Coordination of Cyclic Motion Processes in Free-Ranging Multiple Mobile Robot Systems

Elzbieta Roszkowska

Abstract We consider a Multiple Mobile Robot System (MMRS) viewed as a group of autonomous robots sharing a common 2D motion space. Each robot performs a mission that requires it to travel a number of times along a specific, independently planned closed path. The robots operate asynchronously and are able to control their motion with path-following algorithms that allow each of them to correctly perform its mission when alone on the stage. When sharing the motion space, the robots must refine their motion strategies in order to avoid collisions, through modification of their paths, velocity profiles or both. Following our earlier contributions, we represent MMRS as a class of RAS (Resource Allocation System) that abstracts in a discrete form the motion space and the motion processes of the robots. A model of the feasible dynamic behavior of the robot system is then obtained by mapping the distinguished RAS into a DFSA (Deterministic Finite State Automaton) that ensures collision avoidance among the robots. Based on this model, we formulate the deadlock avoidance problem, discuss its complexity, and demonstrate relevant algorithms to solve it. Finally, we propose a control architecture that implements the described control logic and combines it with the priority control, thus receiving a flexible controller for MMRS.

1 Introduction

The use of a mobile robot team in place of one robot substantially increases the performance of many robotic applications, including those related to transport, area searching, search and rescue, interplanetary exploration, extraction of minerals, or agriculture and forestry. A key issue in the design of such systems is to coordinate the movement of a number of robots operating in the same workspace. Regardless

E. Roszkowska (✉)
Department of Cybernetics and Robotics, Faculty of Electronics, Wrocław University of Science and Technology, 27 Wybrzeże Wyspiańskiego St., 50-372 Wrocław, Poland
e-mail: elzbieta.roszkowska@pwr.edu.pl

W. Bożejko and G. Bocewicz (eds.), *Modelling and Performance Analysis of Cyclic Systems*, Studies in Systems, Decision and Control 241,
https://doi.org/10.1007/978-3-030-27652-2_5

of their tasks, the robots must be able to effectively share a common area in order to prevent the mutual disruption of traffic and effectively pursue their missions.

The prevailing approach to modeling Multiple Mobile Robot Systems (MMRS) consists in the abstraction of the dynamics of the robots in time, and considering the problem of the robots' coordination over time. Earlier works mostly concentrated on motion planning with respect to collision avoidance and performance optimization. According to [17], two categories of approaches to these problems—centralized and decoupled—can be distinguished as opposite ends of the spectrum of solutions. A centralized approach typically constructs a path in a composite configuration space, which is formed by the Cartesian product of the configuration spaces of the individual robots, e.g. [1]. A decoupled approach typically generates paths for each robot independently, and then a coordination diagram is used to plan a collision-free trajectory along the paths, e.g. [3]. Most often the problem of collision-free motion planning is decomposed into two subproblems: path planning and trajectory planning. Path planning finds geometric paths that do not intersect static obstacles, and trajectory planning determines how fast each robot must move along its path to avoid collision with others.

However, the control of a multiple robot system based solely on motion planning has a significant shortcoming. At the robot coordination level, the realization of motion plans is an open-loop control policy, based on deterministic time functions. Such a control is very sensitive to the system randomness, which, given the autonomous and asynchronous operation of the robots, makes the eventual applicability of these open-loop control plans highly questionable.

Therefore, more contemporary solutions use algorithms that calculate robot coordination decisions online, taking into account dynamic models of the robots and information about their current state. Two concepts representative for this approach are Reciprocal Collision Avoidance [16] and the Potential Fields [4]. However, although very effective locally, these methods cannot be easily adapted for the synthesis of the required global system behavior. The continuous-time abstraction used to describe a single robot, when applied to a multiple robot system yields solutions that are not scalable and do not capture the asynchronous character of the robot cooperation. In view of the above, a promising approach is the hybrid control concepts that combines a DES-based (Discrete Event System) supervisory control logic with a CTS-based (Continuous Time System) robot motion control. While various aspects of this type of approach have been recently considered, e.g., [5–8, 10], few works provide formal methodologies that are adaptable to changes in problem settings and guarantee the correct and efficient operation of MMRS in the entire domain of their model definition.

