Multi-Robot Planning Under Uncertain Travel Times and Safety Constraints

Masoumeh Mansouri[1*], **Bruno Lacerda**[2*], **Nick Hawes**[2] and **Federico Pecora**[1]
[1]Örebro University, Sweden
[2]Oxford Robotics Institute, University of Oxford, UK
{masoumeh.mansouri, federico.pecora}@oru.se, {bruno, nickh}@robots.ox.ac.uk

Abstract

We present a novel modelling and planning approach for multi-robot systems under uncertain travel times. The approach uses generalised stochastic Petri nets (GSPNs) to model desired team behaviour, and allows to specify safety constraints and rewards. The GSPN is interpreted as a Markov decision process (MDP) for which we can generate policies that optimise the requirements. This representation is more compact than the equivalent multi-agent MDP, allowing us to scale better. Furthermore, it naturally allows for asynchronous execution of the generated policies across the robots, yielding smoother team behaviour. We also describe how the integration of the GSPN with a lower-level team controller allows for accurate expectations on team performance. We evaluate our approach on an industrial scenario, showing that it outperforms hand-crafted policies used in current practice.

1 Introduction

Multi-robot path planning is important in many real-world robotics applications, such as mining, construction, and warehouse automation. A desirable feature of any automated multi-robot method is the ability to *specify requirements over team behaviour*. For example, we may want to require that a robot is always present to collect the packages output by a conveyor belt; or that a taxi is always present at a taxi rank to collect passengers. Providing such *team-level guarantees* is challenging when dealing with mobile robots due to uncertainty over the durations of navigation actions. This uncertainty stems from many sources, e.g., the dynamics of individual robots are typically only partially known; and interactions between robots jointly navigating in a shared space introduce further unmodelled dynamics. Today's commercial solutions for multi-robot path planning remove uncertainty by extensively engineering the environment, leading to hand-crafted policies for assigning tasks to robots [Pecora *et al.*, 2018]. Whilst these are reliable, every new environment or application requires significant bespoke engineering, and the resulting system provides few *formal guarantees on team behaviour*.

To address this, we propose a novel method for deriving policies that regulate the behaviour of individual robots in accordance with high-level requirements on the overall behaviour of the team. The team is modelled as a generalised stochastic Petri net (GSPN) [Balbo, 2007], in which paths and locations are represented as places, robots are represented as tokens, and the uncertain navigation durations are encoded as probabilistic firing rates of transitions. We refer to this as a *multi-robot GSPN* (MR-GSPN). With this we can represent *team requirements*, specifically *safety specifications*, as restrictions on the markings of the GSPN; and *team performance* as a reward to be maximised over the transitions of the GSPN. Robots are represented *anonymously* in the MR-GSPN, reducing the effect of the exponential blow-up usually associated with multi-agent models. Following Markov automata (MA)-based semantics [Eisentraut *et al.*, 2013], an MR-GSPN can be interpreted as a Markov decision process (MDP) [Puterman, 1994]. This MDP can be solved to generate policies that optimise the team behaviour against the team requirements and performance objective. In order to robustly maintain the safety specification, we use learnt probabilistic models of navigation task duration. These models are learnt from simulations of the team navigating in the target environment. As a consequence, the generated policies account for the kino-dynamics of the robots and of the team as a whole, and are therefore appropriate for regulating the behaviour of a team of real robots.

The main contributions of this paper are the presentations of: (i) MR-GSPN, a novel GSPN based modelling formalism for multi-robot teams; (ii) an MA-based process for using an MR-GSPN to find policies that maximise team performance whilst maintaining a team-level safety specification for as long as possible; and (iii) integration with a lower-level team controller, used to obtain the stochastic rates associated with the duration of navigation actions. To the best of our knowledge, this is the first time the semantics of GSPNs as MAs has been exploited for robot planning. Similarly, this is the first work that builds a GSPN model using accurate transition models from a kino-dynamically feasible, lower-level controller.

2 Related Work

The clear semantics of concurrency in Petri nets (PNs) has led to their prior use in robotics. For example, [Ziparo *et al.*, 2011] use PNs to create single-robot behaviour models which support robust execution strategies. PNs have also been used to model

*These authors contributed equally to this work.

the behaviour of multi-robot systems. [Lacerda and Lima, 2011] used a PN model to synthesise a supervisory controller for a team of soccer playing robots. [Mahulea and Kloetzer, 2018] map a homogenous robot team to tokens in a PN to provide a compact representation of interchangeable robots in order to generate multi-robot paths satisfying a boolean team specification. We also exploit the mapping of robots to tokens, but extend it to GSPNs in order to cope with uncertain action durations. [Costelha and Lima, 2012] use a GSPN to model and analyse robot behaviour, but do not synthesise policies from the PN, as we do here.

