e. the edge itself, the reverse edge, the target node v_2, and all edges that arrive at v_2. For a node $v \in V$, $g(v) = \{v\} \cup \{(v', v'') \in E \mid v'' = v\}$, i.e. all edges that arrive at v, and v itself.

Data Collection. To capture robot dynamics, we require PTDs fitted from data that is representative of real robot execution. We use Gazebo to build action duration datasets under each context by simulating five Clearpath Jackals navigating using Move Base Flex [31] for 24 hours. Whenever a robot traverses an edge, the duration and context are recorded. We record the context when a robot starts an edge as this is when we observe the context in the MRMA. We compare two methods for data collection. The first is random navigation across the topological map. For the second, we exploit regularity within robotic environments, such as the aisles of a warehouse (cf. Fig. 3), by partitioning edges based on length, where equal length edges appear in the same subset. We then target data collection on one edge from each subset by forcing robots to follow routes containing the target edge, which forces interactions to occur. The data for a target edge is used for all edges in its subset. Random navigation observes all edges but is slow to collect data. Edge partitioning obtains a dataset quickly, but ignores differences between equal length edges, such as the floor surface or physical surroundings. We fitted PTDs from the data using the method in [36].

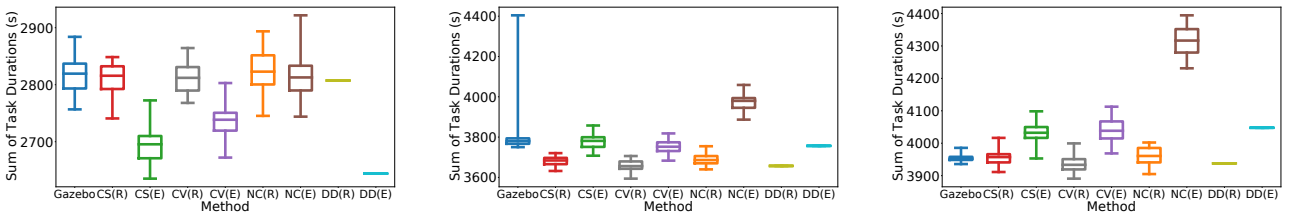
As well as collecting duration data, we also observe the probability of successful edge navigation for each context by counting the number of successes and failures. With this, we add uncertain outcomes to immediate transitions in the MRMA, and sample the outcome in CAMAS when robots choose to navigate. If a robot fails, it stops on the failed edge and executes no further actions, increasing congestion for any future robot navigating that edge.

During data collection, we may not observe all possible contexts. Further, we exclude data for contexts with less than $\theta = 10$ samples. For contexts without data, we use the PTDs and success probabilities of the nearest context for that action with sufficient data, which for scalar contexts we compute using the difference in the number of robots. The nearest vector context is computed by summing the robot differences for each map resource in the context.

Experimental Baselines. To validate the MRMA we compare CAMAS to Gazebo simulations identical to those used for data collection. We also introduce two baseline models to justify our use of contexts and stochastic action durations. Runs can be simulated through these models identically to CAMAS. The first baseline ignores action contexts. For each action, we fit a single PTD and success probability using all of the data collected for that action. Therefore, the distribution over contexts is fixed to that observed during data collection. The second baseline assumes deterministic action durations, but uses vector contexts. The duration for an action and context is the mean value observed during data collection.

Experimental Problem. In both environments, robots simulate pick and place tasks. We generate five random pick locations for each robot in each environment, which are consistent between all methods. After visiting the pick location, robots return to a fixed drop-off location, shown by the blue circles in Fig. 3 and Fig. 4. The supermarket drop-off locations are set close to each other to force congestion. For the supermarket, we also consider a variant where drop-off locations are randomised to compare the duration datasets when congestion is sparse during execution. For each method, each robot has three local state features: their location, their current task, and whether they have reached the pick location. There are no global state features. CAMAS and the baseline simulators terminate when all robots have finished their 5 tasks or stopped due to failure.

