Petri net approach for deadlock and collision avoidance in robot planning

Marius Kloetzer, Cristian Mahulea and J.M. Colom

Abstract

This paper considers the problem of supervising the motion of some mobile robots that evolve in the same environment. The solution consists of a strategy of moving and stopping the robots such that no collisions or deadlocks appear. The problem is formulated on a finite-state representation, some regions of the environment have a limited capacity in terms of number of robots that can simultaneously occupy them, and a set of possible trajectories is available for each robot. The solution comprises the construction of a specific Petri net model for the available trajectories, and the use of resource-allocation techniques based on restricted-capacity regions and on deadlock-free execution.

I. INTRODUCTION

The problem of planning the motion of a mobile robot continues to receive a fair amount of attention. The robotic tasks can vary from classical navigation missions, where the robot should reach a target position while avoiding collisions with existing obstacles [1], [2], to specifications given in a high-level, human-like language [3], [4]. The assumed robot's dynamics range from fully-actuated point robots to non-holonomic mobile agents [5]–[7].

Many studies aim to extend the available algorithms towards planning a whole team of robots such that a global specification is satisfied. A lot of these results are based on abstracting the environment and the robot's control capabilities into a finite-state representation, and on using available computational techniques for such representations [8]–[10]. When planning the motion

M. Kloetzer is with Department of Automatic Control and Applied Informatics, Technical University "Gh. Asachi" of Iasi, Romania {kmarius@ac.tuiasi.ro}. C. Mahulea and J.M. Colom are with Department of Computer Science and Systems Engineering, Univ. of Zaragoza, Spain {cmahulea,jm@unizar.es}

At University of Zaragoza, this work has been partially supported by CICYT - FEDER projects DPI2010-20413 and TIN2011-27479-C04-01.

of a team of robots, the focus usually falls on ensuring a specific task, and in this endeavor various assumptions might appear. Among such assumptions, collision avoidance may be either ignored (e.g., by assuming point robots or local avoidance rules), or is may be avoided by omitting all dangerous solutions. Some researches analyze possible robot behaviors, but without modifying the current execution [11].

This paper proposes a method for avoiding inter-robot collisions for a given set of possible trajectories for each robot. These trajectories may appear either from the solution of planning a team of robots while ignoring collisions among agents, or simply from having multiple independent robots that act (based on single-robot algorithms) in a common environment, without any collaboration. The collision avoidance is guaranteed by waiting modes (pausing the movement of some robots in specific locations). However, such simple waitings can lead to deadlock, and therefore we design a supervising controller that guarantees a live evolution.

The first part of the paper describes, in an intuitive way, the procedure of abstracting the trajectories of the robots to a Petri net. Petri nets have been used for modeling and control of mobile robots in the recent literature [12]–[14]. The methodology we use is similar to the abstraction of manufacturing systems into Resource Allocation Systems (RAS) and is using PNs instead of finite automata as in [15]. The methodology consists in two steps: (1) the sequence of the regions that should be traveled by a robot along each trajectory is modeled as a state machine Petri nets (PN) where each place represents the presence of the robot in a given region (these places are called *trajectory places*), and (2) the capacities of the regions are modeled by using some *resource places*, such that a resource is *allocated* when the robot enters in a specific region and it is *released* when the robot leaves that region. Notice that each trajectory place requires only one resource (the one modeling the capacity of that region), because we assume that a robot can occupy only one region at any moment. The PN model will have a behavior equivalent to a linear S^3PR ($L - S^3PR$) because trajectories are chosen before the beginning of any movement.

In the last part of the paper, we review the structural liveness enforcement techniques of RAS modeled by PN and we interpret them in robot planning terms. In literature, the most studied methods for liveness enforcement consist in two steps: (1) the computation of the minimal bad siphons, and (2) for each minimal bad siphon, add a *monitor place* ensuring that the siphon cannot be emptied. However, the monitor places can introduce new bad siphons in the model and

the procedure should be iteratively executed. Unfortunately, after the introduction of a monitor place, the PN will not be anymore a subclass of S^3PR (will be a S^4PR) and the computation of the minimal bad siphons requires more elaborated algorithms than those used for S^3PR.

Petri nets have been used for modeling and control of mobile robots in the recent literature [12]–[14]. In [13], simulation is used to study some qualitative and quantitative properties of the model. In [12], [14], Stochastic Petri net models are considered for modular modeling and analyzing of robot tasks. Different models are proposed for environment layer and action executor layer.

In this paper the approach is rather different. First, we consider mobile robots evolving in an environment split in regions (for example using a cell decomposition algorithm). Second, we try to use the structural properties of the obtained model to study the properties of the robot trajectories.

II. PRELIMINARIES

A. Environment Abstraction

Let us consider a set of $|R|$ identical robots evolving in a given environment, with the purpose of accomplishing a given mission. For simplicity of exposition, we assume that the motion of a robot in the environment is abstracted into a finite graph given in definition 1.

Definition 1. *A graph abstracting the environment is a tuple $G = (Q, E, cap)$, where:*

- *Q is a finite set of nodes representing locations (or regions) in the environment where robots can travel;*
- *$E \subseteq Q \times Q$ is the set of edges, where $(q, q') \in E$ means a robot starting from location q reach q' without going through other regions from $Q \setminus \{q, q'\}$;*
- *$cap : Q \to \mathbb{N}_+$ is a map illustrating the capacity of each location, with $cap(q)$ showing the maximum number of robots that can be at any given time in region q, $\forall q \in Q$* [1].

We mention that such finite abstractions are used for solving various planning problems for mobile robots. For example, in [9], [16], [17] finite state representations are constructed by cell decomposition techniques, where the free state space in the bounded environment (the space not

[1]We will say that location (or region) $q \in Q$ is *restricted* if $cap(q) < |R|$ and *unrestricted* otherwise.

covered by obstacles) is partitioned into a number of cells having the same geometrical shape. Based on control design techniques for classes of systems in particular partitions [18], [19], the adjacency relations among cells and the robot dynamics yield the possible set of edges linking adjacent cells (or transitions in a so called transition system model). In [6], [20] the environment in which a car-like robot evolves is partitioned into a set of curved-surface regions, based on the constrained steering capabilities. In other situations, the environment can directly yield a finite state representation, as in [8] (where a city-like environment is abstracted by considering each intersection as a node and each road linking two intersections as an arc), or in [4], where a house-like environment can be easily abstracted based on its rooms and on the existing doors.