Our approach is an example of multi-robot path planning under uncertainty. Many existing multi-agent path planning approaches ignore robot kino-dynamics and uncertainty [Felner *et al.*, 2017] and instead rely on lower level control to provide robustness against execution time variations from planned behaviour [Pecora *et al.*, 2018]. Failing to represent the inherent uncertainty in the domain means the system behaviour can be unpredictable. Recent work addresses this by formulating specific instances of multi-robot path planning problems with limited forms of uncertainty, e.g., delay probabilities [Ma *et al.*, 2017]. Multi-agent MDPs [Boutilier, 1996] have been used to synthesise robot team behaviour, but their general nature results in poor scalability. This is typically mitigated by exploiting assumed structure in the MDP, such as sparse interaction between agents [Scharpff *et al.*, 2015; Faruq *et al.*, 2018]. We provide scalability through the use of a GSPN model that yields an MDP with a smaller state space, rather than assuming a particular structure. In contrast to all this prior work, we synthesise team behaviour to meet a formal safety specification. Only limited prior work exists on this topic. [Faruq *et al.*, 2018] create policies for robots independently which are then combined to provide team guarantees on task completions given robot failures. Their approach assumes sparse interactions between robots (not valid in our problem) and requires that all robots wait and synchronise their discrete actions (leading to inefficient robot behaviour).

Generalised semi-Markov decision processes (GSMDPs) have also been applied in robotics [Younes and Simmons, 2004] and are closely related to MA, the model we use to interpret the marking process of a GSPN. The two crucial differences are that GSMDPs allow for more general continuous-time models, whilst MA explicitly separate immediate transitions (decisions) from exponential transitions (waiting for some uncontrollable event to occur). We will investigate the relation between these models in future work. [Messias *et al.*, 2013] use a GSMDP to coordinate a small team of robots on a RoboCup task. This approach uses the continuous time models to remove synchronisation points, demonstrating the power of continuous time models in robotics. Although similar to this work, we additionally provide a probabilistic guarantee over the behaviour of the entire team.

3 GSPNs for Multi-robot Path Planning

3.1 Generalised Stochastic Petri Nets

We start by introducing Petri nets (PNs) and then extend the definition to include timing uncertainty.

Definition 1 *A PN is a tuple $N = \langle P, T, W^-, W^+, \overline{M} \rangle$ where P is a finite set of places; T is a finite set of transitions; $W^+, W^- : P \times T \to \mathbb{N}$ are arc weight functions; and $\overline{M} : P \to \mathbb{N}$ is the initial marking.*

A marking $M : P \to \mathbb{N}$ represents a state of the system, with $M(p) = q$ meaning that in M there are q tokens in place p. The dynamics of a PN are defined by the firing rule, which determines the flow of tokens between places according to the arc weight functions. Intuitively, $W^-(p,t)$ represents the tokens that are *consumed* from p by the firing of t and $W^+(p,t)$ represents the tokens *produced* by the firing of t and placed in p. Transitions are enabled when each place p holds at least as many tokens as the ones to be consumed by t.

Definition 2 *Let M be a marking. Transition t is said to be* enabled *in M if $M(p) \geq W^-(p,t)$ for all $p \in P$.*

A transition t that is enabled in a marking M can *fire*, evolving to a marking where t consumes and produces tokens according to W^- and W^+.

Definition 3 *Let M be a marking and t an enabled transition in M. The firing of t results in the marking M' where, for each $p \in P$, $M'(p) = M(p) - W^-(p,t) + W^+(p,t)$. We denote this as $M \xrightarrow{t} M'$.*

Definition 4 *The set of reachable markings is defined as:*

$$R(N) = \{M \mid \textit{exists } t_0 \cdots, t_n,\ M_0, \cdots, M_{n+1} \textit{ such that } M_0 = \overline{M},\ M_{n+1} = M \textit{ and } M_i \xrightarrow{t_i} M_{i+1}\}$$

For the GSPN models used in this work all arcs have weight 1. Thus, to simplify notation, we represent the arc weight functions using pre and postsets.

Definition 5 *We define the preset of t as ${}^\bullet t = \{p \in P \mid W^-(p,t) = 1\}$ and the postset of t as $t^\bullet = \{p \in P \mid W^+(p,t) = 1\}$.*

GSPNs are an extension of PNs where transitions are partitioned into *immediate* and *exponential* transitions.