In this chapter, we consider a group of mobile robots, whose operation will be viewed as *a set of cyclic robot motion processes* concurrently executed in a shared area. A practical example of such a system can be a Flexible Assembly System (FAS), in which all of the parts that are needed to make one assembly are kitted on one pallet and routed on vehicles through the work stations until complete. The components are palletized into kits and finished assemblies removed from the pallets in a kitting station (KS). An assembly vehicle is dedicated to a pallet from the moment when

it picks the pallet up in the KS to the moment when it returns to the KS. After the vehicle drops the finished assembly off, it picks up the next pallet containing another kitted assembly to be built, or if there are no new jobs that require service, remains in the zone adjacent to the KS. The zone is big enough to accommodate all the vehicles, and this is where they park when the system is shut down. Preparation of the kits in a warehouse can also be done by mobile robots visiting the relevant storing area, and abstracted by cyclic processes.

The objective of this work is to present the DES-based framework for the coordination of mobile robots sharing a common 2D motion space, proposed in [12, 15], show its application to the supervisory control of cyclic robot motion processes, and discuss its implementation in a centralized or a distributed controller. The following section describes the control problem for MMRS, gives the assumptions and requirements defining the sought solution. Section 3 explains the assumed discretization scheme of the continuous robot motion space and motion processes. Section 4 describes the RAS (Resource Allocation System) abstraction of concurrent processes [13] and their application to identifying specific classes of MMRS. A model of the feasible dynamic behavior of the robot system is then obtained by mapping the distinguished RAS into DFSA (Deterministic Finite State Automaton) that ensures collision avoidance among the robots. Based on this model, the subsequent section deals with the deadlock avoidance problem, discusses its complexity, and provides relevant algorithms to solve it. Finally, Sect. 6 concentrates on the control architecture implementing the developed control logic, and the last section concludes this research.

2 Problem Statement

We consider a Multiple Mobile Robot System (MMRS) viewed as a group of autonomous mobile robots sharing a 2D space. Each robot performs a mission that requires it to travel multiple times along a specific closed path. The path of each robot is planned independently, without taking into account any positional constraints introduced by the paths of other robots. The robots operate asynchronously and are able to control their motion with path-following algorithms that allow each of them to correctly perform its mission when alone on the stage. When sharing the motion space, the robots must refine their motion strategies in order to avoid collisions, through modification of their paths, velocity profiles or both.

The objective of the MMRS control is to ensure that the operation of the system is *correct* and *efficient*. The notion of correctness relates to a qualitative criterion and requires that each robot be able to perform its mission without colliding with other robots. That is, depending on the state of other robots, the path of a robot may be re-planned and/or the robot may have to slow down or even come to a stop and wait until the situation changes and it can safely resume further travel. Thus, the correct control must also ensure for each robot, the possibility to resume its travel after a break, i.e., eliminate from the MMRS behavior such phenomena as deadlocks

and robot starvation. The induced modifications of the robot trajectories inevitably cause an increase of their mission completion time, thus impact MMRS performance measures, whose values can vary depending on the employed conflict resolution policies. Consequently, there are two main questions driving the development of the MMRS control.

1. How to modify dynamically the initially assumed motion control of the robots so that:
   a. in a finite time interval, all the robots will have accomplished their missions,
   b. at each moment of this time interval, the areas occupied by any given pair of robots are disjoint.
2. How to induce, within the admissible (i.e. observing requirements (1.a) and (1.b)) robot concurrent operation, efficient MMRS behavior.

As can be noticed, the control satisfying requirements (1.b) and (1.a) ensures, respectively, collision-free and deadlock-free (free of both physical and logical deadlocks) concurrent motion of the robots. Requirement (2) implies the need of a flexible model of MMRS control that leaves room for the optimization of system efficiency and of tools to carry it out. To achieve realization of these postulates, we employ a modular control system, whose subsequent synthesis steps will be discussed in the sequel.

3 Discrete Representation of MMRS

We start from a continuous representation of MMRS. The system consists of a set of mobile robots $A = \{A_1, \dots A_n\}$ that share a finite planar workspace $WS \subset \mathscr{R}^2$ with the XY coordinate system. Each robot $A_i \in A$ is represented by a disk with radius a_i, and its path p_i is viewed as a curve in WS that is given by a pair of functions $p_i(l) = (x_i(l), y_i(l)) \in \mathscr{R}^2, \ \ l \in [0, \bar{l}_i] \subset \mathscr{R}$.