Evaluating Duration-Based Metrics. To validate that the MRMA accurately captures high-level multi-robot behaviour, we first analyse how closely CAMAS predicts task durations. We compare 40 successful Gazebo runs (i.e. no robots fail) to 40 runs of CAMAS and the baseline simulators, where all actions succeed with probability 1. In Fig. 5, we measure the sum of task durations. Further, we compare the empirical duration distributions for each pick and



(a) Warehouse. **(b) Supermarket (fixed drop-off).** **(c) Supermarket (randomised drop-off).**

Figure 5: The sum of task durations for each method.

Table 1: The error between Gazebo and each method, computed using the sum of Kolmogorov-Smirnov test results. Bold values give the lowest error for that problem setup. All values are unitless.

Method	CS(R)	CS(E)	CV(R)	CV(E)	NC(R)	NC(E)	DD(R)	DD(E)
Warehouse	8.55	12.27	**8.00**	10.27	10.60	10.75	18.30	22.68
Supermarket (fixed drop-off)	15.57	**11.80**	16.60	13.15	15.90	16.28	22.85	21.90
Supermarket (randomised drop-off)	**13.45**	15.05	13.88	15.05	14.63	21.87	22.53	24.25

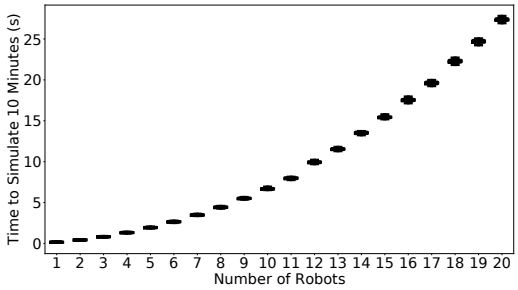


Figure 6: CAMAS scalability results.

place task. For each method and task, we compute the error against Gazebo using a two-sided Kolmogorov-Smirnov (KS) test, which computes the maximum difference between empirical cumulative distribution functions [27]. A low KS result means the distributions are similar, i.e. the method closely predicts the task duration distribution observed in Gazebo. We sum the task errors for each method/problem setup and present the results in Table 1. In all plots and tables, we write CS and CV for scalar and vector CAMAS, NC for the no context baseline, and DD for the deterministic duration baseline. We follow these with (R) or (E) to denote whether random navigation or edge partitioning was used for data collection.

A CAMAS variant gives the lowest error for each problem in Table 1, demonstrating the accuracy of the MRMA. Conversely, the deterministic duration baseline gives the highest error. The Gazebo simulations are inherently uncertain, due to environmental factors and uncertainty caused by robot interactions (cf. Fig. 5). *Though the deterministic duration baseline observes action contexts, it ignores duration uncertainty.* The no context baseline performs well when the distribution over contexts observed during data collection matches what we observe during execution. However, this doesn't generalise, as highlighted when using edge partitioning, where robots frequently travel through the target edge. This causes heavy interactions during data collection which do not occur during the pick and place tasks. *Therefore, the no context baseline inaccurately models the frequency of interactions during execution, decreasing performance.* The MRMA avoids this by using a distribution per context.

CAMAS with vector contexts gives the lowest error in the warehouse, as in small environments we can collect sufficient data for most contexts. *Vector contexts describe the spatiotemporal situation an action is executed in more precisely, which allows us to build distributions that better reflect the subtleties in different interaction scenarios.* However, CAMAS with scalar contexts produces lower errors in the supermarket problems. In large environments, we are unlikely to gain sufficient data for many vector contexts, decreasing the accuracy of our duration models. Scalar contexts improve the duration models by aggregating data, thus building larger datasets.

Random navigation produces better duration models in the warehouse. As the environment is small, we obtain large samples for each edge and context. *Further, random navigation enables contexts to occur in a natural way that is representative of real execution.* Edge partitioning creates artificial scenarios to observe high-interaction contexts, which creates less realistic datasets. However, edge partitioning improves performance for the supermarket with fixed drop-off locations, where the drop-off locations force congestion. The environment size prevents random navigation from gaining sufficient data for many observable contexts. *Though edge partitioning collects data from less realistic scenarios, it provides a better view of high-interaction contexts, which improves performance.* Random navigation produces better duration models in the supermarket with randomised drop-off locations as the data is more representative and there are few interactions, which renders much of the data collected by edge partitioning unnecessary.