Definition 6 *A GSPN is tuple $N = \langle P, T, W^-, W^+, \overline{M}, \lambda \rangle$, where P, T, W^-, W^+ and $\overline{M}$ are the same as for a PN, with T partitioned into a set T^i of immediate transitions and a set T^e of exponential transitions; and $\lambda : T^e \to \mathbb{R}_{>0}$ associates each exponential transition t^e with a rate $\lambda(t^e)$.*

Enabled immediate transitions t^i can be fired in zero time and, in this work, represent actions available to a control policy. Our goal will be to find transition firings that optimise some objective, as defined in Section 3.3. If the control policy does not execute an enabled immediate transition, then enabled exponential transition t^e takes some stochastic time to fire, according to an exponential distribution with expected value $1/\lambda(t^e)$. When more that one exponential transitions is enabled, there is a *race condition*. Race conditions are resolved stochastically according to the rates of the racing transitions. For example, assume that k exponential transitions $t^e_1, ..., t^e_k$ are enabled in marking M. The probability of transition t^e_i being the first to fire is given by:

$$P(t^e_i \mid M) = \frac{\lambda(t^e_i)}{\sum_{j=1}^{k} \lambda(t^e_j)}$$

3.2 Multi-Robot Navigation GSPNs

Consider a team of n robots in an environment discretised into a *navigation graph* $G = \langle V, E\rangle$, with $init(v)$ representing the initial number of robots at node v. Assume that in certain nodes the robots interact with a process that is not under our control (e.g., a robot collecting the packages output by a conveyor belt; or a taxi being filled with passengers). The nodes of the navigation graph are thus partitioned into $V = V^i \cup V^e$. Nodes in V^i represent nodes where the control policy can decide to trigger a navigation action. Nodes in V^e are nodes where the robots must wait for the external process to finish.

Definition 7 *A multi-robot navigation GSPN (MR-GSPN) is a tuple* $N_G = \langle P_G, T_G, W_G^-, W_G^+, \overline{M_G}, \lambda\rangle$ *where:*

- $P_G = P_{V^i} \cup P_{V^e} \cup P_E$, *where* $P_{V^i} = \{p_v \mid v \in V^i\}$, $P_{V^e} = \{p_v \mid v \in V^e\}$ *and* $P_E = \{p_{(v,v')} \mid (v, v') \in E\}$*;*
- $T_G = T^i_{E,s} \cup T^e_{E,s} \cup T^e_{E,f}$, *where:*

$$T^i_{E,s} = \{t^i_{(v,v'),s} \mid v \in V^i \text{ and } (v, v') \in E\}$$

$$T^e_{E,s} = \{t^e_{(v,v'),s} \mid v \in V^e \text{ and } (v, v') \in E\}$$

$$T^e_{E,f} = \{t^e_{(v,v'),f} \mid (v, v') \in E\}$$

- W_G^- *and* W_G^+ *are obtained from the following pre and postset definitions:*
  - *For* $t_{(v,v'),s} \in T^i_{E,s} \cup T^e_{E,s}$, ${}^\bullet t_{(v,v'),s} = \{p_v\}$ *and* $t_{(v,v'),s}{}^\bullet = \{p_{(v,v')}\}$*;*
  - *For* $t_{(v,v'),f} \in T^e_{E,f}$, ${}^\bullet t^e_{(v,v'),f} = \{p_{(v,v')}\}$ *and* $t^e_{(v,v'),f}{}^\bullet = \{p_{v'}\}$*;*

  Furthermore, we impose that for $p_v \in P_{V^e}$ *there exists exactly one transition* t *such that* $p_v \in {}^\bullet t$*;*
- $\overline{M_G}(p_v) = init(v)$ *for* $p_v \in P_{V^i} \cup P_{V^e}$ *and* $\overline{M_G}(p_{(v,v')}) = 0$ *for all* $p_{(v,v')} \in P_E$*.*

The key point of the MR-GSPN is that we *represent robots as tokens*. Furthermore, we split navigation between two locations, v and v', into two transitions. The first transition represents the triggering of the navigation action to move from v to v'. If $v \in V^i$, this transition is immediate and is under our control. If $v \in V^e$, the transition is exponential, with a rate representing the expected duration of the external process occurring in v. We also impose that, for nodes $v \in V^e$, only one transition can remove tokens from p_v (intuitively, from v there is no choice of where to navigate). This assumption is without loss of generality and can be dropped by adding extra places to the model. We refrain from doing so for the sake of clarity of notation. The second transition is an exponential transition representing the expected time a robot will spend traversing the edge (v, v'). If a control policy chooses not to fire an immediate transition (or there are no immediate transitions enabled), a race condition is triggered and one of the exponential transitions fires, with the probabilities of firing of each transition being defined by their rates. Fig. 1 depicts the MR-GSPN representation of a single navigation edge.

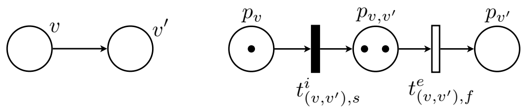

Figure 1: Example of construction of a MR-GSPN. Left: two nodes v and v' connected by an edge in a navigation graph; Right: The fragment of the MR-GSPN representing navigation between v and v', where $v \in V^i$. Note that if $v \in V^e$, then both depicted transitions would be exponential. The depicted marking represents a state where one robot is at v and two robots are navigating from v to v'.