In the case of a single robot evolving in the environment, many automated techniques use the abstraction for finding a solution to a given task, the possible tasks ranging from reaching a target while avoiding obstacles to complex missions expressed in human-like language with the help of regular expressions or temporal logics. Some studies extend these classes of specifications to a team of collaborating robots, of course with the price of an increased computation complexity and of more restrictive specifications.

The goal of this paper is not to design robotic paths (sequences of nodes from Q to be visited), but it is to ensure that some given trajectories are followed such that the environment restrictions are satisfied (e.g. capacities of regions) and deadlocks do not occur. Therefore, the assumed robots can either be independent, each of them solving a given task in the same environment, or they can collaborate for a mission requiring more than one robot. In either case, the inputs of our problem are the environment representation in the form G and one or more possible trajectories for each robot. Such inputs are enabled by a wide range of approaches, some of them being referred above.

B. Petri Net Models

In this subsection we introduce the basic definition of Petri nets (for a more detailed introduction we refer the reader to [21]).

Definition 2. *A Petri net (PN) is a tuple $\mathcal{N} = \langle P, T, F \rangle$, where P and T are two non-empty disjoint sets of places and transitions, and the set $F \subseteq (P \times T) \cup (T \times P)$ is the* incidence flow relation.

For a node $h \in P \cup T$, the sets of its input and output nodes are denoted as ${}^\bullet h$ and $h^\bullet$, respectively. Formally, ${}^\bullet h = \{v \in P \cup T | (v,h) \in F\}$ and $h^\bullet = \{v \in P \cup T | (h,v) \in F\}$. This notion can be naturally extended to a set of nodes as follows: given $H \subseteq P \cup T$, ${}^\bullet H = \bigcup_{h \in H} {}^\bullet h$ and $H^\bullet = \bigcup_{h \in H} h^\bullet$.

A self-loop free ($\forall h \in P \cup T$, ${}^{\bullet\bullet}h \cap h = \emptyset$) Petri net $\mathcal{N} = \langle P, T, F \rangle$ can alternatively be represented as $\mathcal{N} = \langle P, T, \boldsymbol{C} \rangle$, where $\boldsymbol{C}$ is the net token flow matrix, $\boldsymbol{C} = \boldsymbol{C}^+ - \boldsymbol{C}^-$, with: $\boldsymbol{C}^+[p,t] = 1$ if $(t,p) \in F$ and $\boldsymbol{C}^+[p,t] = 0$ otherwise; $\boldsymbol{C}^-[p,t] = 1$ if $(p,t) \in F$ and $\boldsymbol{C}^+[p,t] = 0$ otherwise.

Let p_i, $i = 1, \ldots, |P|$ and t_j, $j = 1, \ldots, |T|$ denote the places and transitions. Each place can contain a non-negative integer number of tokens, this number representing the *marking* of the place. The distribution of tokens in places is denoted by vector $\boldsymbol{m}$, where $\boldsymbol{m}[p_i]$ (or simply m_i) is the marking of place p_i. The initial token distribution, denoted by $\boldsymbol{m}_0 \in \mathbb{N}^{|P|}$, is called the initial marking of the net. A PN with an initial marking is a PN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$. The enabling degree of a transition t_j at a marking $\boldsymbol{m}$ is given by $enab(t_j, \boldsymbol{m}) = \min_{p_i \in {}^\bullet t_j} \left\{ \left\lfloor \frac{\boldsymbol{m}[p_i]}{\boldsymbol{C}^-[p_i,t_j]} \right\rfloor \right\}$, which represents the maximum amount in which t_j can fire.

A transition $t_j \in \boldsymbol{T}$ is enabled at $\boldsymbol{m}$ if $enab(t_j, \boldsymbol{m}) > 0$. An enabled transition t_j can fire in any integer amount α, with $0 < \alpha \leq enab(t_j, \boldsymbol{m})$, leading to a new state $\boldsymbol{m'} = \boldsymbol{m} + \alpha \cdot \boldsymbol{C}[\cdot, t_j]$, where and $\boldsymbol{C}[\cdot, j]$ is the j^{th} column of matrix $\boldsymbol{C}$. It will be said that $\boldsymbol{m}'$ is a *reachable marking* that has been reached from $\boldsymbol{m}$ by firing t_j and this firing is denoted by $\boldsymbol{m}\,[t_j\rangle \boldsymbol{m'}$. The set of all reachable markings from $\boldsymbol{m}_0$ is denoted by $\mathcal{R}\,(\mathcal{N}, \boldsymbol{m}_0)$.

A PN is *strongly connected* if from each node $h \in P \cup T$ there exists a direct path to any other node $v \in P \cup T$. $\mathcal{N}$ is a *state machine* if all transitions have at most one input and at most one output place, i.e., $\forall t \in T$, $|{}^\bullet t| \leq 1$ and $|t^\bullet| \leq 1$.

A set of places $S \subseteq P$ is a *siphon* if ${}^\bullet S \subseteq S^\bullet$ (the set of input transitions is included in the set of output transitions) and it is a *trap* if $S^\bullet \subseteq {}^\bullet S$ (the set of output transitions is included in the set of input transitions).

The Resource Allocation System (RAS) analysis requires models retaining only those aspects relevant to the representation of the resource allocation. Different applications of RAS in terms of PNs have been considered in literature in many domains [22]–[25], leading to several Petri net subclasses. A broad perspective of Petri nets classes used for RAS abstraction of discrete event systems is given in [26]. The subclass of S^3PR is one of the most widely studied and

the liveness characterization has been studied in terms of some siphons of the net [22], [24]. In this paper we abstract the trajectories of a set of robots to a RAS whose model belongs to the class of *linear* S^3PR ($L-S^3PR$) class. This class has been defined in [27] and many structural properties are studied. In the following the definition of this class is given.