The tessellation of the robot motion space that leads to a discrete abstraction of their motion can take place in many different ways. Here, similar to [12], we assume a simple tesselation scheme provided by a grid of horizontal and vertical lines spaced at a distance $d \geq 2\rho$ and centered at the origin of a coordinate system, (x, y), that is superimposed on the motion plane. The resulting cells will be denoted by $W = \{w[i, j] : \ i \in \{-\underline{I}, \dots, -1, 0, 1, \dots, \overline{I}\}, \ \ j \in \{-\underline{J}, \dots, -1, 0, 1, \dots, \overline{J}\}\}$, where $-\underline{I}$, $\overline{I}$, $-\underline{J}$, and $\overline{J}$ are taken large enough to encompass the entire area $\mathscr{U}$, that supports the robot motion. Then, given a point $(x, y) \in \mathscr{U}$ and a cell $w[i, j]$, we define:

$$(x, y) \in w[i, j] \Longleftrightarrow (i - 1) \cdot d \leq x \leq i \cdot d \ \wedge \ (j - 1) \cdot d \leq y \leq j \cdot d$$

The size d of the grid, that defines the length of the cell edges, should be selected by considering the efficiency criteria mentioned above. In general, a smaller value of d

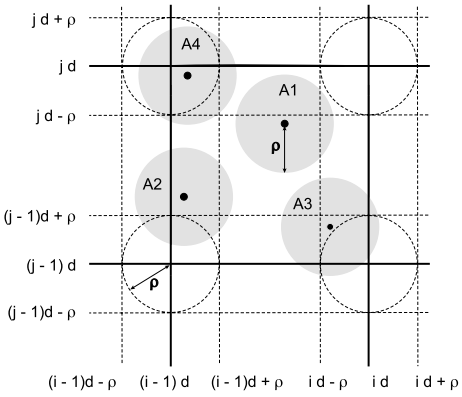

Fig. 1 Motion space partition (solid line) and regions of constant cell occupation (dashed line)

can accommodate a larger number of robots, and therefore, can lead to a higher space occupancy, but at the same time, it will lead to more disruption of the robot travels by the superimposed resource allocation process, and possibly to more congested traffic and longer delays.

In the sequel, we shall say that a robot (with its disk) centered at (x^c, y^c) *occupies* cell $w[i, j]$ *iff* its disk overlaps the cell, i.e., there exists $(x, y) \in w[i, j]$ with $||(x, y) - (x^c, y^c)|| \leq \rho$, where $|| \cdot ||$ denotes the Euclidean norm. A graphical illustration of this concept is given in Fig. 1. More specifically, the adopted tesselation is defined by the grid of the solid horizontal and vertical lines, and the mobile robots are depicted by the grey disks in it. As can be noticed, a robot can occupy one cell (as in the case of $A1$), two neighboring cells (as in the case of $A2$), three neighboring cells (as in the case of $A3$), or four neighboring cells (as in the case of $A4$).

Moreover, for the considered tesselation scheme, the subset of cells occupied by a mobile robot that is located at (x^c, y^c) is effectively determined by the relative positioning of (x^c, y^c) with respect to another partitioning of the motion plane, that is induced by the original tesselation scheme and the robot geometry. In Fig. 1, this induced partitioning is depicted by the dashed lines. If the disk center of a robot is located in one of the circles then the robot occupies all four adjacent cells (as in the case of $A4$). a robot occupies three cells if it is centered in any region that is the difference between any circle and the square that describes it (as in the case of $A3$). Next, a robot occupies two cells if its center lies in a rectangle located along the tesselation line (as in the case of $A2$). And finally, a robot occupies one cell if it is centered in the square located in the middle of the cell (as in the case of $A1$).

The above characterized tesselation, depicted in Fig. 1 by the dashed lines, partitions robot paths into maximal segments of *constant cell occupation*. That is, the subset of cells occupied by the robot centered at any of point of a given segment is the same, and it is different from the set of cells occupied by the robot in the sectors preceding and succeeding the considered one. Consequently, it is convenient to view the set of cells W as the set of MMRS resources, and abstract the motion process of a robot as a sequence of stages, each of which requires for its execution a specific sub-