3.3 Goal Specification

We specify the goal as gathering as much reward as possible until a set of *failure markings* is reached.

Definition 8 *We define the subset of markings satisfying a linear constraint over the markings of* N_G *as* $C = \{M \in R(N_G) \mid \sum_{p\in P_G} k_p M(p) \bowtie b\}$, *where, for* $p \in P_G$, $k_p \in \mathbb{N}$, $\bowtie \in \{<, \le, =, \ge, >\}$, *and* $b \in \mathbb{N}$*. The set of of markings that satisfies the conjunction of a set of linear constraints is defined as* $\mathcal{C} = C_1 \cap \dots \cap C_m$*. Finally, the set of failure markings is defined as* $bad = R(N_G) \setminus \mathcal{C}$*.*

We assume a special constraint $C_1 = \{M \in R(N_G) \mid \sum_{p\in P_{V^e}} M(p) \ge b\}$, i.e., a constraint that requires that the number of tokens in places representing nodes for which the robots undergo an external process must be maintained above a bound b. Note that the transitions removing tokens from such places are exponential transitions, subject to an uncontrollable external process with an exponentially distributed duration. Since we can only fire a finite number of immediate transitions consecutively (at most equal to the number of robots) before getting into a marking where no immediate transition is enabled, race conditions will occur infinite times in any infinite run of N_G. In these race conditions there is some probability of tokens from places in P_{V^e} being removed. Thus, it is not possible to indefinitely keep the MR-GSPN in markings that satisfy C_1. This is needed to guarantee convergence of our objective.

Definition 9 *A* transition firing reward *is a function* $r_{T^i} : T^i \to \mathbb{R}_{\ge 0}$.

Transition firing rewards represent the utility of firing immediate transitions in the MR-GSPN (e.g., t^i can represent an AGV starting to move after unloading goods at a processing station, or a bus leaving after dropping off its passengers).

Problem 1 *Given* $\mathcal{C} = C_1 \cap \dots \cap C_m$ *and* $r_{T^i} : T^i \to \mathbb{R}_{\ge 0}$, *find a mapping from* $R(N_G)$ *to* T^i *that maximises the expected cumulative value of* r_{T^i} *until a marking in* bad *is reached.*

In Section 4.3, we will pose Problem 1 as a cumulative reward maximisation problem in an MDP representing N_G.

4 MDPs as Planning Models for MR-GSPNs

4.1 Markov Decision Processes

We start by introducing the concepts and notation required to formalise the translation of the MR-GSPN to an MDP.

Definition 10 *An MDP is a tuple* $\mathcal{M} = \langle S, \overline{s}, A, \delta\rangle$, *where:* S *is a finite set of states;* $\overline{s} \in S$ *is the initial state;* A *is a finite* *set of actions; and $\delta : S \times A \times S \to [0,1]$ is a probabilistic transition function, where $\sum_{s' \in S} \delta(s,a,s') \in \{0,1\}$ for all $s \in S$ and $a \in A$. We define the set of* enabled *actions in $s \in S$ as $A_s = \{a \in A \mid \delta(s,a,s') > 0$ for some $s' \in S\}$.*

An MDP model represents the possible evolutions of the state of a system: in each state s, any of the enabled actions $a \in A_s$ can be selected and the system evolves to a successor state s' according to the values of $\delta(s,a,s')$. We assume that the MDP has no deadlocks, i.e., $A_s \neq \emptyset$ for all $s \in S$. This can be ensured by adding self-loops labelled with a dummy action to deadlocked states.

Definition 11 *An infinite path through $\mathcal{M}$ starting in s is a sequence $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots$ where $s_0 = s$ and $\delta(s_i, a_i, s_{i+1}) > 0$ for $i \in \mathbb{N}$. We denote the set of all infinite paths of $\mathcal{M}$ starting in s by $IPath_{\mathcal{M},s}$.*

Definition 12 *A* deterministic and stationary policy *is a function $\pi : S \to A$.*

Policies map the current state s to an action $a \in A_s$. Under a particular policy π for $\mathcal{M}$, all nondeterminism is resolved and the behaviour of $\mathcal{M}$ is fully probabilistic. This leads to the construction of a probability measure $Pr^{\pi}_{\mathcal{M},s}$ over paths of $\mathcal{M}$ starting in s and following π, which in turn allows us to reason about the *expected cumulative value of a reward function*.

We will pose Problem 1 as that of finding policies maximising an expected reward until an *unavoidable* state is reached.