Definition 3. *A linear S^3PR is a PN $\mathcal{N}=\langle P,T,F\rangle$ such that:*

1) $P=P_A\cup P_R\cup P_0$ *is a partition such that:*

a) $P_0=\{p_0^1,\ldots,p_0^k\}$, $k>0$ *($p\in P_0$ is called an* idle *place)*

b) $P_A=\bigcup_{i=1}^k P_A^i$, *where* $P_A^i\cap P_A^j=\emptyset$, *for all* $i\neq j$ *($p\in P_A$ is called an* operation *or* activity *place);*

c) $P_R=\{\bar{r}_1,\ldots,\bar{r}_n\}$, $n>0$ *($\bar{r}\in P_R$ is called a* resource *place).*

2) $T=\bigcup_{i=1}^k T^i$, *where* $T^i\cap T^j=\emptyset$, *for all* $i\neq j$.

3) $\forall i\in\{1,\ldots,k\}$ *the subnet $\mathcal{N}^i$ generated by $\{p_0^i\}\cup P_A^i\cup T^i$ is a strongly connected state machine such that every cycle contains $\{p_0^i\}$ and* $\forall p\in P_A^i$, $|p^\bullet|=1$.

4) $\mathcal{N}$ is strongly connected.

5) $\forall i\in\{1,\ldots,k\}$, $\forall p\in P_A^i$, ${}^{\bullet\bullet}p\cap P_R=p^{\bullet\bullet}\cap P_R$ *and* $|{}^{\bullet\bullet}p\cap P_R|=1$.

Definition 4. *Let $\mathcal{N}=\langle P_A\cup P_R\cup P_0,T,F\rangle$ be a $L-S^3PR$. An initial marking $\boldsymbol{m}_0$ is called admissible if:*

- $\boldsymbol{m}_0[p]\geq 1$, $\forall p\in P_0\cup P_R$;
- $\boldsymbol{m}_0[p]=0$, $\forall p\in P_A$.

III. Robot Trajectories and Resources

After formulating the problem we're interested in, this section shows how a set of mobile robot trajectories given on the graph defined in Sec. II-A can be modeled as a Resource Allocation System (RAS) in terms of Petri nets.

A. Problem Statement

We consider an environment represented by a graph $G=(Q,E,cap)$ as in definition 1. There are $|R|$ robots (the set of robots is denoted by $R=\{r_1,r_2,\ldots,r_{|R|}\}$) deployed in this environment, and the initial position of robot r_i is denoted by $q_{0i}\in Q$, $i=1,\ldots,|R|$. Based

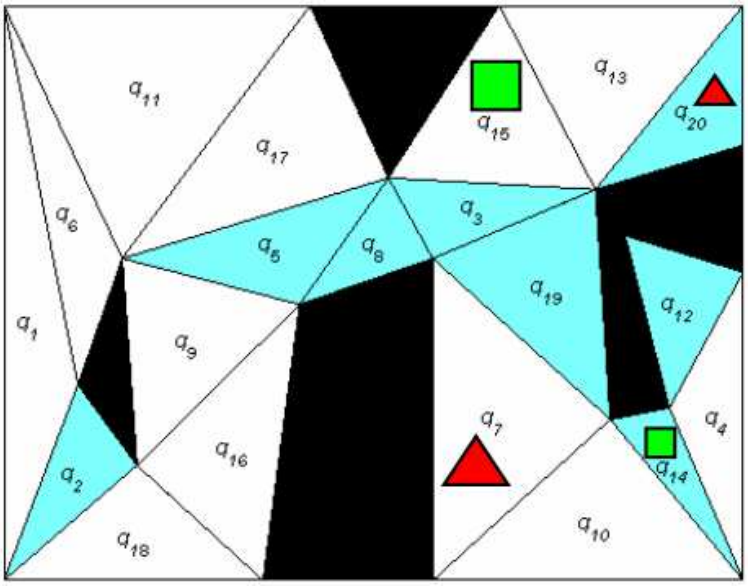

Fig. 1. A planar environment with the free space composed by 20 simplices.

on some mission requirements, we assume that for each robot r_i, $i = 1, \ldots, |R|$, we have a set $\mathcal{T}_i$ of possible finite-length trajectories, all of them starting from q_{0i}. Due to the finite state representation, each trajectory consists of a finite string of elements from Q. For example, if nodes from Q correspond to a cell decomposition of the environment, a trajectory is the sequence of cells that should be followed by a robot. The set of all trajectories for all robots in the environment is denoted as $\mathcal{T} = \bigcup \mathcal{T}_i$, with the obvious inclusion $\mathcal{T} \subset Q^*$, where Q^* denotes the Kleene closure of set Q (the set of finite-length strings that can be generated by concatenating elements from Q).

The problem solved in this paper can be formulated as follows:

Problem 1. *Given the environment modeled by $G = (Q, E, cap)$, with restricted capacity of some nodes, a set of robots $R = \{r_1, r_2, \ldots, r_{|R|}\}$ and the sets $\mathcal{T}_i$ of possible finite-length trajectories for each robot $r_i \in R$, find a deadlock prevention policy ensuring that each robot terminates its chosen trajectory.*

In problem 1, if there are more possible trajectories for a robot, it is assumed that one of them is chosen before the movement. Furthermore, the strategy will consist in finding waiting modes for each robot (pauses in its motion), and the adjective "correctly" means that no robots

collide between them and no deadlock appears due to simultaneous waitings.