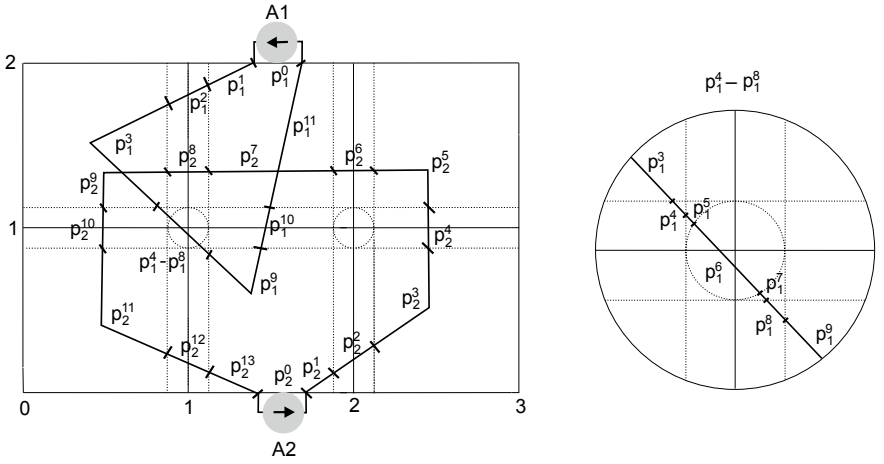

Fig. 2 Example paths of two mobile robots and the corresponding resource allocation profiles that are defined by the path partitioning into maximal segments with the same cell occupation. The right part of the figure details the profile obtained for robot $A1$

Table 1 The resource allocation induced by the path segmentation of Fig. 2

(a) Robot 1

Stage No.	Required resources
0	$\emptyset$
1	$w[1, 1]$
2	$w[0, 1]$, $w[1, 1]$
3	$w[0, 1]$
4	$w[0, 0]$, $w[0, 1]$
5	$w[0, 0]$, $w[0, 1]$, $w[1, 1]$
6	$w[0, 0]$, $w[0, 1]$, $w[1, 0]$, $w[1, 1]$
7	$w[0, 0]$, $w[1, 0]$, $w[1, 1]$
8	$w[0, 0]$, $w[1, 0]$
9	$w[1, 0]$
10	$w[1, 0]$, $w[1, 1]$
11	$w[1, 1]$

(b) Robot 2

Stage No.	Required resources
0	$\emptyset$
1	$w[1, 0]$
2	$w[1, 0]$, $w[2, 0]$
3	$w[2, 0]$
4	$w[2, 0]$, $w[2, 1]$
5	$w[2, 1]$
6	$w[1, 1]$, $w[2, 1]$
7	$w[1, 1]$
8	$w[0, 1]$, $w[1, 1]$
9	$w[0, 1]$
10	$w[0, 1]$, $w[0, 0]$
11	$w[0, 0]$
12	$w[0, 0]$, $w[1, 0]$
13	$w[1, 0]$

set of resources $W' \subset W$. An example partitioning of two paths is demonstrated in Fig. 2. Path p_1 of robot $A1$ consists of twelve (maximal) segments p_1^0–p_1^{11}, and path p_2 of robot $A2$ consists of fourteen such segments, p_2^0–p_2^{13}. Also, Table 1 specifies the cells occupied by the two robots at the various stages of their route.

4 Collision Avoidance in MMRS

The discrete representation of the motion space and robot paths, discussed in the previous section, makes it possible to view MMRS as a sub-class of Resource Allocation Systems (RAS) [13], called FREE-RANGE-RAS [12] and defined as follows.

Definition 5.1 A *FREE-RANGE-RAS* is defined as a 4-tuple $\Phi = (\mathscr{R}, C, \mathscr{P}, D)$ such that:

1. $\mathscr{R} = \{R_1, \ldots, R_m\} = W$ is the set of system resources, representing the set of cells.
2. $C : \mathscr{R} \to \mathbb{Z}^+$ is the resource capacity function that defines the maximal number of robots that can occupy each particular cell at a time.
3. $\mathscr{P} = \{P_1, \ldots, P_n\}$ is the set of processes, representing the motion of each particular robot A_i along its path. Each process P_i is characterized by an ordered set of stages $\Xi_i = \{\Xi_{i1}, \ldots, \Xi_{i,l(i)}\}$ corresponding to the motion of robot A_i (observed through its disk center) along the consecutive segments of its path.
4. $D : \bigcup_{i=1}^{n} \Xi_i \to 2^R$ is the resource requirement function that defines the resources $D(\Xi_{ij}) = D_{ij}$ required by every process P_i to execute its each particular stage $\Xi_{i,j}$.
5. The sets of stages Ξ_i, $i = 1, \ldots, n$, and function D arise from a geometrical system. That is, there exists a set of planar paths p, which can be divided into segments p_{ij}, $j = 1, \ldots, l(i)$, traversing the cells so that they induce function D.

