Multi-Robot Planning Under Uncertain Travel Times and Safety Constraints

Masoumeh Mansouri[1]* Bruno Lacerda[2]* Nick Hawes[2] Federico Pecora[1]

***Abstract*— We present a novel modelling and planning approach for multi-robot systems under uncertain travel times, based on the use of generalised stochastic Petri nets (GSPNs). Our approach allows for the specification of requirements on team behaviour, including safety constraints and rewards. The GSPN is interpreted as a Markov decision process (MDP) for which we can generate policies that optimise the high-level requirements. This representation is more compact than the equivalent multi-agent MDP, allowing us to scale better. Furthermore, we describe how the integration of the GSPN with a lower-level team controller allows for accurate expectations on team performance. We evaluate our approach based on an industrial scenario where it outperforms hand-crafted policies used in current practice.**

I. 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 members of a robot team jointly navigating in a shared space introduces further unmodelled dynamics deriving from their interaction. 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 [1]. Whilst these are reliable, every new environment or application requires significant bespoke engineering, and the resulting system provides no *formal guarantees on team behaviour*. To address this we propose an automated planning method that can be used to replace current practice.

More specifically, we propose a novel method for deriving policies that regulate the behaviour of individual robots in accordance to high-level requirements on the overall behaviour of the team. The team is modelled as a generalised stochastic Petri net (GSPN) [2], 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 [3], an MR-GSPN can be interpreted as a Markov decision process (MDP) [4]. 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 learned from simulations of the team navigating in the target environment. As a consequence, the policies obtained from the MR-GSPN MDP 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.

Our main contributions are the the introuction of (i) MR-GSPN, a novel GSPN based modelling formalism for multi-robot teams; and (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. 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.

II. Related Work

The clear semantics of concurrency in Petri nets (PNs) has led to their prior use in robotics. For example, [5] use PNs to create single-robot behaviour models which support robust execution strategies. PNs have been used previously to model the behaviour of multi-robot systems. [6] used a PN model to synthesise a supervisory controller for a team of soccer playing robots. [7] 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.

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 [8] and instead rely on lower level control to provide robustness

*The authors contributed equally to this work.

[1]Örebro University, Sweden, <name>.<surname>@oru.se

[2]Oxford Robotics Institute, University of Oxford, UK, {bruno,nickh}@robots.ox.ac.uk

against execution time variations from planned behaviour [1]. Multi-agent MDPs [9] 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 [10], [11]. 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. [11] 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 for the team to complete discrete actions (leading to inefficient robot behaviour).

III. GSPNs for Multi-robot Path Planning

Modelling. A Petri Net (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 given arc weight functions. For a marking M, transition t is said to be *enabled* in M if each place p holds at least as many tokens as the ones to be consumed by t. A transition t enabled in a marking M can *fire*, resulting in the marking M' where t consumes and produces tokens according to W^- and W^+.

GSPNs are an extension of PNs where transitions are partitioned into *immediate* and *exponential* transitions. More precisely, a GSPN is tuple $N = \langle P, T, W, \overline{M}, \lambda \rangle$, where P, T, 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 represent actions available to a control policy. Our approach essentially boils down to *finding transition firings that optimise some objective*. As explained below, this objective encodes a goal specification that is expressed in terms of *firing rewards* and *markings that should be avoided*.

Now, 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.

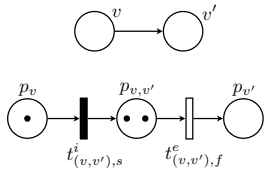

Fig. 1. Example of construction of a MR-GSPN. Top: two nodes v and v' connected by an edge in a navigation graph; Bottom: The fragment of the MR-GSPN representing navigation between v and v', where $v \in V^i$. Note that if $v \in V^e$, then both depicted transitions would be exponential. The depicted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

By splitting navigation between two locations, v and v', into two transitions, with the first transition representing the triggering of the navigation action to move from v to v' and the second the expected duration of the navigation action, we can represent the above as a GSPN. This construction yields what we call a Multi-robot GSPN (MR-GSPN) and is illustrated in Fig. 1. The key point is that via GSPNs, we can *represent robots as tokens*. We also partition V into V^i and V^e. If $v \in V^i$, then this transition is immediate and is under our control. If $v \in V^e$, then this transition is exponential, with a rate 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.

Goal Specification. A goal specification is defined in two parts. First, we define the subset of markings satisfying linear constraints over the markings of an MR-GSPN. Among these, we assume there to be a special constraint C_1 requiring that the number of tokens in places representing nodes for which the robots undergo an external process must be maintained above a bound b. 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 the MR-GSPN. In these race conditions there is some probability of tokens from places associating to V^e being removed. Thus, it is not possible to indefinitely keep the MR-GSPN in markings that satisfy this special constraint. The set of markings in which this constraint is not satisfied is a set of *failure markings*.

For the second part of the goal specification, we consider a *transition firing reward* $r_{T^i} : T^i \to \mathbb{R}_{\geq 0}$, representing the utility of firing certain 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 starting to move after dropping its passengers at its destination). Our goal is to find a mapping from all possible markings to T that maximises the expected cumulative value of r_T until a failure marking is reached. Note that in any infinite run the system will eventually reach a failure marking, because C_1 will

eventually stop being satisfied. This means that we assume that, regardless of what we do, the system will eventually fail. Our goal is to gather as much reward as possible before it does so.

