# Optimizing Robotic Team Performance with Probabilistic Model Checking[⋆]

Sagar Chaki, Joseph Giampapa, David Kyle, and John Lehoczky

Carnegie Mellon University, Pittsburgh, PA, USA

**Abstract.** We present an approach to analytically construct a robotic team, i.e., team members and deployment order, that achieves a specific task with quantified probability of success. We assume that each robot is Markovian, and that robots interact with each other via communication only. Our approach is based on probabilistic model checking (PMC). We first construct a set of Discrete Time Markov Chains (DTMCs) that each capture a specific "projection" of the behavior of an individual robot. Next, given a specific team, we construct the DTMC for its behavior by combining the projection DTMCs appropriately. Finally, we use PMC to evaluate the performance of the team. This procedure is repeated for multiple teams, the best one is selected. In practice, the projection DTMCs are constructed by observing the behavior of individual robots a finite number of times, which introduces an error in our results. We present an approach – based on sampling using the Dirichlet distribution – to quantify this error. We prove the correctness of our approach formally, and also validate it empirically on a mine detection task by a team of communicating Kilobots.

## 1 Introduction

Autonomous robots are increasingly being used in teams to communicate and achieve tasks in a collaborative manner. Given a collection of robots and a specific mission, a designer solves the coalition formation problem and selects the coordination strategy so as to maximize the chances of mission success. Currently this is done in an ad-hoc manner since navigating the solution space and selecting the best one manually is impossible. This is true even if the designer is able to observe each robot individually to construct a model of its behavior. First, it is not clear which modeling formalism to use. Second, since these models are complex (if they are to be precise) it is impossible to compose them manually to make predictions about the overall behavior of a robotic team.

In this paper, we present an analytic approach to solve a simplified but common version of this problem. Specifically, we assume robots are Markovian, and

[⋆] This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution. DM-0001326.

only influence each other via communication (i.e., no physical interaction such as collisions). We call this a communicating multi-robot mission (CMRM). Given a CMRM $S$ and a deadline $D$, our approach computes the following class of properties: (i) probability of an event $e$ happening when $S$ is executed up to time $D$; and (ii) expected value of an attribute $a$ of $S$ when it is executed up to time $D$. We make the following contributions.

First, we formalize a CMRM as a *modal* Discrete Time Markov Chain (DTMC). A modal DTMC consists of a finite set of component DTMCs. Each component DTMC corresponds to a robot, and engages alternately in two kinds of moves: (i) deterministic – the DTMC's state changes instantly according to a "mode-change" function that depends on the current states of the other DTMCs; and (ii) probabilistic – according to its own transition relation; this happens synchronously with the other DTMCs and takes unit time. We show (see Theorem 1) that a modal DTMC is semantically equivalent to a specific combination of the "projections" of its component DTMCs. This result enables us to compute properties of modal DTMCs, and therefore a CMRMs, using existing probabilistic model checkers, such as PRISM [9], that verify DTMCs.

Second, we present an approach to quantify the error in our predictions. In practice, each projection DTMC is constructed by observing and measuring a finite set of runs of the corresponding robot. This means that the DTMC differs from the true DTMC for the robot's behavior. Therefore, predictions based on them lie within an error margin of the correct values. We present an approach that quantifies this error and estimates the correct property value. Specifically, we sample a set of DTMCs "around" the constructed projection, and make prediction using each sample. From these predictions, we use statistical theory to estimate the real property value and the error margin. A key aspect of our sampling procedure is its use of the Dirichlet distribution [7]. To our knowledge, this is a new approach for error estimation in the DTMC context.

Finally, we implement our approach and evaluate it using a team of Kilobots [11]. To construct the projection DTMC for a Kilobot, we: (i) run it and record its actual behavior; (ii) reproduce this behavior in the V-REP [1] simulator using manually tuned parameters; and (iii) construct the projection DTMC from measurements of multiple simulation runs. Using a simulator enables us to perform many runs and lower our error margins. At the same time, tuning the simulation parameters to replicate observed robot behavior grounds our results in reality. We show that our approach yields accurate predictions that match observed results with Kilobot teams. Further details are presented in Section 4.

*Related Work.* Konur et al. [8] model coordinated robotic behavior in the context of swarms. Like us, they compose individual models of robots into a model of team behavior. However, they assume that all robots have the same behavioral characteristics. This allows them to produce a team model which operates on counts of robots in each state, instead of tracking each robot individually. We do not assume homogeneous robots. Their attempt to track robots individually met untenable state explosion past a team size of 3. Our individual models are significantly more complex, but still due to the use of projections, we were able to verify teams of up to 6 robots.

Ghorbal et al. [4] present an approach for predicting intervals of result probabilities based on intervals of transition probabilities. This addresses the issue of error propagation, but assumes that a true range of probabilities for each transition is known with certainty. Effectively, this is a 100% confidence interval, which is unrealistic . Their approach is fully analytic, while we rely on sampling. They develop a new verification algorithm which is validated on a 21 state model, while we use existing tools and handle systems with thousands of states.

This paper builds on a wide body of work in modeling and verifying probabilistic systems [12]. In particular, probabilistic model checking has been used to verify systems ranging from pacemakers [2], root contention protocols [10] and biological pathways [6]. We extend the application of probabilistic model checking to yet another domain – communicating autonomous multi-robot missions.