Simulation Speed. To evaluate the scalability of CAMAS, we simulate 40 runs of 10 minutes for team sizes ranging from 1 to 20 robots. Robots operate in the supermarket environment, and navigate randomly during the simulation. The results in Fig. 6 demonstrate the

Table 2: Evaluation of the failure-based metrics. Bold values have the closest mean to Gazebo. Highlighted cells show methods whose distribution is not statistically significantly different to Gazebo, according to a Mann-Whitney U test with $p = 0.05$.

Predictions for the average number of tasks completed, ± standard deviation.							
Method	Gazebo	CS(R)	CS(E)	CV(R)	CV(E)	NC(R)	NC(E)
Warehouse	22.07 ± 3.52	19.93 ± 3.83	19.68 ± 4.17	**20.70 ± 3.61**	20.08 ± 4.14	20.57 ± 3.46	19.62 ± 4.23
Supermarket (fixed drop-off)	23.72 ± 2.79	16.45 ± 4.49	17.82 ± 3.21	**22.85 ± 2.14**	10.38 ± 3.13	20.77 ± 3.32	15.03 ± 4.33
Supermarket (randomised drop-off)	24.13 ± 1.72	23.52 ± 2.41	17.40 ± 3.79	**24.17 ± 1.54**	15.70 ± 4.83	23.75 ± 1.91	15.45 ± 4.95
Predictions for the average number of robots succeeded, ± standard deviation.							
Method	Gazebo	CS(R)	CS(E)	CV(R)	CV(E)	NC(R)	NC(E)
Warehouse	3.73 ± 0.98	3.53 ± 1.06	3.45 ± 1.13	3.65 ± 1.01	3.57 ± 1.09	**3.75 ± 0.83**	3.37 ± 1.17
Supermarket (fixed drop-off)	4.40 ± 1.17	1.92 ± 1.42	2.75 ± 0.87	**4.22 ± 0.71**	0.63 ± 0.68	3.40 ± 1.11	1.98 ± 1.18
Supermarket (randomised drop-off)	4.63 ± 0.71	4.40 ± 0.88	2.67 ± 1.07	**4.68 ± 0.53**	2.4 ± 1.16	4.53 ± 0.64	2.23 ± 1.19
Predictions for the probability of a run being successful.							
Method	Gazebo	CS(R)	CS(E)	CV(R)	CV(E)	NC(R)	NC(E)
Warehouse	0.283	0.183	0.200	**0.233**	0.217	0.167	0.150
Supermarket (fixed drop-off)	0.717	0.067	0.017	**0.383**	0.000	0.200	0.017
Supermarket (randomised drop-off)	0.767	0.617	0.033	**0.717**	0.033	0.600	0.033

quadratic complexity of CAMAS. *However, for* 20 *robots, CAMAS still runs* 22 *times faster than real-time, validating it as a scalable DES.*

Evaluating Failure-Based Metrics. Next, we validate the MRMA accuracy on problems with uncertain outcomes by evaluating CAMAS against metrics concerning navigation failure. All methods use the success probabilities obtained during data collection. We carry out 60 simulations of Gazebo, CAMAS and the baseline simulators, where navigation may fail, and evaluate 3 metrics:

(1) *The average number of tasks completed.* A task is completed if a robot does not fail during its execution.
(2) *The average number of robots succeeded.* A robot succeeds if it completes all 5 of its tasks.
(3) *The probability of a run being successful.* A run is successful if all 5 robots complete all of their tasks without failure.

We provide results for these metrics in Table 2. We omit results for the deterministic duration baseline as failures are modelled identically to the MRMA with vector contexts. *CAMAS with vector contexts under the random navigation dataset produces the closest Gazebo predictions, except for the average robots succeeded in the warehouse.* In this case, the no context baseline provides the closest prediction, as the observed distribution over contexts during data collection is close to that observed during the pick and place tasks.