Definition 13 *Let $F \subset S$. We write $Pr^{\pi}_{\mathcal{M},s}(\Diamond F)$ to denote the probability of reaching a state in F when following policy π, starting from s; and $Pr^{\min}_{\mathcal{M},s}(\Diamond F)$ to denote the minimum probability (across all policies) of reaching F. We say F is* unavoidable *if any infinite path $\rho \in IPath_{\mathcal{M},\overline{s}}$ eventually reaches F, i.e., $Pr^{\min}_{\mathcal{M},\overline{s}}(\Diamond F) = 1$.*

Definition 14 *Let $F \subset S$ be unavoidable and $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \in IPath_{\mathcal{M},\overline{s}}$. We write n_F to denote the minimum value such that $s_{n_F} \in F$.*

Definition 15 *Let $r : S \times A \to \mathbb{R}_{\geq 0}$ be a reward function, $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \in IPath_{\mathcal{M},s}$, and $F \subset S$ be a set of unavoidable states. We define the reward accumulated by ρ until F is reached as:*

$$cumul^F_r(\rho) = \sum_{i=0}^{n_F} r(s_i, a_i)$$

Problem 2 *Let $E^{\pi}_{\mathcal{M},\overline{s}}(cumul^F_r)$ denote the expected value of $cumul^F_r$ when following policy π. Calculate the maximum value of $E^{\pi}_{\mathcal{M},\overline{s}}(cumul^F_r)$ across all policies, along with the corresponding optimal policy $\pi^* : S \to A$, i.e., find:*

$$\pi^* = \arg\max_{\pi} E^{\pi}_{\mathcal{M},\overline{s}}(cumul^F_r)$$

By first preprocessing the MDP, replacing transitions from states in F with zero-reward self-loop transitions, we can reduce Problem 2 to an infinite horizon cumulative reward maximisation problem, which can be solved with standard techniques such as value iteration – note that convergence is guaranteed because F is unavoidable and is made *absorbing with zero reward absorbing states* by the described preprocessing step. Hence, any path through $\mathcal{M}$ will eventually reach F and stop accumulating reward from then on. For this problem, deterministic and stationary policies suffice [Puterman, 1994].

4.2 From GSPNs to MDPs

Our GSPN definition differs from the traditional definition in two ways that reflect the way we interpret immediate transitions. First, we do not include *random switches*, which are typically used to determine the firing probabilities for immediate transitions enabled in the same marking. This is because we do not want to fix the behaviour of the GSPN at design time. Instead, we want to generate control policies that choose which immediate transition to fire. Second, we do not disallow the firing of exponential transitions in markings where there are also immediate transitions are enabled. This is because in certain cases we might want some robots to *wait* for more information about the actions being executed by other robots before committing to a decision. This definition of GSPN does not allow us take the usual interpretation of the marking process of a GSPN as a continuous-time Markov chain [Balbo, 2007]. Instead, we take the more general interpretation as a *Markov automaton* (MA) [Eisentraut *et al.*, 2013]. In this paper, we are interested in maximising rewards until reaching a set of unavoidable states. This type of property is time-abstract and thus can be optimised in an MDP representation of the MA [Hatefi and Hermanns, 2012]. Thus, we introduce the translation from a GSPN to an MDP directly, rather than via an MA. We call this the *embedded MDP*.

Definition 16 *Let $N = \langle P, T, W, \overline{M}, \lambda \rangle$ be a GSPN. The embedded MDP is defined as $\mathcal{M}_N = \langle S_N, \overline{M}, A_N, \delta_N \rangle$ where:*

- *$S_N = R(N)$, i.e., the state space is the set of reachable markings of N;*
- *$A_N = T^i \cup \{wait\}$, i.e., the set of actions is the set of immediate transitions plus a special* wait *action that represents waiting for a race condition to be resolved;*
- *$\delta_N : R(N) \times (T^i \cup \{wait\}) \times R(N)$ is such that:*

$$\delta_N(M, a, M') = \begin{cases} 1 & \text{if } a \in T^i \text{ and} \\ & M \xrightarrow{a} M' \\ P(t^e \mid M) & \text{if } a = wait \text{ and} \\ & M \xrightarrow{t^e} M' \\ 0 & \text{otherwise} \end{cases}$$

In the embedded MDP, we can choose an enabled immediate action to fire and evolve the state accordingly; or wait and in that case the state evolution is according to the probabilities of resolution of the race condition currently active.

4.3 Goal Specification as Reward Maximisation

We now formalise Problem 1 over the embedded MDP as the maximisation of a reward until a set of unavoidable states is reached in $\mathcal{M}_{N_G}$, i.e., an instance of Problem 2.