The rest of this paper is organized as follows. In Section 2 we present our approach to predict properties of modal DTMCs by combining projections. In Section 3, we present our approach for quantifying error. In Section 4, we present our experimental results, and in Section 5, we conclude.

## 2 Modal DTMC and Verification

In this section we define modal DTMCs and present an algorithm to compute their properties. We begin with preliminary notation and concepts. Given a set $X$, a probability density function (PDF) over $X$ is a mapping $\pi : X \mapsto [0,1]$ such that: $\sum_{x \in X} \pi(x) = 1$. The set of all PDFs over $X$ is denoted by $\Pi(X)$. Given two sets $X_1$ and $X_2$, and PDFs $\pi_1 \in \Pi(X_1)$ and $\pi_2 \in \Pi(X_2)$, the joint PDF $\pi_1 \otimes \pi_2 \in \Pi(X_1 \times X_2)$ is defined as follows:

$$\forall (x_1, x_2) \in X_1 \times X_2 \centerdot (\pi_1 \otimes \pi_2)(x_1, x_2) = \pi_1(x_1) \times \pi_2(x_2)$$

A DTMC is a triple $(S, I, R)$ where: (i) $S$ is a finite set of states; (ii) $I \in S$ is the initial state; and (iii) $R : S \mapsto \Pi(S)$ is the transition probability matrix.

We use Probabilistic Computation Tree Logic (PCTL) [5] to express properties. For a PCTL formula $\varphi$, and a DTMC $S$, $S \models \varphi$ is the probability that $S$ satisfies $\varphi$. For example, if $\varphi = \mathsf{F}(p \vee q)$, then $S \models \varphi$ is the probability that the DTMC eventually reaches a state where either $p$ or $q$ holds, where $p$ and $q$ are propositions that are either TRUE or FALSE in each state. For DTMCs $S_1, S_2$ we write $S_1 \equiv S_2$ to mean that for every PCTL formula $\varphi$, $(S_1 \models \varphi) = (S_2 \models \varphi)$, i.e., $S_1$ and $S_2$ satisfy all PCTL formulas with equal probability.

### 2.1 Modal DTMC

A $n$-component modal DTMC is a $2n$-tuple $(M_1, \dots, M_n, \delta_1, \dots, \delta_n)$ where $M_i = (S_i, I_i, R_i)$ are DTMCs, and:

$$\delta_i : S_i \mapsto 2^{S_1 \times \cdots \times S_{i-1} \times S_{i+1} \times \cdots \times S_n} \times S_i$$

are "mode transition" functions. Informally $\delta_i(s_i) = (E, \bar{s_i})$ means that if DTMC $M_i$ is in state $s_i$, and the other DTMCs are in state $e = (s_1, \dots, s_{i-1}, s_{i+1}, \dots s_n)$, then either (i) $e \in E$ and $M_i$ makes a mode change by moving to state $\bar{s_i}$; or (ii) $e \notin E$ and $M_i$ remains in state $s_i$.

Formally, the semantics of the modal DTMC $P = (M_1, \dots, M_n, \delta_1, \dots, \delta_n)$, denoted $[\![P]\!]$, is the DTMC $(\tilde{S}, \tilde{I}, \tilde{R})$ where: (i) $\tilde{S} = S_1 \times \cdots \times S_n$; (ii) $\tilde{I} = (I_1, \dots, I_n)$; and (iii) $\tilde{R} : \tilde{S} \mapsto \Pi(\tilde{S})$ is defined as:

$$\tilde{R}(s_1, \dots, s_n) = R_1(s'_1) \otimes \cdots \otimes R_n(s'_n)$$

where for all $i \in [1, n]$, if $\delta_i(s_i) = (E, \bar{s_i})$ then:

$$(s_1, \dots, s_{i-1}, s_{i+1}, \dots s_n) \in E \wedge (s'_i = \bar{s_i}) \bigvee$$
$$(s_1, \dots, s_{i-1}, s_{i+1}, \dots s_n) \notin E \wedge (s'_i = s_i)$$

Note that, in the definition of $\tilde{R}(s_1, \dots, s_n)$ above, state $(s'_1, \dots, s'_n)$ denotes the result of (instantaneous) mode change due to exchange of information between the component DTMCs. This is followed by simultaneous probabilistic transition made by each component DTMC (which requires one unit of time), as denoted by the application of $R_1, \dots, R_n$ to states $s'_1, \dots, s'_n$, respectively, and composing the resulting PDFs via the $\otimes$ operator.

In the rest of the paper, for simplicity of explanation, we consider only a 2-component modal DTMC $P$, i.e., $P = (M_1, M_2, \delta_1, \delta_2)$. The generalization to an arbitrary (but finite) number of components is done in a natural manner. Our overall goal is to verify a PCTL formula $\varphi$ over $P$, the topic of Section 2.2.