Moreover, we will distinguish two sub-classes of FREE-RANGE-RAS, namely FREE-RANGE-k-RAS, where $k \in \mathbb{Z}^+$, and FREE-RANGE*-RAS.

Definition 5.2 A FREE-RANGE-k-RAS is a FREE-RANGE-RAS in which the capacity of each cell $R \in \mathscr{R}$ is $C(R) = k$.

Definition 5.3 A FREE-RANGE*-RAS is a FREE-RANGE-RAS in which for all $i = 1, \ldots, n$, $j = 1, \ldots, l(i)$, $\mid D_{ij} \mid \in \{1, 2\}$. That is, no robot ever occupies more than two cells at a time, which is equivalent to that no robot's path overlaps any corner square of the tesselation grid.

A 4-tuple $\Phi = (\mathscr{R}, C, \mathscr{P}, D)$, specifies the parameters of a particular MMRS and gives its static abstraction. A dynamic model of MMRS can be developed using the formalism of the Deterministic Finite State Automaton (DFSA) [2], defined as follows.

Definition 5.4 A deterministic finite state automaton (DFSA) is a 6-tuple $G = (S, E, \Gamma, f, s_0, S_M)$, where:

- S is the (finite) set of *states*.
- E is the (finite) set of *events*.
- $\Gamma : S \to 2^E$ is the *active-event function*. Event $e \in E$ can occur in state $s \in S$ iff $e \in \Gamma(s)$.

- $f : S \times E \to S$ is the (partial) *transition function*, defined for pairs (s, e) such that $e \in \Gamma(s)$. $s' = f(s, e)$ returns the state that results from the occurrence of event e in state s.
- $s_0 \in S$ is the *initial state* of G
- $S_M \subseteq S$ is the set of *marked states*

The above DFSA starts its operation from state s_0. In each state $s \in S$, an event e can only occur if the state transition function $f()$ is defined for the pair (s, e), i.e., if $e \in \Gamma(s)$. In that case, we say that event e is *enabled* in state s. The occurrence of event e in s results in a new state $s' = f(s, e)$, which can be changed subsequently by the occurrence of event e' that is enabled in state s', and so on. In order to capture state transitions arising from strings of events, the state transition function f can be inductively extended to $S \times E^*$ by the following assumptions:

$$\begin{aligned} &\forall s \in S && \big(f(s, \varepsilon) \equiv s\big) \\ &\forall s \in S\ \forall u \in E^*\ \forall e \in E && \big(f(s, ue) \equiv f(f(s, u), e)\big) \end{aligned}$$

In the above equations, ε denotes the empty string, and E^* denotes the set of all strings that can be constructed with the elements of the set $E \cup \{\varepsilon\}$. Moreover, it is implicitly assumed that the involved single-step transitions correspond to the enabled events, i.e., to the state-event pairs for which the original function f is defined; otherwise, the extended version of f is undefined on the corresponding state-string pair. Furthermore, we say that state $s \in S$ is *reachable* from state s_0 if there exists string $u \in E^*$ such that function $f(s, u)$ is defined; the set of all states reachable from s is called the *reachability set* of s and denoted by $Re(s)$. A special case of such sets, $Re(s_0)$, is called the reachability set of the DFSA G. Using the discussed formalism, a dynamic model of a particular MMRS can be obtained by the following mapping of its specification Φ into DFSA.

Definition 5.5 The DFSA $G(\Phi) = (S, E, \Gamma, f, s_0, S_M)$ abstracting the feasible dynamics of a FREE-RANGE-RAS $\Phi = (\mathscr{R}, C, \mathscr{P}, D)$ is defined as follows:

1. The *state set* S consists of all vectors $s = (s_1, s_2, \ldots, s_n) \in \mathbb{Z}^n$ such that:

$$\begin{aligned} &a. && \forall i \in \{1, \ldots, n\} && \big(0 \le s_i \le l(i)\big), \\ &b. && \forall R \in \mathscr{R} && \big(a(s, R) = |\{s_i : R \in D_{i,s_i}\}| \le C(R)\big). \end{aligned}$$

   Each component s_i of s indicates the current stage of process P_i (the motion process of robot A_i). In particular, $s_i \neq 0$ indicates that robot A_i is in the s_i-th path segment of its route, and $s_i = 0$ indicates that robot A_i is located off the shared space (thus holding no resource $R \in \mathscr{R}$), where it ends one cycle of its travel and starts another. For each $R \in \mathscr{R}$, $a(s, R)$ indicates the number of units of resource R that are allocated in state s.
2. The *event set* $E = \{e_i : i = 1, \ldots, n\}$, where for every $i = 1, \ldots, n$, e_i represents the transition of robot A_i to its next stage.