Problem 3 *Let $N_G = \langle P_G, T_G, W^-_G, W^+_G, \overline{M_G}, \lambda \rangle$ be a MR-GSPN (Def. 7), $\mathcal{M}_{N_G} = \langle S_{N_G}, \overline{M_{N_G}}, A_{N_G}, \delta_{N_G} \rangle$ its embedded MDP (Def. 16), bad a set of failure markings (Def. 8) and $r_{T^i} : T^i \to \mathbb{R}_{\geq 0}$ a transition firing reward (Def. 9). We define $r_{N_G} : S_{N_G} \times A_{N_G}$ such that $r_{N_G}(s,a) = r_{T^i}(a)$. Find:*

$$\pi^* = \arg\max_{\pi} E^{\pi}_{\mathcal{M}_{N_G},\overline{M_{N_G}}}(cumul^{bad}_{r_{N_G}})$$

as defined in Problem 2.

Reward function $r_{N_G}(s, a)$ translates the transition reward r_{T^i} to the MDP encoding of the MR-GSPN. Also, recall that bad is unavoidable in $\mathcal{M}_{N_G}$ due to special constraint C_1.

4.4 Discussion

We finish with a brief discussion on the assumptions made and advantages of using MR-GSPNs. Note that the linear constraints must be satisfied by the team in all states, however they cannot be held true by only one team member. To ground the discussion, consider the special constraint $C_1 = \{M \in R(G_N) \mid M(p_{v^e}) \geq 1\}$ for some $p_{v^e} \in P_{V^e}$. This constraint is especially interesting as it quantifies universally over the state space, but existentially over the team members, i.e., it is of the form "for all states there must exist at least one robot at v^e". Also, one robot cannot keep C_1 true for all states and must be replaced by a team member, which must be ready to replace it. One approach to address this constraint optimally is to take into account the joint state of the team, using models such as multi-agent MDPs (MMDP) [Boutilier, 1996], in order to plan for replacement of the robot maintaining the constraint. MMDPs have two main issues in the context of multi-robot systems. First, scalability: even disregarding uncertain durations, a navigation graph with k nodes and n robots entails at least k^n states in the MMDP. Second, MMDPs assume fully synchronised action execution: this can lead to very inefficient team behaviour, as robots need to wait for the team to synchronise at every decision point [Messias *et al.*, 2013]. We are able to mitigate some of the scalability issues by exploiting: the assumption of a homogeneous robot team; the fact that our objective is not robot specific; and our modelling of robots as tokens in the GSPN (making robots anonymous). In our approach, a navigation graph with k nodes and l transitions entails a GSPN with $k + l$ places. Each marking represents the distribution of exactly n tokens over the $k + l$ places, hence the number of reachable markings is given by "$k + l$ multichoose n".

While the state still grows exponentially with the number of robots, it allows for significant savings in number of states, as we show in Section 6.1. As the GSPN is an event-based model, synchronisation is not required: triggered immediate transitions correspond to sending navigation commands to robots; as they navigate and change state, the control policy updates its state immediately by firing the corresponding exponential transition, yielding smooth, asynchronous policy execution.

5 Simulation-Based Duration Estimates

In order to robustly maintain the safety specification, we require accurate models of the durations of navigation edges in G. Obtaining such models is challenging as the precise dynamics of a robot are usually unknown. Interactions among robots also affect durations (e.g., robots yielding to, or avoiding each other), as well as other sources of uncertainty in the environment. For these reasons, we learn the durations by observing realistic simulations of the robot team performing navigation tasks in the target environment.

To explore the range of multi-robot navigation experiences relevant for the target environment, the robot team must operate in a way that is as similar to the desired behaviour as possible. To achieve this, we control the team in simulation using an existing team controller which integrates coordination, motion planning and robot control [Pecora *et al.*, 2018], and supports the injection of external navigation choices for robots. Given these choices, the fleet controller generates multi-robot paths that take into account the kino-dynamic constraints of individual robots. These paths are jointly executed and supervised by the controller. When generating data for learning we use a randomised policy to provide navigation choices. One simulation run of approximately one hour per team size provides us sufficient data for fitting exponential distributions for each transition of the MR-GSPN. In the experiments described below we use the same multi-robot controller to execute the policies produced from our MR-GSPN approach. This ensures both that the policies are realisable on the robots, and that the transition models match well to runtime performance.

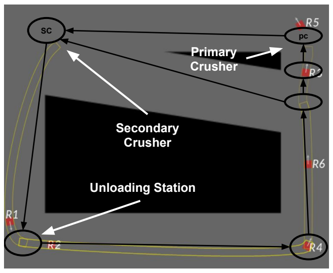

Figure 2: Evaluation environment. Robots are red, planned paths yellow. The navigation graph is overlayed with key locations indicated.