## 2.2 Modal DTMC Verification

For any set $X$, and $x \in X$, $\Delta(x) \in \Pi(X)$ is the PDF that maps $x$ to 1 and all other elements of $X$ to 0. Recall that our target modal DTMC is $P = (M_1, M_2, \delta_1, \delta_2)$. We verify $P$ by constructing a model based on observing $M_1$ and $M_2$ individually. Our approach relies on several key ideas:

1. A state of a component DTMC (i.e., $M_1$ and $M_2$) records at least the current time and the time at which the last mode change happened. Thus, each state is of the form $(t, m, d)$ where $t$ is the current time, $m$ is the time of mode change ($m = \infty$ means that mode change has not happened yet), and $d$ is the remaining state information. Thus, the initial state is of the form $(0, \infty, d)$. Also, we know that $m \neq t$ since up to the point of mode change $m = \infty \neq t$, and after the mode change $m < t$.
2. On any execution of the DTMC, a mode change happens at most once, and is instantaneous. Thus,
$$\delta_i(t, m, d) = (E, (\bar{t}, \bar{m}, \bar{d})) \Rightarrow (m = \infty) \wedge (\bar{t} = \bar{m} = t)$$
3. The system is time-bounded, i.e., there is some time $T \geq 0$ at which the system stutters. In terms of the transition relation $R$, this means that if $s = (T, m, d)$ for some $m$ and $d$, then $R(s) = \Delta(s)$. For our experiments, the time bound equals the deadline specified in the property.

4. During our individual observations, we can change the mode of $M_i$ at arbitrary time points. This is because a mode change is controlled via software, which we are able to reprogram.

*Approach.* Our overall approach is as follows:

– Let DTMC $\langle M_i, t\rangle$ be the projection of $M_i$ under the restriction that mode change always happens at time $t$. Construct all projections $\{\langle M_1, t\rangle \mid 0 \leq t \leq T\}$ and $\{\langle M_2, t\rangle \mid 0 \leq t \leq T\}$ for $M_1$ and $M_2$, respectively.
– Construct a DTMC $\widehat{M}$ by "re-combining" the projection DTMCs using the definitions of the mode change functions $\delta_1, \dots, \delta_n$. Prove that $\widehat{M} = [\![P]\!]$.
– Compute $\widehat{M} \models \varphi$ using a probabilistic model checker, e.g., PRISM [9].

## 2.3 DTMC Projection

We now define the projection of a DTMC based on mode change time. First define function $\widehat{\delta_i} : S_i \mapsto S_i$ as follows:

$$\forall s_i \in S_i \centerdot \widehat{\delta_i}(s_i) = \bar{s_i} \centerdot \exists E \centerdot \delta_i(s_i) = (E, \bar{s_i})$$

Thus, $\widehat{\delta_i}$ is the projection of $\delta_i$ on the second component of its range, and is well-defined. Let $M_i = (S_i, I_i, R_i)$. Then the projection of $M_i$ under the restriction that mode change always happens at time $\mathbf{t}$ (where $0 \leq \mathbf{t} < T$) is the DTMC $\langle M_i, \mathbf{t}\rangle = (S_i, I_i, \langle R_i, \mathbf{t}\rangle)$ such that $\forall s = (t, m, d) \in S_i$:

$$\langle R_i, \mathbf{t}\rangle(s) = \begin{cases} R_i(\widehat{\delta_i}(s)) & \text{if } t = \mathbf{t} \\ R_i(s) & \text{otherwise} \end{cases}$$

## 2.4 Combining Projections

Consider the projections $\{\langle M_1, t\rangle \mid t \in [0, T]\}$ and $\{\langle M_2, t\rangle \mid t \in [0, T]\}$ of $M_1$ and $M_2$. Define the DTMC $\widehat{M} = (\widehat{S}, \widehat{I}, \widehat{R})$ as follows:

$$\widehat{S} = S_1 \times S_2 \qquad \widehat{I} = (I_1, I_2) \qquad \text{and}$$
$$\forall s_1 \in S_1 \centerdot \forall s_2 \in S_2 \centerdot s_1 = (t_1, m_1, d_1) \wedge s_2 = (t_2, m_2, d_2) \wedge$$
$$\delta_1(s_1) = (E_1, \bar{s}_1) \wedge \delta_2(s_2) = (E_2, \bar{s}_2) \Rightarrow \widehat{R}(s_1, s_2) = A_1 \otimes A_2 \quad \text{where}$$
$$A_1 = \begin{cases} \langle R_1, t_1\rangle(s_1) & \text{if } s_2 \in E_1 \\ \langle R_1, m_1\rangle(s_1) & \text{otherwise} \end{cases} \qquad A_2 = \begin{cases} \langle R_2, t_2\rangle(s_2) & \text{if } s_1 \in E_2 \\ \langle R_2, m_2\rangle(s_2) & \text{otherwise} \end{cases}$$

We now present our main result, Theorem 1, which states that the combination of projection DTMCs described above is equivalent to the modal DTMC.

**Theorem 1.** $\widehat{M} = [\![P]\!]$.

*Proof.* Recall that $\widehat{M} = (\widehat{S}, \widehat{I}, \widehat{R})$ and $[\![P]\!] = (\tilde{S}, \tilde{I}, \tilde{R})$. By definition, we already know that: (i) $\widehat{S} = \tilde{S} = S_1 \times S_2$ and $\widehat{I} = \tilde{I} = (I_1, I_2)$. Hence it suffices to show that $\widehat{R} = \tilde{R}$.

Let $s_1 = (t_1, m_1, d_1)$ and $s_2 = (t_2, m_2, d_2)$. Then $\tilde{R}(s_1, s_2) = R_1(s'_1) \otimes R_2(s'_2)$. First we show that $R_1(s'_1) = A_1$. Let $\delta_1(s_1) = (E_1, \bar{s}_1)$.