Remark 1. *We assume that, in the case of a collaborative team of robots, there are no synchronization moments imposed along the robotic trajectories, i.e., there are no intermediate nodes that should be reached by the robots at the same time. If trajectories with intermediate synchronizations are necessary, they should be split in more unsynchronized trajectories, and the procedure we propose can be iterated for the resulting unsynchronized sequences.*

Remark 2. *The solution provided in this paper does not account for choosing a specific trajectory for each robot from the set of its possible trajectories such that a criterion as the number of necessary waiting moments is minimized. Such a computationally tractable extension (other than the brute force approach that tests all possible permutations of individual trajectories) is left for future research.*

For the sake of correctness in formulating problem 1, it is assumed that the initial deployment of robots in nodes q_{0i}, $i = 1, \ldots, |R|$, satisfies the maximum capacity of these nodes (given by map cap). Also, the fulfillment of a desired mission is guaranteed as long as each robot r_i *correctly* follows any trajectory from set $\mathcal{T}_i$. Thus, the sets $\mathcal{T}_i$ can represent a priori computed solution of an imposed robotic mission (e.g., by using some research results mentioned in Sec. II-A), and the procedure we propose here can be regarded as a strategy ensuring the correctness of following such a solution, under restricted capacity of environment regions.

Our approach begins with the construction of a PN model based on the robotic trajectories (Sec. III-B and III-C), such that the maximum capacities of visited nodes are satisfied during motion. Then, a deadlock prevention strategy is embedded into this PN model (Sec. IV).

B. Intuitive PN model definition

This subsection informally presents the ideas behind the PN model construction. In order to facilitate the understanding, the following example will be considered throughout this subsection.

Consider the planar environment from Fig.1, cluttered with obstacles (the black regions) and whose free space is composed by 20 regions ($Q = \{q_1, \ldots, q_{20}\}$). Assume there are 2 robots, initially placed in regions q_1 and q_2, respectively. The mission requires that the silver from q_{20} is moved to q_7, and the gold from q_{14} is deposited to q_{15}. As mentioned, the construction of robotic

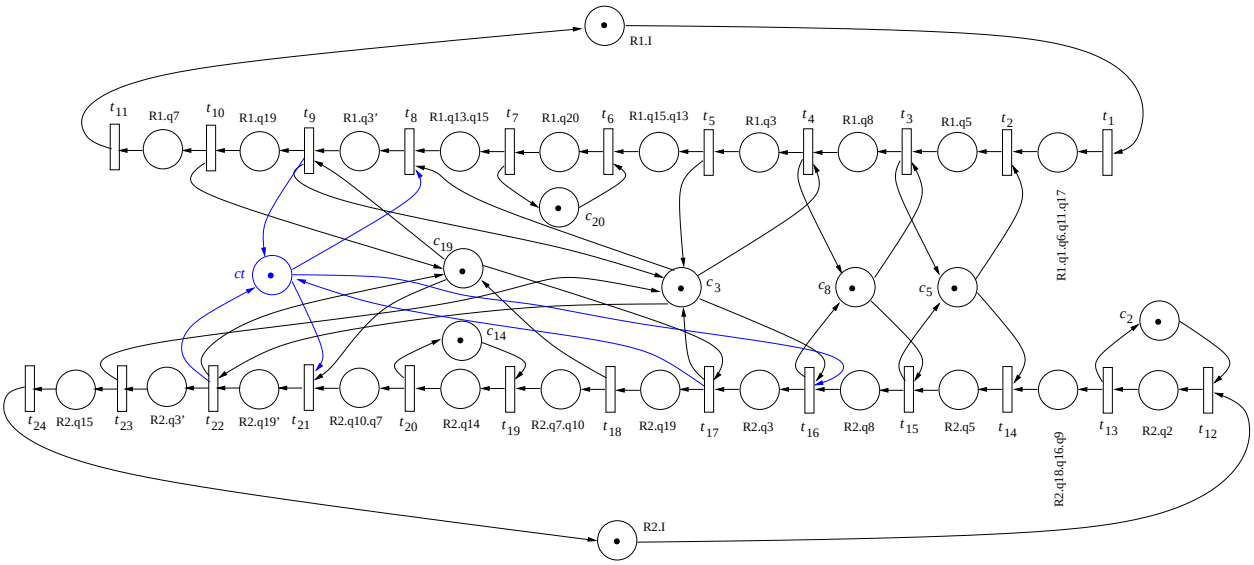

Fig. 2. The RMPN model of two robots, each following one trajectory in the environment from Fig. 1.

trajectories is not within the scope of this work, and we just mention that for this mission they can be obtained by task allocation and graph searches.

For simplicity of exposition, we consider only one trajectory for each robot, as follows:

- Robot r_1 goes through regions: q_1, q_6, q_{11}, q_{17}, q_5, q_8, q_3, q_{15}, q_{13}, q_{20}, q_{13}, q_{15}, q_3, q_{19}, q_7;
- Robot r_2 goes through regions: q_2, q_{18}, q_{16}, q_9, q_5, q_8, q_3, q_{19}, q_7, q_{10}, q_{14}, q_{10}, q_7, q_{19}, q_3, q_{15}.

The environment regions are restricted such that in the blue (gray) regions at most one robot can be at any time moment, i.e. $cap(q_i) = 1$, $\forall i \in \{2, 3, 5, 8, 12, 14, 19, 20\}$, and $cap(q_i) = 2$ otherwise (for having the map cap well-defined over set Q, we can set its value larger or equal to the number of robots for the unrestricted regions).

Obviously, if the robots individually follow their trajectories, they could collide (in q_3, q_5, q_8, or q_{19}). Even if a collision avoidance is locally used by each robot, a deadlock may occur (e.g. if r_1 is in q_3 trying to go to q_{19}, while r_2 is in q_{19} trying to go to q_3).

The PN model constructed in this section accounts only for region capacity (i.e. guaranteed collision avoidance), and the deadlock-prevention will be added in Sec. IV. The PN corresponding to the above example is sketched in Fig. 2 (excepting for the blue place ct, which is not added in this section), and its structure is explained in the following.