IV. MDPs as Planning Models for MR-GSPNs

Taking an MA-based semantincs [3], we interpret the GSPN as an MDP for the purpose of computing policies that maximize firing reward while adhering to the constraints over markings. The MDP represents the possible evolutions of the state of the system: in each state and discrete time step t, any of the enabled actions at time t can be selected, and the system evolves to a successor state according to the values of the MDP's transition function. States represent possible markings of the modelled MR-GSPN, and the transition function captures the possible transitions among them.

We consider *deterministic and stationary policies* that maximise the *expected cumulative value of a reward structure*. The goal specification over the MR-GSPN is reduced to the problem of finding policies that maximise an expected reward until a set of unavoidable states is reached, which correspond to the failure markings in the MR-GSPN. Furthermore, a reward structure is defined so as to reflect the triggering reward mentioned above. By first preprocessing the MDP, replacing transitions from failure states with zero-reward self-loop transitions, we effectively reduce the problem to an *infinite horizon cumulative reward maximisation problem*, which can be solved with standard techniques such as value iteration. In the resulting MDP, we can choose an enabled immediate action to fire. Also, 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 certain decision.

V. Simulation-Based Duration Estimates

In order to robustly maintain the safety specification, we require accurate models of the durations of navigation on the edges of the navigation graph G. Obtaining such models is challenging as the precise dynamic model of a robot is often unknown. Moreover, interactions among multiple robots moving in a shared environment affect durations (e.g., robots yielding to, or avoiding each other). Other sources of uncertainty in the environment further complicate the computation of duration models. 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 low-level robot control [1]. This controller is specifically designed to support external injection of navigation choices for robots. Given these choices, the 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.

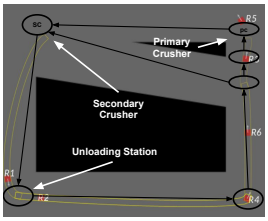

Fig. 2. Evaluation environment. Robots are red, planned paths yellow. The navigation graph is overlayed with key locations indicated.

VI. Evaluation

We evaluate the quality of the obtained policies given varying environmental dynamics. 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. Also, robots have a limited capacity (4 metric tons of gravel), and are thus *required to leave the PC when full*. The secondary crusher (SC) does not have these constraints, as the gravel produced there is loaded onto haulers manually. 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.

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

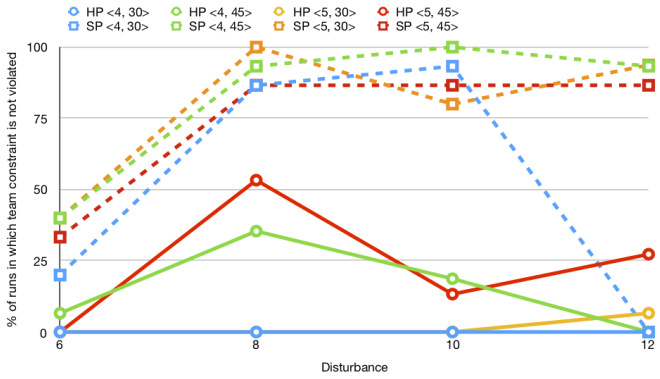

Fig. 3. Success rate for HP and SP for different disturbance profiles, in 8 problems. Legend is of the form $< n, PC >$.

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 and less than 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. Also our experiments show that 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.

VII. Discussion

We finish with a brief discussion on the problem tackled in this paper, the assumptions made and advantages of using MR-GSPNs. We 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. One typical 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) [9], 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, there is an issue with scalability. Even without considering uncertain durations, given a navigation graph with k nodes and n robots, we need at least n^k 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.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).

Finally, by using a GSPN, an event based model, we bypass the need for synchronisation: triggered immediate transitions correspond to sending navigation commands to robots; as they navigate and change state they inform the control policy and the state is immediately updated by firing the corresponding exponential transition and updating state accordingly. This allows for smooth and asynchronous execution of the policies.

Acknowledgments. This work is 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

[1] F. Pecora, H. Andreasson, M. Mansouri, and V. 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)*, 2018.

[2] G. Balbo, "Introduction to generalized stochastic Petri nets," in *Proc. ot the 7th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM)*, 2007.

[3] C. Eisentraut, H. Hermanns, J.-P. Katoen, and L. Zhang, "A semantics for every GSPN," in *Proc. of the 34th Int. Conf. on Applications and Theory of Petri Nets and Concurrency*. Springer, 2013.

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

[5] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara, "Petri net plans – A framework for collaboration and coordination in multi-robot systems," *Autonomous Agents and Multi-Agent Systems*, vol. 23, no. 3, pp. 344–383, Nov 2011.

[6] B. Lacerda and P. U. Lima, "Designing Petri net supervisors from LTL specifications," in *Proceefings of Robotics: Science and Systems VII (RSS)*, 2011.

[7] C. Mahulea and M. Kloetzer, "Robot planning based on boolean specifications using Petri net models," *IEEE Transactions on Automatic Control*, vol. 63, no. 7, pp. 2218–2225, July 2018.

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

[9] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in *Proceedings of the 6th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*, 1996.

[10] J. Scharpff, D. M. Roijers, F. A. Oliehoek, M. T. J. Spaan, and M. de Weerdt, "Solving multi-agent MDPs optimally with conditional return graphs," in *Proc. of the 10th AAMAS Workshop on Multi-Agent Sequential Decision Making in Uncertain Domains*, 2015.

[11] F. Faruq, B. Lacerda, N. Hawes, and D. Parker, "Simultaneous Task Allocation and Planning Under Uncertainty," in *Proc. of the 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.