Case 1: $s_2 \in E_1$. In this case, $s'_1 = \bar{s}_1 = \widehat{\delta}_1(s_1)$. Then $R_1(s'_1) = R_1(\widehat{\delta}_1(s_1)) = \langle R_1, t_1\rangle(s_1) = A_1$.

Case 2: $s_2 \notin E_2$. In this case, $s'_1 = s_1$. Recall that $m_1 \neq t_1$. Hence, $\langle R_1, m_1\rangle(s_1) = R_1(s_1)$. Hence, $R_1(s'_1) = R_1(s_1) = \langle R_1, m_1\rangle(s_1) = A_1$.

In a symmetric manner, we can show that $R_2(s'_2) = A_2$. Thus, $\tilde{R}(s_1, s_2) = R_1(s'_1) \otimes R_2(s'_2) = A_1 \otimes A_2 = \widehat{R}(s_1, s_2)$.

Theorem 1 enables us to compute a property of the modal DTMC $P$ by combining projections of its components to construct DTMC $\widehat{M}$, and then applying probabilistic model checking. In practice, a projection $\langle M_i, t\rangle$ is constructed from observations of multiple runs of the corresponding robot. This inevitably introduces an error in our results. The next section presents our approach to quantify and bound this error.

## 3 Error Quantification

Suppose we model a real-world system $G$, e.g., a robot, using a DTMC $\widehat{M}$. For a given PCTL formula $\varphi$, let $\widehat{p} = (\widehat{M} \models \varphi)$, i.e., the probability that $\widehat{M}$ satisfies $\varphi$, and let $p = (G \models \varphi)$. If the model is perfect, i.e., $\widehat{M} = G$, then $\widehat{p} = p$. However, such perfect modeling is impracticable for several reasons. First, $G$ might not be Markovian. Second, suppose $G$ is Markovian, i.e., $G \equiv \boldsymbol{M}_\diamond$ for some DTMC $\boldsymbol{M}_\diamond$. Note that, in this case, $(\boldsymbol{M}_\diamond \models \varphi) = (G \models \varphi) = p$. However, $\widehat{M}$ might not capture enough *state* to accurately model $G$; i.e., there could be hidden variables in $G$ resulting in behaviors that are absent in $\widehat{M}$. Finally, $\widehat{M}$ might have the same states as $\boldsymbol{M}_\diamond$, but different *transition probabilities*, and hence diverges from $G$.

In practice, we should expect transition probabilities to be imprecise. Ultimately, the only approaches to obtain these probabilities for a real-world system are based on intuitive guesses or finite observations of the system. Thus, we should also expect any predictions made by a model to deviate from reality by some margin, where that margin is related to the uncertainty of those transition probabilities. This is the error we seek to quantify.

### 3.1 Constructing an Approximation

Recall that there exists a DTMC $\boldsymbol{M}_\diamond = (S, I, R)$ such that $\boldsymbol{M}_\diamond \equiv G$. Suppose that each state of $\boldsymbol{M}_\diamond$ corresponds to a known combination of observable characteristics in $G$. Thus, if we execute a trial of $G$, and observe it at each discrete time point, then from each observation we can compute the corresponding state of $\boldsymbol{M}_\diamond$. Suppose we execute several trials of $G$, and record our observations as a list of evidence $E$, where each evidence $e \in E$ is a sequence of states $\langle s_0, s_1, \ldots, s_k\rangle$ of $\boldsymbol{M}_\diamond$ corresponding to observations of a trial of $G$ at discrete time points.

Using $E$, we construct a transition probability matrix $\widehat{R} : S \mapsto \Pi(S)$ as follows. Given a sequence of states $e$, and states $s, \bar{s}$, let $e(s)$ and $e(s, \bar{s})$ denote, respectively, the number of times $\langle s \rangle$ and $\langle s, \bar{s} \rangle$ appear as a subsequence of $e$. We generalize this to $E$ as follows: $E(s) = \sum_{e \in E} e(s)$ and $E(s, \bar{s}) = \sum_{e \in E} e(s, \bar{s})$. Thus $E(s)$ is the number of times we observe $G$ to reach state $s$, and $E(s, \bar{s})$ is the number of times we observe $G$ to move from $s$ to $\bar{s}$ in one time step.

Then, we have two cases: (i) if $E(s) = 0$, then $\widehat{R}(s) = \Delta(s)$; (ii) otherwise $\forall \bar{s} \in S \centerdot \widehat{R}(s)(\bar{s}) = \frac{E(s,\bar{s})}{E(s)}$. Thus, for states not observed in our trials (case-i) we assume self-transitions. For other states (case-ii) we use the "frequentist" approach. Note that $\widehat{R}$ is a well-defined PDF, and provides the best possible approximation of $R$ given the available evidence $E$. Then, our approximate DTMC is $\widehat{M} = (S, I, \widehat{R})$. Note that $\widehat{M}$ deviates from $\boldsymbol{M}_\diamond$ only in its transition probabilities.

### 3.2 Distribution Definitions

The construction of $\widehat{M}$ described in the previous section does not provide any insight into how $\widehat{M} \models \varphi$ relates to $\boldsymbol{M}_\diamond \models \varphi$ (and thus $G \models \varphi$) in terms of error. Note that each possible transition $(s, \bar{s})$ can be viewed as a Bernoulli trial. Thus, statistical methods allow us to estimate the error of each individual transition probability of $\widehat{M}$. However, understanding how these errors – i.e., the difference between $R$ and $\widehat{R}$ – affect error in $\widehat{M} \models \varphi$ is not straightforward.