Edge partitioning leads to poorer predictions of navigation failure, as the failure probabilities of the target edges are shared amongst all equal length edges, which may not hold in practice. Even if this probability is small, if robots execute long routes the probability of sampling a failure quickly increases.

For the same dataset, vector contexts typically lead to better predictions than scalar contexts. *Vector contexts provide a more precise view of interactions than scalar contexts, which aggregate multiple vector contexts.* As a result, scalar contexts generalise failure probabilities across all vector contexts it encapsulates. Though data aggregation in the scalar context will decrease this probability, it results in less accurate models of navigation failure.

Comparing the empirical distributions for the metrics in Table 2, CAMAS is not always statistically similar to Gazebo, as modelling assumptions in the MRMA prevent us from perfectly capturing multi-robot behaviour. However, the MRMA still allows us to closely predict the values in Table 2, while providing faster simulations.

In summary, the combined use of contexts and stochastic duration models in the MRMA accurately captures high-level multi-robot behaviour. Vector contexts precisely discriminate between different interaction situations. However, if limited data is available, scalar contexts can provide better task duration estimates. For duration-based metrics, edge partitioning is beneficial in large environments with frequent robot interactions. However, problems arise when generalising uncertain action outcomes. For smaller environments, or those with fewer interactions, random navigation collects more realistic execution data, improving performance.

9 CONCLUSION

In this paper we have presented an MRMA formulation that captures asynchronous multi-robot execution in continuous time. We capture robot dynamics with PTDs, and model spatial interactions between robots. Our model is the first to explicitly consider how robot interactions affect action durations. Moreover, we presented CAMAS, a scalable DES that samples through the MRMA to accurately evaluate high-level multi-robot properties. Though we have focused on robotics, the MRMA and CAMAS are applicable to any multi-agent system. In future work, we will use the MRMA as a model-based surrogate for reinforcement learning. Further, we will apply statistical model checking techniques to the MRMA, and consider generalising PTDs across different environments.

ACKNOWLEDGMENTS

This work is supported by the Honda Research Institute Europe GmbH, UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) hub [EP/R026084/1], the EPSRC Programme Grant 'From Sensing to Collaboration' [EP/V000748/1], and a gift from Amazon Web Services.

REFERENCES

[1] Carlos Azevedo, Bruno Lacerda, Nick Hawes, and Pedro Lima. 2020. Long-Run Multi-Robot Planning for Persistent Tasks. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[2] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

[3] Tim Bakker, Garrett L Ward, Siva T Patibandla, and Robert H Klenke. 2013. RAMS: A Fast, Low-Fidelity, Multiple Agent Discrete-Event Simulator. In *Proceedings of the 2013 Summer Computer Simulation Conference (SCSC)*. 1–10.

[4] Raphen Becker, Shlomo Zilberstein, Victor Lesser, and Claudia V Goldman. 2003. Transition-Independent Decentralized Markov Decision Processes. In *Proceedings of the 2nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 41–48.

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

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

[7] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. 2010. Multi-Agent Reinforcement Learning: An Overview. *Innovations in Multi-Agent Systems and Applications-1* (2010), 183–221.

[8] Yuliya Butkova, Arnd Hartmanns, and Holger Hermanns. 2021. A Modest Approach to Markov Automata. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 31, 3 (2021), 1–34.

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

[10] Hugo Costelha and Pedro Lima. 2012. Robot Task Plan Representation by Petri Nets; Modelling, Identification, Analysis and Execution. *Autonomous Robots* 33, 4 (2012), 337–360.

[11] Bruno Damas and Pedro Lima. 2004. Stochastic Discrete Event Model of a Multi-Robot Team Playing an Adversarial Game. *IFAC Proceedings Volumes* 37, 8 (2004), 974–979.

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

[13] Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. 2013. A Semantics for Every GSPN. In *Proceedings of the 34th International Conference on Applications and Theory of Petri Nets and Concurrency (Petri Nets)*. Springer.