We begin by collapsing each sequence of successive unrestricted regions (having capacity larger or equal to $|R|$) followed by a robot into a single place of the PN model. For the given example, the place $R1.q1.q6.q11.q17$ corresponds to the first part of r_1's trajectory (regions q_1, q_6, q_{11}, q_{17}), and the same idea applies for places $R1.q15.q13$, $R1.q13.q15$, $R2.q18.q16.q9$, $R2.q7.q10$ and $R2.q10.q7$. Note that restricted regions that are not visited by more robots (as are q_2 or q_{14}) are not included in the above collapsing, because they could raise issues if additional trajectories are added to sets $\mathcal{T}_i$.

There are more places added for each restricted region (having capacity smaller than $|R|$) that appears along robotic paths: one place models the limited capacity, and the others model the region occurrence along trajectories. For example, place c_5 corresponds to the capacity of region q_5, and it initially has marking given by $cap(q_5) = 1$. Places $R1.q5$ and $R2.q5$ correspond to robots visiting region q_5, and they are connected via transitions to c_5 such that only one robot can be at a time instant in the corresponding region. Assume that when both robots are traveling towards q_5, robot r_1 arrives first and enters this restricted region. Then, one token is removed from the place corresponding to its capacity (c_5). Thus, even if r_2 arrives to q_5 while r_1 is still there, it cannot enter the region because transition t_2 cannot be fired without one token in c_5. The token to c_5 is returned (released) when r_1 exits q_5, and then r_2 can enter q_5. In other words, c_5 can be viewed as a shared resource for robots r_1 and r_2, and the marking of c_5 with its input and output transitions model the correct access to this resource. The same idea applies for more robots and restricted capacity larger than one, by simply placing the corresponding number of tokens in the place modeling the capacity.

All places that correspond to regions that appear along a trajectory are connected by transitions in their appearing order, e.g., $R1.q1.q6.q11.q17$ is connected to $R1.q5$ via t_2 (meaning that r_1 crosses from q_{17} to q_5) and so on. Note that if a region is visited more than once along the same trajectory, more places are added in the PN model. For example, there are five places corresponding to q_3: one models its limited capacity, two model its occurrence along trajectory of r_1 and two model its occurrence along trajectory of r_2.

A number of $|R|$ places model the *Idle state* of the robots: $R1.I$ and $R2.I$. One token in $R1.I$ (or $R2.I$) means that the robot 1 (or 2) is either waiting to begin its motion, or it stopped after finishing the trajectory. If there are more possible trajectories for a robot i ($|\mathcal{T}_i| > 1$), its idle place will have $|\mathcal{T}_i|$ output transitions, one for each trajectory. Choosing one of those trajectories

means firing the corresponding transition and moving the token from the idle state to a state corresponding to actual movement.

Remark 3. *Note that when a robot finishes its chosen trajectory, the token is returned to robot's idle place. However, the PN model could be used again for supervising the motion only after the robot is placed back in its initial region from the environment. Thus, when tokens are returned to all idle places, the mission is accomplished. We don't model the mission accomplishment with different places than the idle ones, because such places would be blocking, and the PN would belong to a different class (for which deadlock avoidance strategies might not be characterized).*

As a summary of the PN model constructed so far, some places correspond to the regions visited by robots along their trajectories, other places correspond to limited capacity of some regions (graph nodes), and $|R|$ idle places correspond to beginning or ending the assignment. The transitions of the PN correspond to robots entering new regions from their trajectories, and the connections between places and transitions ensure that the capacity of each visited region is satisfied.

C. Formal Definition

In this subsection we give the formal definition and construction procedure for the PN model corresponding to robot motions, and we prove that this model has a behavior equivalent to $L - S^3PR$.

Definition 5. *A robot motion PN (RMPN) is a PN $\mathcal{N} = \langle P, T, F \rangle$ such that:*

1) $P = P_T \cup P_C \cup P_I$ *is a partition such that:*
   - *a)* $P_I = \left\{p_I^1, \ldots, p_I^{|R|}\right\}$, $|R| > 0$ *($p_I^i \in P_I$ is called the idle place of robot r_i);*
   - *b)* $P_T = \bigcup_{i=1}^{|R|} P_T^i$, *where* $P_T^i \cap P_T^j = \emptyset$, *for all* $i \neq j$ *($p \in P_T^i$ is modeling the presence of the robot r_i in a region from one of its possible trajectories);*
   - *c)* $P_C = \{c_i$ *corresponding to* $q_i | q_i \in Q$ *and* $cap(q_i) < |R|\}$.

2) $T = \bigcup_{i=1}^{|R|} T^i$, *where* $T^i \cap T^j = \emptyset$, *for all* $i \neq j$.

3) $\forall i \in \{1, \ldots, |R|\}$ *the subnet $\mathcal{N}^i$ generated by $\{p_I^i\} \cup P_T^i \cup T^i$ is a strongly connected state machine such that every cycle contains $\{p_I^i\}$ and* $\forall p \in P_T^i$, $|p^\bullet| = 1$.

4) $\mathcal{N}$ *is strongly connected.*

5) $\forall i \in \{1, \ldots, k\}$, $\forall p \in P_T^i$, ${}^{\bullet\bullet}p \cap P_C = p^{\bullet\bullet} \cap P_C$ *and* $|{}^{\bullet\bullet}p \cap P_C| \leq 1$.

The initial marking should be admissible.

Definition 6. *Let* $\mathcal{N} = \langle P_T \cup P_C \cup P_I, T, F \rangle$ *be a RMPN. An initial marking* $\boldsymbol{m}_0$ *is called admissible if:*

- $\boldsymbol{m}_0[p_I^i] = 1$, $\forall p_I^i \in P_I$*;*
- $\boldsymbol{m}_0[p_c] = cap(q_i) \geq 1$, $\forall p_c \in P_C$ *where* p_c *is modeling the capacity of region* q_i*;*
- $\boldsymbol{m}_0[p] = 0$, $\forall p \in P_T$.