Let $\mathbb{M}$ be the set of all DTMCs of the form $(S, I, \tilde{R})$, i.e., $\mathbb{M}$ is the set of all DTMCs that have the same states and initial state as $\boldsymbol{M}_\diamond$. Given a real number $r \in [0,1]$, let $\boldsymbol{M}_r = \{M \in \mathbb{M} \mid (M \models \varphi) \leq r\}$ be the set of DTCMs that satisfy $\varphi$ with probability at most $r$. Let $\mathscr{M} = \{\boldsymbol{M}_r \mid r \in [0,1] \wedge \forall r' \in [0,1] \centerdot r < r' \implies \boldsymbol{M}_r \subset \boldsymbol{M}_{r'}\}$. Note that the elements of $\mathscr{M}$ are strictly ordered by size.

Given evidence $E$ from trials of $G$, we define the PDF $\widehat{\mathcal{M}_E} \in \Pi(\mathscr{M})$ by the following cumulative density function (CDF):

$$\mathit{CDF}(\widehat{\mathcal{M}_E})(\boldsymbol{M}) = \mathsf{P}(\boldsymbol{M}_\diamond \in \boldsymbol{M} | E) \tag{1}$$

That is, the CDF of $\widehat{\mathcal{M}_E}$ maps each set of DTMCs $\boldsymbol{M} \in \mathscr{M}$ to the probability that some DTMC in $\boldsymbol{M}$ is "correct", and thus equivalent to $\boldsymbol{M}_\diamond$ and $G$, given the evidence $E$. Let $\mathrm{dom}(\mathscr{M}) = \{r \in [0,1] \mid \boldsymbol{M}_r \in \mathscr{M}\}$. Next, given $\widehat{\mathcal{M}_E}$, define a PDF $\widehat{\mathcal{P}_E} \in \Pi(\mathrm{dom}(\mathscr{M}))$ by the following CDF:

$$\mathit{CDF}(\widehat{\mathcal{P}_E})(r) = \mathsf{P}((\boldsymbol{M}_\diamond \models \varphi) \leq r \mid E) \tag{2}$$

Thus, the CDF of $\widehat{\mathcal{P}_E}$ maps $r$ to the probability that $G$ satisfies $\varphi$ with probability at most $r$ given evidence $E$. Note that, from (1) and (2), we have:

$$\mathit{CDF}(\widehat{\mathcal{P}_E})(r) = \mathsf{P}(\boldsymbol{M}_\diamond \in \boldsymbol{M}_r \mid E) = \mathit{CDF}(\widehat{\mathcal{M}_E})(\boldsymbol{M}_r) \tag{3}$$

We now show how to construct $\widehat{\mathcal{M}_E}$ and $\widehat{\mathcal{P}_E}$.

## 3.3 Constructing Distributions

To construct $\widehat{\mathcal{M}_E}$, we define a mapping $T : S \mapsto \Pi(\Pi(S))$ from states to a *PDF over PDFs* of other states to transition to. We use the Dirichlet distribution since it produces sets of variates that sum to one (i.e., a PDF), and it is a conjugate prior for the Multinomial distribution [7]. We define $T$ in several steps. First, we define a prior $T^0$ to $T$ as:

$$\forall s \in S \centerdot \forall \pi \in \Pi(S) : T^0(s)(\pi) = \mathsf{P}(\pi|\boldsymbol{\alpha_s}) = \text{Dirichlet}(\pi|\boldsymbol{\alpha_s})$$

where $\boldsymbol{\alpha_s}$ is a vector of pseudo-counts of prior belief in transition likelihood of transitioning from $s$ to each state in $S$. Thus, for each $s \in S$, $T^0(s)$ is a Dirichlet distribution with parameters $\boldsymbol{\alpha_s}$. In our implementation, we used the union of the individual robots' DTMCs created during unit testing as the prior.

Next, recall that $E(s, \bar{s})$ is the number of times we observe $G$ to move from state $s$ to $\bar{s}$ in one time step during our trials. Also, $R(s)(\bar{s})$ is the probability of transitioning from $s$ to $\bar{s}$ in $\boldsymbol{M_\diamond}$. Let $\boldsymbol{c_s} = (\forall \bar{s} \in S : E(s, \bar{s}))$ be the vector of counts of transitions from $s$ to each $\bar{s}$. Since $\boldsymbol{M_\diamond}$ perfectly models $G$, we have:

$$\forall s \in S : \boldsymbol{c_s} \sim \text{Multinomial}(\forall \bar{s} \in S : R(s)(\bar{s}))$$

That is, $\boldsymbol{c_s}$ is drawn from a Multinomial distribution, which the true distribution for state $s$ in $G$. Finally, given the well known relationship between Multinomial distributions and Dirichlet priors [7], we construct $T$ as the posterior Dirichlet distribution of transition probabilities. In other words:

$$\begin{aligned}&\forall s \in S \centerdot \forall \pi \in \Pi(S) \centerdot T(s)(\pi) = \mathsf{P}(\pi|\boldsymbol{\alpha_s}, \boldsymbol{c_s}) \propto \mathsf{P}(\boldsymbol{c_s}|\pi) p(\pi|\boldsymbol{\alpha_s})\\ &= \text{Multinomial}(\boldsymbol{c_s}|\pi) T^0(s)(\pi) = \text{Multinomial}(\boldsymbol{c_s}|\pi)\, \text{Dirichlet}(\pi|\alpha_s)\\ &= \text{Dirichlet}(\pi|\boldsymbol{\alpha_s} + \boldsymbol{c_s})\end{aligned}$$