6 Evaluation

Our industrially-motivated evaluation scenario features a team of autonomous electric haulers operating in a quarry, moving between stations. At the unloading station, a hauler can unload gravel obtained from two crushers. The primary crusher (PC) constantly produces gravel, which is continuously output via a conveyor belt. The production of gravel at the PC cannot be stopped under normal circumstances, hence, there should *always be a robot under the PC* so that gravel does not accumulate on the ground, obstructing access to the PC and halting the entire process. The secondary crusher (SC) does not have this constraint, as the gravel produced there is loaded onto haulers manually. Finally, robots are *required to leave the PC when full*. Thus, for this scenario, the safety constraint is that there should always be a hauler under the PC, and reward is obtained when a hauler drives to the unloading station. In our evaluations we use an instance of this problem shown in Fig. 2, which matches a real-world quarry.

6.1 Scalability

The scenario in Fig. 2 was modelled with a MR-GSPN with 15 places: 6 places for locations, 7 places for edges, plus 2 extra places to model the (deterministic) time spent under the PC using an Erlang distribution with 3 states [Younes and Simmons, 2004], i.e., the time spent under the PC is modelled using the place corresponding to the PC location plus 2 extra places that allow us to approximate the amount of time a robot already spent under the PC. Table 1 shows the number of states and the time to build the embedded MDP (our

n	6^n	$\lvert S_{N_G} \rvert$	Construction time (s)
5	7776	11628	1.34
6	46656	38760	6
7	279936	116280	18
8	1679616	319770	57
9	10077696	817190	167
10	60466176	1961256	661

Table 1: Model sizes and time to construct $\mathcal{M}_{G_N}$.

approach was run on a standard desktop pc). For comparison, we compare to 6^n, an underestimate of the size of a MMDP for this problem, as one would need more states to model the number of robots navigating an edge. It can be seen that the proposed MR-GSPN representation is more compact than an MMDP. Solution times are on the order of minutes for the larger models, e.g., for 8 robots it took approximately 7 minutes to compute a policy. We note that, after being computed, the policy can be efficiently executed online and can cope with the uncertain travel times without replanning. Furthermore, most solutions to MMDPs, even approximate ones, do not typically scale to this number of robots.

6.2 Robustness to Disturbances in Travel Times

The purpose of the following evaluation is to measure how robust the team behaviour is to disturbances affecting navigation duration. We compare team behaviour as regulated by two policies: a synthesised policy (SP) obtained from our automated planning approach and a hand-coded policy (HP) currently used in an industrial setting. The HP prescribes that a robot should be assigned to the SC if there is at least one robot queuing behind the robot under the PC.

Note that the HP ignores the durations of navigation tasks, and is hence not capable of predicting the likelihood of a robot being able to reach the PC before the robot currently loading from it leaves. Conversely, the SP is computed with knowledge of the transition rate models, which are in turn directly related to the durations of navigation. We therefore expect the HP to perform well in terms of reward (representing tons of dumped material), but to fail to maintain the team requirement (always one robot under the PC) when disturbances are applied to the duration of the navigation actions. Conversely, we expect the SP to be more conservative in dispatching robots to the SC, as it can consider the probability of robots taking longer to reach their destinations, according to the corresponding learnt exponential transitions. While this should entail less reward than the HP in nominal situations, we expect the SP to be more robust to higher disturbances in navigation duration.

Experimental setup. We use the simulation depicted in Fig. 2. 32 problems of the form $\langle n, PC, D\rangle$ are generated, where n is the number of robots, D is the disturbance profile, and PC is the number of seconds it takes the PC to fill a robot. The disturbance profile is a delay in seconds, applied to a robot as it navigates between any two locations, with a fixed probability of 0.5. Each policy is run for 15 times on each problem for 10 minutes each time. We halt the simulation before 10 minutes if there is no robot under the PC.

Results. Fig. 3 shows the success rate for problem categories, with $n \in \{4, 5\}$, $D \in \{6, 8, 10, 12\}$, and $PC \in \{30, 45\}$. The rate measures the percentage of the 15 instances of each problem type in which team constraint was not violated. SPs are in average 64% more successful than the corresponding HPs. The accumulated reward for problems with 5 robots is shown in Fig. 4. Despite the low success rate of the HPs, the accumulated rewards are similar between the HPs and the SPs. However, note that if we leave the system running for more time, the higher robustness of the SPs allows them to continue accumulating reward, whilst the runs with HPs will likely halt earlier. The HPs have a higher success rate when the time spent at the PC is 45 seconds, as robots spend more time queueing, and thus they are able to send more robots to the SC, also increasing the reward. In contrast, the SPs tend to be more conservative and keep more robots queueing.

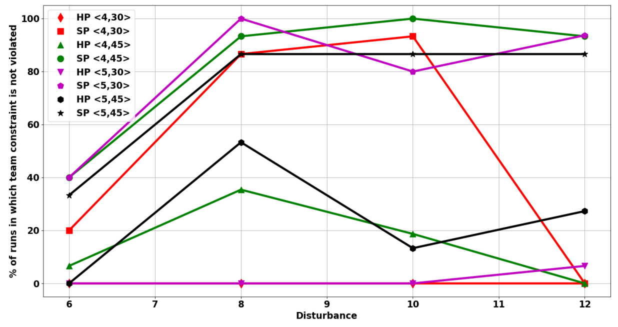

Figure 3: Success rate for HP and SP for different disturbance profiles, in 8 problems. Legend is of the form $< n, PC >$.