In the RMPN in Fig. 2, $P_I = \{R1.I, R2.I\}$, $P_T^1 = \{R1.q1.q6.q11.q17$, $R1.q5$, $R1.q8$, $R1.q3$, $R1.q15.q13$, $R1.q20$, $R1.q13.q15$, $R1.q3'$, $R1.q19$, $R1.q7\}$, $P_T^2 = \{R2.q2$, $R2.q18.q16.q9$, $R2.q5$, $R2.q8$, $R2.q3$, $R2.q19$, $R2.q7.q10$, $R2.q14$, $R2.q10.q7$, $R2.q19'$, $R2.q3'$, $R2.q15\}$; $P_C = \{c_2$, c_3, c_5, c_8, c_{14}, c_{19}, $c_{20}\}$.

Notice that some places belonging to set P_T are not holder of resource places. But this is only a simplification of the model because we can add a resource place as those from P_C but with an initial marking equal to the number of robots. Observe that in terms of Petri nets, this is an *implicit place* [28] that can be removed. In effect, any resource place in $L - S^3PR$ is structurally implicit, i.e., its row in the incidence matrix is a linear combination of other places $\boldsymbol{C}[c_i, \cdot] = \sum_{p \in (P_I \cup P_T) \setminus \{p_i | p_i \text{ is a cell with capacity } c_i\}} \boldsymbol{C}[p, \cdot]$ and the place becomes implicit with an initial marking greater than or equal to the sum of the initial markings of the idle places, i.e., the number of robots $|R|$.

Proposition 1. *Let* $\mathcal{N} = \langle P, T, F \rangle$ *be a RMPN with an admissible initial marking. Its behavior is equivalent to a* $L - S^3PR$ *net system.*

Proof: (Sketch) Given $\mathcal{N}$*, we can add a place* c_i *for each unrestricted cell. For any* $p \in P_T$ *that represents the visit of cell* i *by the robot* j*, add the following arcs:*

1) From c_i *to each transition in* ${}^{\bullet}p$*;*

2) From each transition in $p^{\bullet}$ *to* c_i*.*

Add $|R|$ *tokens to* c_i*. The place* c_i *is implicit as previously commented. But the resulted net is a* $L - S^3PR$ *with the same set of firing sequences than the original net. Therefore, they are equivalent (implicit places preserve the language of firing sequences [28]).* ■

Given the environment graph G, the set of robots R and for each robot r_i the set of possible trajectories $\mathcal{T}_i$, $i = 1, \ldots, |R|$, Algorithm 1 can be used for obtaining the RMPN model. The steps of Algorithm 1 follow the ideas from Sec. III-B: each trajectory of each robot is modeled by a state machine (lines 7-13), the successive places corresponding to unrestricted regions from trajectory are collapsed into one (lines 15-18), and the state machine is connected to robot's idle place (line 14). Then, the capacity of each encountered restricted region is modeled by a place which is connected to all transitions corresponding entering and exiting that region (lines 19-25). The initial marking of RMPN model contains one token in each idle place (line 4) and a corresponding number of tokens in each place modeling the capacity of restricted regions (line 24). In Algorithm 1, for a place p that models the presence of a robot into a region, the corresponding location from Q is denoted by q_p.

IV. Deadlock Prevention Based on PN Model

In general, liveness of a Petri net model is a difficult property to check. Siphons are structural elements that are related to this property. According to their definition, if the places of a siphon become empty during the evolution, it is not possible to mark them again, since the set of their input transitions is included into the set of their output transitions.

If we want to ensure the liveness of a $L - S^3PR$ it is sufficient to compute all minimal bad siphons and then add a controller (a monitor place) to ensure that these siphons are not emptied. The problem lies in the computational complexity of determining the siphons, which is very high for arbitrary PNs. However, in the case of the subclass of $L - S^3PR$, these siphons are strongly related to the concept of circular wait of resources [29].

Definition 7. *Let $\mathcal{N} = \langle P_T \cup P_C \cup P_I, T, F \rangle$ be a RMPN. A* circuit of resources *is a non-empty path t_1, c_1, t_2, c_2, …, t_m, c_m with $t_i \in T$, $c_i \in P_C$, $\forall i \in \{1, 2, \ldots, m\}$, such that:*

- $c_i \in t_i^\bullet$, $\forall i \in \{1, 2, \ldots, m\}$;
- $c_i \in {}^\bullet t_{i+1}$, $\forall i \in \{1, 2, \ldots, m-1\}$;
- $c_m \in {}^\bullet t_1$.

For example, t_9, c_3, t_{22}, c_{19} is a circuit of resources of the Robot Motion PN (RMPN) in Fig. 2. Let S_C denote the set of places belonging to the circuit of resources (in this particular case, $S_C = \{c_3, c_{19}\}$) and let $S_T = \bigcup_{t \in {}^\bullet S_C} \{p \in P_T | p \in {}^\bullet t, {}^\bullet t \cap S_C = \emptyset\}$ (in the mentioned

Algorithm 1: Obtain the RMPN model

Input: G, R, $\mathcal{T}_1, \ldots, \mathcal{T}_{|R|}$; /* environment abstraction, robots and all possible trajectories */

Output: $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$; /* the RMPN system */

Set $P_C = \emptyset$

for $r_i \in R$ **do**

  Add a place $P_I^i = \{p_I^i\}$ modeling the Idle state of r_i

  Set initial marking of p_I^i equal to 1

  Set $P_T^i = T^i = \emptyset$

  for $\sqcup_j \in \mathcal{T}_i$ **do**

    Assume $\sqcup_j = q_1^j, q_2^j, \ldots, q_{|\sqcup_j|}^j$

    $\bar{P} = \{p_1^j, p_2^j, \ldots p_{|\sqcup_j|}^j\}$

    $\bar{T} = \{t_1^j, t_2^j, \ldots, t_{|\sqcup_j|}^j\}$

    ${t_k^j}^\bullet = \{p_k^j\}, \forall k = 1, 2, \ldots, |\sqcup_j|$

    ${}^\bullet p_k^j = t_k^j, \forall k = 1, 2, \ldots, |\sqcup_j|$

    ${p_k^j}^\bullet = \{t_{k+1}^j\}, \forall k = 1, 2, \ldots, |\sqcup_j| - 1$

    ${}^\bullet t_{k+1}^j = p_k^j, \forall k = 1, 2, \ldots, |\sqcup_j| - 1$

    Add one arc from p_I^i to t_1^j, and one arc from $t_{|\sqcup_j|}^j$ to p_I^i

    while $\exists p_{h1}^j, p_{h2}^j \in \bar{P}$ *s.t.* ${}^{\bullet\bullet}p_{h2}^j = \{p_{h1}^j\} \wedge cap(q_{p_{h1}^j}) \geq |R| \wedge cap(q_{p_{h2}^j}) \geq |R|$ **do**