That is, for any state $s \in S$, $T(s)$ is a Dirichlet distribution with parameters $\boldsymbol{\alpha_s} + \boldsymbol{c_s}$. Next, recall that $\widehat{\mathcal{M}_E} \in \Pi(\mathbb{M})$. We define $\widehat{\mathcal{M}_E}$ using $T$ as follows:

$$\forall M = (S, I, \widetilde{R}) \in \mathbb{M} \centerdot \widehat{\mathcal{M}_E}(M) = \prod_{s \in S} T(s)(\widetilde{R}(s))$$

That is, for any $M$ which has a set of states and an initial state which are the same as $\boldsymbol{M_\diamond}$, the probability of drawing it from $\widehat{\mathcal{M}_E}$ is the probability of drawing each of its states transition probabilities from the corresponding Dirichlet distributions in $T$.

Given $\widehat{\mathcal{M}_E}$, we still do not know how to construct $\widehat{\mathcal{P}_E}$, or calculate its statistic measures. Since $\mathbb{M}$ is infinite, an exhaustive construction of $\widehat{\mathcal{P}_E}$ is impossible. Instead, we sample $\widehat{\mathcal{P}_E}$ to generate a vector $\widetilde{\boldsymbol{M_E}}$ of $n$ DTMCs, such that $\forall \widetilde{M} \in \widetilde{\boldsymbol{M_E}} : \widetilde{M} \sim \widehat{\mathcal{M}_E}$. More specifically, we have $\widetilde{\boldsymbol{M_E}} = (\widetilde{M}_1, \ldots, \widetilde{M}_n)$ such that for $1 \leq i \leq n$, $\widetilde{M}_i \sim \widehat{\mathcal{M}_E} = (S, I, \widetilde{R}_i)$ where $\forall s \in S \centerdot \widetilde{R}_i(s) = \pi_{is} \sim T(s) \sim \text{Dirichlet}(\boldsymbol{\alpha_s} + \boldsymbol{c_s})$. That is, we construct each DTMC $\widetilde{M}_i$ by drawing the transition probabilities for each of its states from the corresponding Dirichlet distribution in $T$. The algorithm for drawing variates from Dirichlet distributions is well known [3]. Finally, given $\widetilde{\boldsymbol{M}_{\boldsymbol{E}}}$, we construct the vector of probabilities $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}} = \langle r_1, \ldots, r_n \rangle$, such for $1 \leq i \leq n$, $r_i = \widetilde{M_i} \models \varphi$. We compute each $r_i$ using a probabilistic model checker. Our main result is that $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$ is drawn from $\widehat{\mathcal{P}_E}$, as expressed in Theorem 2.

**Theorem 2.** *If $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$ and $\widehat{\mathcal{P}_E}$ are defined as above, then $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}} \sim \widehat{\mathcal{P}_E}$.*

*Proof.* We wish to show that each $\forall r_i \in \widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$, $r_i \sim \widehat{\mathcal{P}_E}$, or equivalently, $\forall r \in \mathrm{dom}(\mathscr{M}) \centerdot \mathsf{P}(r_i \leq r) = \mathit{CDF}(\widehat{\mathcal{P}_E})(r)$. This holds because:

$$\mathsf{P}(r_i \leq r) = \mathsf{P}((M \models \varphi) \leq r \mid M \sim \widehat{M_E})$$
$$= \mathit{CDF}(\widehat{M_E})(\{M \mid (M \models \varphi) \leq r\}) = \mathit{CDF}(\widehat{M_E})(\boldsymbol{M}_r) = \mathit{CDF}(\widehat{P_E})(r)$$

The last equality follows from (3). This completes the proof.

□

To quantify the error between $\widehat{M} \models \varphi$ and $G \models \varphi$, we analyze $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$ with the usual statistical measurements, and determine whether $\widehat{M}$ is a suitable approximation of $G$. Specifically, for our experiments, we determine the 5th and 95th percentiles of $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$ to find a 90% credible interval. If this interval is too wide, we narrow it by performing more trials of $G$ to increase the size of $E$.

Even for a fixed $E$, our analysis approximately characterizes the true distribution $\widehat{\mathcal{P}_E}$. However, with a sufficiently large number of samples of $\widehat{\mathcal{P}_E}$, i.e., $|\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}|$, we expect these approximations to approach true values. Since each sample is obtained by running an automated tool (e.g., PRISM), as opposed to running a trial of a physical robot team, it is feasible to construct a sufficiently large $\widetilde{\boldsymbol{P}_{\boldsymbol{E}}}$.

## 4 Experimental Results

We validate our approach on an example representing a demining operation by a team of communicating Kilobots. Our tools and examples, and instructions to reproduce our experiments are available at `https://db.tt/Wc9tBsNd`. All our experiments were done on a 4 core 3.1GHz i5 machine with 4GB of RAM. Kilobots are very simple robots. They can communicate with each other within a very limited range (a few inches) by bouncing infrared signals off the table they are moving on, and they move by vibrating two motors at different speeds. They possess no reliable localization.