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

[15] Jaime Pulido Fentanes, Bruno Lacerda, Tomáš Krajník, Nick Hawes, and Marc Hanheide. 2015. Now or Later? Predicting and Maximising Success of Navigation Actions from Long-Term Experience. In *Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1112–1117.

[16] Fei Gao and Missy Cummings. 2012. Using Discrete Event Simulation to Model Multi-Robot Multi-Operator Teamwork. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, Vol. 56. SAGE Publications Sage CA: Los Angeles, CA, 2093–2097.

[17] Xianping Guo and Onésimo Hernández-Lerma. 2009. *Continuous-Time Markov Decision Processes: Theory and Applications*. Springer-Verlag Berlin Heidelberg.

[18] David Ha and Jürgen Schmidhuber. 2018. Recurrent World Models Facilitate Policy Evolution. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS)*. 2455–2467.

[19] Jens Happe and Jean Berger. 2010. CoUAV: A Multi-UAV Cooperative Search Path Planning Simulation Environment. In *Proceedings of the 2010 Summer Computer Simulation Conference (SCSC)*. 86–93.

[20] Benjamin Herd, Simon Miles, Peter McBurney, and Michael Luck. 2015. Quantitative Analysis of Multiagent Systems through Statistical Model Checking. In *Proceedings of the International Workshop on Engineering Multi-Agent Systems*. Springer, 109–130.

[21] Jemin Hwangbo, Joonho Lee, and Marco Hutter. 2018. Per-Contact Iteration Method for Solving Contact Dynamics. *IEEE Robotics and Automation Letters* 3, 2 (2018), 895–902.

[22] Maria Hybinette, Eileen Kraemer, Yin Xiong, Glenn Matthews, and Jaim Ahmed. 2006. SASSY: A Design for a Scalable Agent-Based Simulation System using a Distributed Discrete Event Infrastructure. In *Proceedings of the 2006 Winter Simulation Conference*. IEEE, 926–933.

[23] Nathan Koenig and Andrew Howard. 2004. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sendai, Japan, 2149–2154.

[24] Axel Legay, Benoît Delahaye, and Saddek Bensalem. 2010. Statistical Model Checking: An Overview. In *Proceedings of the International Conference on Runtime Verification*. 122–135.

[25] Adam Lipowski and Dorota Lipowska. 2012. Roulette-Wheel Selection via Stochastic Acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.

[26] Masoumeh Mansouri, Bruno Lacerda, Nick Hawes, and Federico Pecora. 2019. Multi-Robot Planning Under Uncertain Travel Times and Safety Constraints. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), August 10-16, Macao, China*. 478–484.

[27] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov Test for Goodness of Fit. *J. Amer. Statist. Assoc.* 46, 253 (1951), 68–78.

[28] João Vicente Messias, Matthijs Spaan, and Pedro Lima. 2013. GSMDPs for Multi-Robot Sequential Decision-Making. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*. 1408–1414.

[29] Olivier Michel. 2004. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. *International Journal of Advanced Robotic Systems* 1, 1 (2004), 5.

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

[31] Sebastian Pütz, Jorge Santos Simón, and Joachim Hertzberg. 2018. 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)*. 3416–3421.

[32] E. Rohmer, S. P. N. Singh, and M. Freese. 2013. CoppeliaSim (formerly V-REP): A Versatile and Scalable Robot Simulation Framework. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[33] Charlie Street, Bruno Lacerda, Manuel Mühlig, and Nick Hawes. 2020. Multi-Robot Planning Under Uncertainty with Congestion-Aware Models. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[34] Charlie Street, Sebastian Pütz, Manuel Mühlig, Nick Hawes, and Bruno Lacerda. 2021. Congestion-Aware Policy Synthesis for Multirobot Systems. *IEEE Transactions on Robotics* (2021).

[35] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. *Artificial Intelligence* 112, 1-2 (1999), 181–211.

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

[37] Håkan LS Younes and Reid G Simmons. 2004. Solving Generalized Semi-Markov Decision Processes using Continuous Phase-Type Distributions. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence*. 742–747.