      $\bar{P} = \bar{P} \setminus \{p_{h2}^j\}$

      $\bar{T} = \bar{T} \setminus \{{}^\bullet p_{h2}^j\}$

      ${p_{h1}^j}^\bullet = {p_{h2}^j}^\bullet$

    for $p \in \bar{P}$ *s.t.* $cap(q_p) < |R|$ **do**

      if *capacity of region* q_p *is modeled by* $p_c \in P_C$ **then**

        Add one arc from p_c to ${}^\bullet p$, and one arc from $p^\bullet$ to p_c

        else

          Add place p_c to P_C for modeling capacity of q_p

          Set initial marking of p_c equal to $cap(q_p)$

          Add one arc from p_c to ${}^\bullet p$, and one arc from $p^\bullet$ to p_c

    $P_T^i = P_T^i \cup \bar{P}$

    $T^i = T^i \cup \bar{T}$

  Set initial markings of places P_T^i as zero

construct $\mathcal{N}$ using the set of places $\left(\bigcup_{i=1}^{|R|} P_I^i\right) \cup \left(\bigcup_{i=1}^{|R|} P_T^i\right) \cup P_C$ and the set of transitions

example, $S_T = \{R1.q3, R1.q19, R2.q19, R2.q3'\}$). The set of places $S = S_C \cup S_T$ is composing a siphon (${}^\bullet S = \{t_4, t_5, t_7, t_8, t_{14}, t_{15}, t_{17}, t_{18}\} \subseteq \{t_4, t_5, t_6, t_7, t_8, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}\} = S^\bullet$) which can be emptied.

For $L - S^3PR$, the liveness characterization we have says that the net is not live iff there exists a minimal siphon and a reachable marking under which the siphon is empty.

Let us define the set of places $C_S = S_C{}^{\bullet\bullet} \setminus S$ which is the set of places corresponding to regions restricted by places in S_C that do not belong to the siphon S. Places from C_S are the ones that could *take* all the resources, thus emptying the siphon S. In the case of the RMPN in Fig. 2, $C_S = \{R1.q3', R2.q3, R2.q19'\}$. By adding a place ensuring that the marking of all places in C_S is at most $|S_C| - 1$ (i.e., the number of resources minus one), it will be impossible to empty the siphon. In our example, by limiting the markings of places $R1.q3', R2.q3, R2.q19'$ to 1, the deadlock will not be possible anymore. The solution consists in adding the place ct in the net in Fig. 2, with initial marking equal to 1, and connect it to places in C_S in the following way: for each place $p \in C_S$, add an arc $(ct, {}^\bullet p)$ and one arc $(p^\bullet, ct)$.

Approaches based on siphon computation. For the S^3PR, any deadlock state is associated with at least one minimal empty siphon. Therefore, is important to have algorithms to compute the set of all minimal siphons. Two kinds of approaches exist: the first one is based on *Integer Linear Programming* (ILP) [30] while the second one is based on the *pruning graph* [23]. Moreover, in [30] the deadlock prevention is solved by an iterative algorithm. However, in the case of $L - S^3PR$ we cannot use the relation between the circuits of resources and minimal siphons to provide faster algorithms, because after the first iteration, when at least one monitor place is introduced, the net can go out from the subclass, becoming S^4PR. This means that for analysis purposes we will use the simplest liveness characterization due to the class $L - S^3PR$, but for synthesis purposes we must use the techniques of S^4PR because the iterative nature of the method is closed in the class S^4PR.

In the example from Sec.III-B, the monitor place ct (drawn in blue/grey) acts like a semaphore that prohibits the deadlock situation of having r_1 in q_3 and heading to q_{19}, and r_2 in q_{19} and heading to q_3. This place is computed from the siphon $S = \{c_3, c_{19}, R1.q3, R1.q19, R2.q19, R2.q3'\}$ and with an initial marking containing one token less than the initial tokens in the siphon. The supervising strategy based on the RMPN model allows robots to enter new regions if and only if the corresponding transitions are possible from the current PN marking, and otherwise pauses

the motion of corresponding robots. The inclusion in the PN model of capacity restrictions and monitor places guarantees that each robot eventually finishes its trajectory, without colliding and without entering indefinite waitings.

Approaches based on the addition of resources. These approaches consist in adding copies of resources of the siphon to prevent that the number of these copies goes under a dangerous value [31]. In the case of RMPN, this implies an increase in the capacity of some regions. For example, in the original RMPN of Fig. 2 (without the place c_t) by simply putting $cap(q_3) = 2$ the $L-S^3PR$ becomes live. At the level of implementation, this may be done for example by making q_3 a *two-level region*. The main advantage of this solution with respect to the previous one is that here is not necessary anymore a centralized implementation of the controller. Moreover, if one wants a complete decentralized solution, the region q_3 can be divided into two subregions, one part to be used *only* by r_1 and the other one only by r_2. A survey describing all these techniques can be consulted in [31].

V. CONCLUSIONS

This work proposes an approach for allowing correct execution of some robotic trajectories in a shared environment. The method's outcome consists in moving or pausing robots' movements along chosen trajectories, and the correctness of execution means that there are no collisions between robots or deadlocks because of simultaneous waitings. The algorithmic solution is based on the construction of a $L - S^3PR$ PN model that embeds the possible robot trajectories and the limitations on capacity of some environment regions. Two deadlock prevention techniques on this PN system are discussed, and the result yields a strategy for correctly following the trajectories.

REFERENCES

[1] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Boston: MIT Press, 2005.