*The Scenario.* A mine has been placed in a culvert beneath a road. The culvert is too small to admit advanced robots, so Kilobots must be used instead. One, two, or three Kilobots are used as a team of sweepers. Each Kilobot enters the culvert, traverses it in search of the mine, and returns to the mouth of the culvert for recovery, and to communicate whether it found the mine. We represent the mine and the base station at the mouth of the culvert by two (static) Kilobots. We place sweeper Kilobots immediately adjacent to the base station on deployment. The mine continually broadcasts a "Mine Here" message, while the base station initially broadcasts a "Base Here" message.

Each sweeper behaves as follows: (i) upon hearing the "Base Here" message for the first time, it begins moving forward; (ii) if it hears no other messages, it turns around at a predefined timeout (2 minutes); (iii) at any time, if it hears a "Mine Here" message or "Mine Found" message it begins transmitting a "Mine Found" message itself; also, if it hasn't turned around yet, it does so; (iv) if the base station hears a "Mine Found" message, it begins transmitting a "Mission Success" message instead; (v) if a sweeper moving backward hears a "Base Here" or "Mission Success" message, it assumes it is in the recovery zone, and stops.

Note that while each robot in our scenario has the same intent, it has a distinct behavior due to its motion characteristic and its release time. In other words, different robots released at the same time would behave differently, and the same robot released at a different time would also behave differently. Moreover, our approach also allows for different robots to have different intents, and perform multiple tasks in sequence or in parallel. This would lead to more complex DTMCs and increased verification time. However, in our experience, PRISM is able to handle systems with millions of reachable states.

*Metrics.* We use the following three measures of success: (i) $f$: the probability that at least one Kilobot finds the mine (i.e., transmits "Mine Found"); (ii) $s$: the probability that the base station knows about the mine (i.e., transmits "Mission Success"); and (iii) $r$: the expected number of Kilobots that return to the recovery zone, irrespective of whether it found the mine or not. Note that each of these measures provides valuable information about the mission success that is not supplied by the other two.

*Modeling Kilobots.* We first reproduced our scenario physically in the laboratory using actual Kilobots. We ran the scenario 190 times and noted 7 distinct Kilobot behaviors. Next, we reproduced these behaviors in the VREP simulator by tuning the Kilobot model parameters appropriately. Subsequently, all our experiments were done via simulation. This physical-simulation hybrid approach enabled us to perform as many experiments as needed while grounding our results on observed behavior of physical Kilobots.

The Kilobot model in VREP has four parameters, each corresponding to the speed of a motor. To reproduce the observed behavior of a real Kilobot we first manually tuned them to appropriate values. Each VREP simulation is completely deterministic. Next, in order to introduce randomness across different experiments, we modified these parameters at each simulation step to a value selected from a normal distribution with mean equal to its tuned value and standard deviation 25. VREP has a sophisticated physics engine, and our approach produced simulated behaviors that are observably analogous to actual Kilobots.

*Constructing Projection DTMCs.* We discretized time at 20s units, and space into a $3 \times 8$ 2-dimensional grid. Since each Kilobot $G_i$ has a maximum turn around time of 120s (when it times out), it has 7 projections corresponding to 0s, 20s, 40s, 60s, 80s, 100s, 120s. We constructed each projection $\langle M_i, t_j \rangle$ by simulating Kilobot $G_i$ 30 times with a pre-programmed turn around time of $t_j$. Using the results of these simulations as our evidence $E$, $\langle M_i, t_j \rangle$ is constructed as described in Sec. 3.1.

**Table 1.** Experiment results (3x8)

| Team | $f^*$ | $\widehat{f}$ | $\widetilde{f_\mu}$ | $\widetilde{f_5}$ | $\widetilde{f_{95}}$ | $s^*$ | $\widehat{s}$ | $\widetilde{s_\mu}$ | $\widetilde{s_5}$ | $\widetilde{s_{95}}$ | $r^*$ | $\widehat{r}$ | $\widetilde{r_\mu}$ | $\widetilde{r_5}$ | $\widetilde{r_{95}}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3-2-1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.96 | 0.96 | 0.91 | 0.99 | 2.20 | 2.17 | 2.17 | 1.97 | 2.38 |
| 4-6-1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.97 | 0.96 | 0.96 | 0.91 | 0.99 | 1.67 | 1.23 | 1.23 | 1.14 | 1.33 |
| 4-6-2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.47 | 0.43 | 0.43 | 0.29 | 0.58 | 0.83 | 0.70 | 0.70 | 0.55 | 0.89 |
| 5-6-2 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.50 | 0.43 | 0.43 | 0.28 | 0.61 | 0.83 | 0.72 | 0.73 | 0.53 | 0.91 |
| 5-6-7 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.43 | 0.29 | 0.29 | 0.19 | 0.38 |
| 6-1-7 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.93 | 0.96 | 0.96 | 0.91 | 0.99 | 1.57 | 1.23 | 1.24 | 1.14 | 1.35 |
| 6-5-7 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.20 | 0.29 | 0.30 | 0.19 | 0.41 |
| 7-3-5 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.70 | 0.83 | 0.83 | 0.72 | 0.92 | 0.70 | 0.85 | 0.85 | 0.73 | 0.94 |
| 7-3-6 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.83 | 0.83 | 0.84 | 0.74 | 0.92 | 1.17 | 1.11 | 1.12 | 0.95 | 1.25 |
| 7-6-1 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 0.90 | 0.96 | 0.96 | 0.92 | 0.99 | 1.63 | 1.23 | 1.24 | 1.13 | 1.34 |

*Team Selection.* To validate our approach we used teams of 3 Kilobots. Since each team is specified by a set of Kilobots and their deployment order, there are 210 such possible teams. For practicality, we validated our approach on ten randomly selected teams as a representative sample. For each team in the sample, we conducted two experiments, which we describe now in more detail.