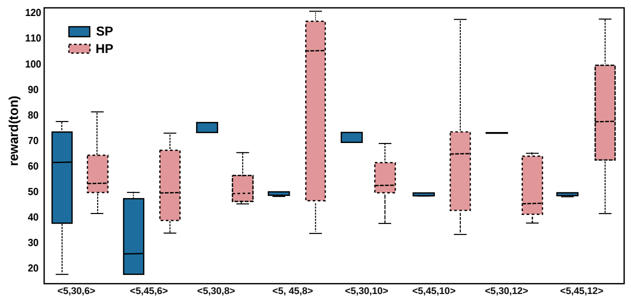

Figure 4: Boxplot of the accumulated reward in terms of dumped material for 5 robots in 8 problems.

7 Conclusions

We presented MR-GSPN, an approach for planning for multi-robot teams. In the future, we intend to investigate the use of the MA representation of the GSPN to plan for more general specifications of team behaviour. Furthermore, given the more compact state space of our approach when compared to MMDPs, we intend to exploit heuristic search and planning under uncertainty techniques over the MR-GSPN model in order to scale up to larger teams.

Acknowledgments

This work was supported by the Swedish Knowledge Foundation (KKS) project "Semantic Robots" and by UK Research and Innovation and EPSRC through the Robotics and Artificial Intelligence for Nuclear (RAIN) research hub [EP/R026084/1].

References

[Balbo, 2007] Gianfranco Balbo. Introduction to generalized stochastic Petri nets. In *Proceedings ot the 7th International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*, pages 83–131. Springer, 2007.

[Boutilier, 1996] Craig Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, pages 195–210, 1996.

[Costelha and Lima, 2012] Hugo Costelha and Pedro Lima. Robot task plan representation by Petri nets; Modelling, identification, analysis and execution. *Autonomous Robots*, 33(4):337–360, Nov 2012.

[Eisentraut *et al.*, 2013] Christian Eisentraut, Holger Hermanns, Joost-Pieter Katoen, and Lijun Zhang. A semantics for every GSPN. In *Proceedings of the 34th International Conference on Applications and Theory of Petri Nets and Concurrency (Petri Nets)*, pages 90–109. Springer, 2013.

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

[Felner *et al.*, 2017] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 29–37, 2017.

[Hatefi and Hermanns, 2012] Hassan Hatefi and Holger Hermanns. Model checking algorithms for Markov automata. *Electronic Communications of the EASST*, 53, 2012.

[Lacerda and Lima, 2011] Bruno Lacerda and Pedro U. Lima. Designing Petri net supervisors from LTL specifications. In *Proceedings of Robotics: Science and Systems VII (RSS)*, 2011.

[Ma *et al.*, 2017] Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding with delay probabilities. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*, pages 3605–3612, 2017.

[Mahulea and Kloetzer, 2018] Cristian Mahulea and Marius Kloetzer. Robot planning based on boolean specifications using Petri net models. *IEEE Transactions on Automatic Control*, 63(7):2218–2225, July 2018.

[Messias *et al.*, 2013] João Vicente Messias, Matthijs T. J. Spaan, and Pedro U. Lima. GSMDPs for multi-robot sequential decision-making. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1408–1414, 2013.

[Pecora *et al.*, 2018] Federico Pecora, Henrik Andreasson, Masoumeh Mansouri, and Vilian Petkov. A loosely-coupled approach for multi-robot coordination, motion planning and control. In *Proceedings of the 28th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 485–493, 2018.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[Scharpff *et al.*, 2015] Joris Scharpff, Diederik M. Roijers, Frans A. Oliehoek, Matthijs T. J. Spaan, and Mathijs de Weerdt. Solving multi-agent MDPs optimally with conditional return graphs. In *Proceedings of the 10th AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains (MSDM)*, 2015.

[Younes and Simmons, 2004] Håkan L. S. Younes and Reid G. Simmons. Solving generalized semi-Markov decision processes using continuous phase-type distributions. In *Proceedings of the 19th AAAI Conference on Artificial Intelligence (AAAI)*, pages 742–747, 2004.

[Ziparo *et al.*, 2011] Vittorio Amos Ziparo, Luca Iocchi, Pedro U. Lima, Daniele Nardi, and Pier Francesco Palamara. Petri net plans – A framework for collaboration and coordination in multi-robot systems. *Autonomous Agents and Multi-Agent Systems*, 23(3):344–383, Nov 2011.