[2] S. M. LaValle, *Planning Algorithms*. Cambridge, 2006, *http://planning.cs.uiuc.edu*.

[3] C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. Pappas, "Symbolic planning and control of robot motion [grand challenges of robotics]," *Robotics Automation Magazine, IEEE*, vol. 14, no. 1, pp. 61–70, 2007.

[4] B. Johnson and H. Kress-Gazit, "Probabilistic analysis of correctness of high-level robot behavior with sensor error," in *Robotics: Science and Systems*, 2011, pp. 137–144.

[5] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343–352, 2009.

[6] D. Conner, A. Rizzi, and H. Choset, "Integrating planning and control forsingle-bodied wheeled mobile robots," *Autonomous Robots*, vol. 30, no. 3, pp. 243–264, 2011.

[7] R. Cowlagi and P. Tsiotras, "Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles," *IEEE Trans. on Robotics*, vol. 28, no. 2, pp. 379–395, 2012.

[8] Y. Chen, X. Ding, A. Stefanescu, and C. Belta, "A formal approach to the deployment of distributed robotic teams," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 158–171, 2012.

[9] X. C. Ding, M. Kloetzer, Y. Chen, and C. Belta, "Automatic deployment of robotic teams," *IEEE Robotics and Automation Magazine*, vol. 18, no. 3, pp. 75–86, 2011.

[10] M. Kloetzer and C. Mahulea, "A Petri net based approach for multi-robot path planning," *Discrete Event Dynamic Systems*, 2013, in press.

[11] S. Konur, C. Dixon, and M. Fisher, "Analysing robot swarm behaviour via probabilistic model checking," *Robotics and Autonomous Systems*, vol. 60, no. 2, pp. 199–213, 2012.

[12] H. Costelha and P. Lima, "Robot task plan representation by Petri nets: modelling, identification, analysis and execution," *Journal of Autonomous Robots*, pp. 1–24, 2012.

[13] J. King, R. Pretty, and R. Gosine, "Coordinated execution of tasks in a multiagent environment," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 33, no. 5, pp. 615–619, 2003.

[14] G. Kim and W. Chung, "Navigation behavior selection using generalized stochastic Petri nets for a service robot," *IEEE Transactions on Systems, Man and Cybernetics. Part C, Applications and Reviews*, vol. 37, no. 4, pp. 494–503, 2007.

[15] S. Reveliotis and E. Roszkowska, "Conflict Resolution in Free-Ranging Multivehicle Systems: A Resource Allocation Paradigm," *IEEE Transactions on Robotics*, vol. 27, no. 2, pp. 283–296, 2011.

[16] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Hybrid controllers for path planning: A temporal logic approach," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 4885 – 4890.

[17] M. Kloetzer, C. Mahulea, and O. Pastravanu, "A probabilistic abstraction approach for planning and controlling mobile robots," in *ETFA11: 16th IEEE Conference on Emerging Technologies and Factory Automation*, Toulouse, France, 2011, pp. 1 – 8.

[18] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen, "Reachability and control synthesis for piecewise-affine hybrid systems on simplices," *IEEE Transactions on Automatic Control*, vol. 51, pp. 938–948, 2006.

[19] C. Belta and L. Habets, "Constructing decidable hybrid systems with velocity bounds," in *43rd IEEE Conference on Decision and Control*, Paradise Island, Bahamas, 2004.

[20] D. Conner, H. Kress-Gazit, H. Choset, A. Rizzi, and G. Pappas, "Valet parking without a valet," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 572–577.

[21] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.

[22] J. Ezpeleta, J. Colom, and J. Martinez, "A petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 2, pp. 173–184, Apr. 1995.

[23] E. Cano, C. Rovetto, and J. Colom, "An algorithm to compute the minimal siphons in S^4PR nets," *Discrete Event Dynamic Systems*, vol. 22, no. 4, pp. 403–428, Dec. 2012.

[24] S. Reveliotis, *Real-Time Management of Resource Allocation Systems: A Discrete Event Systems Approach*, ser.

International Series in Operations Research & Management Science. Springer Science+Business Media, Incorporated, 2005. [Online]. Available: http://books.google.es/books?id=BuYmkvo4RlkC

[25] Z. Li and M. Zhou, *Deadlock Resolution in Automated Manufacturing Systems: A Novel Petri Net Approach*, ser. Advances in Industrial Control. Springer, 2009. [Online]. Available: http://books.google.es/books?id=Phz99CpZHvsC

[26] J. López-Grao and J. Colom, *Structural methods for the control of Discrete Event Dynamic Systems – The case of the Resource Allocation Problem*, ser. Lecture Notes in Control and Information Sciences. Springer, 2013, vol. 433, ch. 13, pp. 257–278.

[27] J. Desel and M. Silva, Eds., *Application and Theory of Petri Nets 1998, 19th International Conference, ICATPN '98, Lisbon, Portugal, June 22-26, 1998, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1420. Springer, 1998.

[28] F. Garcia-Valles and J.-M. Colom, "Implicit places in net systems," in *Petri Nets and Performance Models, 1999. Proceedings. The 8th International Workshop on*, 1999, pp. 104–113.

[29] J. Ezpeleta, F. García-Vallés, and J.-M. Colom, "A class of well structured petri nets for flexible manufacturing systems," in *ICATPN*, ser. Lecture Notes in Computer Science, J. Desel and M. Silva, Eds., vol. 1420. Springer, 1998, pp. 64–83.

[30] F. Tricas, F. Garcia-Valles, J.-M. Colom, and J. Ezpeleta, "A Petri Net Structure-Based Deadlock Prevention Solution for Sequential Resource Allocation Systems," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2005, pp. 271–277.

[31] J.-P. López-Grao, J.-M. Colom, and F. Tricas, "Structural deadlock prevention policies for flexible manufacturing systems: A petri net outlook," in *Formal Methods in Manufacturing*, ser. Series on Industrial Information Technology, J. Campos, C. Seatzu, and X. Xie, Eds. CRC Press/Taylor and Francis, 2014, ch. 7, to appear.