*Experiment One: Predictions Using Projections.* First, we computed the predicted values of $f$, $s$ and $r$ by combining the constructed projections $\langle M_i, t_j \rangle$ as described in Sec. 2.4 and verifying the resulting DTMC using PRISM [9]. These predictions are denoted $\widehat{f}$, $\widehat{s}$, and $\widehat{r}$, respectively and shown in the corresponding column of Table 1. To evaluate the accuracy of these predictions, we also computed the corresponding "observed values" $f^*$, $s^*$ and $r^*$. Specifically, each observed value (e.g., $f^*$) is the average of the corresponding success measure (e.g., $f$) over 30 simulations of the corresponding team using VREP. During each team simulation (e.g., team 5-6-2), Kilobots are introduced to the scene at 20s intervals in the order specified by the team (e.g., $G_5 \rightarrow G_6 \rightarrow G_2$). The observed values are shown in the corresponding columns of Table 1. Note that, in each case, the predicted and observed results are close, indicating that our approach is sound, and that our assumptions do not distort results significantly.

*Experiment Two: Error Quantification.* Next, we computed $(\widetilde{f_\mu}, \widetilde{f_5}, \widetilde{f_{95}})$, $(\widetilde{s_\mu}, \widetilde{s_5}, \widetilde{s_{95}})$, and $(\widetilde{r_\mu}, \widetilde{r_5}, \widetilde{r_{95}})$, which correspond, for each measure, to the mean 5th-percentile, and the mean 95th-percentile of the Dirichlet sampling error estimation method described in section 3. Each mean was computed using 200 Dirichlet samples. The results are also shown in the corresponding columns of Table 1. Note that in most cases, the observed results, e.g., $f^*$, fall within the 90% confidence interval, e.g., between $\widetilde{f_5}$ and $\widetilde{f_{95}}$.

We also experimented with teams of varying size. The average model checking times were 1.98s, 2.19s, 4.94s, 10.71s, and 42.20s, for team sizes 2 through 6, respectively. For teams of size 7, PRISM ran out of memory on our 4G machine.

## 5 Conclusion

We presented an approach to analytically construct a team of communicating Markovian robots that achieves a specific task with quantified probability of success. We first construct a set of Discrete Time Markov Chains (DTMCs) that each capture a specific "projection" of the behavior of an individual robot. Next, given a specific team, we construct the DTMC for its behavior by combining the projection DTMCs appropriately. Finally, we compute the performance of the team using Probabilistic Model Checking. The best team is selected by repeating this process for multiple candidates. We also show how to quantify the error in our results due to finite sampling when constructing the projection DTMCs. We prove the correctness of our approach formally, and also validate it empirically on a mine detection task by a team of communicating Kilobots. An important direction for future work is to extend our approach to non-Markovian systems, and also to quantify the error due to discretization of time and space.

## References

1. V-REP: Virtual robot experimentation platform (2014)
2. Chen, T., Diciolla, M., Kwiatkowska, M.Z., Mereacre, A.: Quantitative Verification of Implantable Cardiac Pacemakers. In: Proceedings of the 33rd Real-Time Systems Symposium (RTSS 2012), San Juan, PR, USA, pp. 263–272. IEEE Computer Society (December 2012)
3. Devroye, L.: Non-Uniform Random Variate Generation. Springer, New York (1986)
4. Ghorbal, K., Duggirala, P.S., Kahlon, V., Ivančić, F., Gupta, A.: Efficient probabilistic model checking of systems with ranged probabilities. In: Finkel, A., Leroux, J., Potapov, I. (eds.) RP 2012. LNCS, vol. 7550, pp. 107–120. Springer, Heidelberg (2012)
5. Hansson, H., Jonsson, B.: A Logic for Reasoning about Time and Reliability. Formal Aspects of Computing (FACJ) 6(5), 512–535 (1994)
6. Heath, J., Kwiatkowska, M.Z., Norman, G., Parker, D., Tymchyshyn, O.: Probabilistic model checking of complex biological pathways. Theoretical Computer Science (TCS) 391(3), 239–257 (2008)
7. Huang, J.: Maximum likelihood estimation of dirichlet distribution parameters. Technical report, Robotics Institute, Carnegie Mellon University (2005)
8. Konur, S., Dixon, C., Fisher, M.: Analysing robot swarm behaviour via probabilistic model checking. In: Robotics and Autonomous Systems (2011)
9. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
10. Kwiatkowska, M.Z., Norman, G., Sproston, J.: Probabilistic Model Checking of Deadline Properties in the IEEE 1394 FireWire Root Contention Protocol. Formal Aspects of Computing (FACJ) 14(3), 295–318 (2003)
11. Rubenstein, M., Ahler, C., Nagpal, R.: Kilobot: A low cost scalable robot system for collective behaviors. In: IEEE Intl. Conf on Robotics and Automation (ICRA), p. 6 (2012)
12. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, Available as Technical Report MIT/LCS/TR-676 (1995)