3. For each pair (s, e_i), the *state transition function* returns the new state $s' = f(s, e_i)$, whose components s'_k, $k = 1, \ldots, n$, are given by:

$$s'_k = \begin{cases} (s_k + 1) \mod (l(k) + 1) & \text{if } k = i \\ s_k & \text{otherwise} \end{cases}$$

4. Function $f()$ is defined for a pair (s, e) *iff* $e \in \Gamma(s)$, where the *set of feasible events* Γ is defined by $\Gamma(s) \equiv \{e \in E \mid s' = f(s, e) \in S\}$.
5. The *initial state* $s_0 = \mathbf{0}$, which corresponds to the situation where all the robots are in their private space and therefore, all the system resources are free.
6. The *set of marked states* S_M is the singleton $S_M = \{s_0\}$, and it expresses the requirement for complete process runs.

If $\Phi \in$ FREE-RANGE-1-RAS, that is, if the capacity of all cells is $C(R_i) = 1$ then the above defined DFSA model enforces mutually exclusive occupation of the cells by the robots and, consequently, their collision-free motion. Otherwise, that is, if for some cell R_i, $C(R_i) > 1$ then still no collisions can occur among robots that have been allocated disjoint sets of cells, but an additional local coordination system is needed to prevent internal collisions of the robots within each cell R_i. Such a system can be based on a DES model, obtained by further discretization of the cell, creating a local FREE-RANGE-1-RAS, or employ some of the reactive collision avoidance methods, e.g., based on the potential field [9].

5 Deadlock Avoidance in MMRS

The operation of the $MMRS$ model $G(\Phi)$, ensuring the disjoint motion of the robots' disks, satisfies Requirement (1.b) defined in Sect. 2. However, Requirement (1.a) is not satisfied, as the reachability set of $G(\Phi)$ can contain states s that are not *safe*, i.e., such that the initial state s_0 is not reachable from s.

To observe this, consider again the example of two robots and their path segmentation, depicted in Fig. 2, and the resource requirements induced by this path segmentation, given in Table 1. Let us assume that $\Phi \in$ FREE-RANGE-1-RAS, that is, the capacity of the cells is 1. Next notice that in the initial state event sequence $u = e_1, e_2, e_2, e_1, e_2, e_2, e_1, e_2$ is feasible and drives the system from state $s_0 = [0, 0]$ to state $s_1 = f(s_0, u] = [3, 5]$. Then two state transitions are feasible: to state $s_2 = f(s_1, e_1) = [4, 5]$ and to state $s_3 = f(s_1, e_2) = [3, 6]$. State $s_2 = [4, 5]$ is safe, as the initial state s_0 can be reached from it by completing first the cycle of robot A_1 and then completing the cycle of robot A_2. In state $s_3 = [3, 6]$ only two event sequences are feasible: $u' = e_1, e_2$ and $u'' = e_2, e_1$, and both drive the system to state $s_4 = f(s_3, u') = f(s_3, u'') = [4, 7]$, which is a deadlock as $\Gamma(s_4) = \emptyset$. No event is feasible in s_4 because for its next stage, robot A_1 requires resource $w[1, 1]$, which is held by robot A_2, and robot A_2 requires resource $w[0, 1]$, which is held by robot A_1.

In order to enforce the correct operation of $G(\Phi)$, it is necessary to introduce a *supervisor* that extends the *feasible-event function* $\Gamma(s)$ to a more restrictive *admissible-event function*. The supervisor disables the occurrence of some state transitions and thus constrains the behavior of MMRS so that for each admissible event sequence, there exists its admissible extension driving the system to the initial state. This makes the system *reversible*, hence deadlock-free, and allows each process to repeat its cycle any arbitrary number of times.

The optimal, i.e., the least restrictive supervisor should accept an event $e \in E$ in state $s \in S$ if and only if $e \in \Gamma(s)$ (event e is enabled in state s) and the next state $s' = f(s, e)$ is safe. Thus, any algorithm to check these conditions must solve the following problem.

Safety problem: Given a FREE-RANGE-RAS $\Phi = (\mathscr{R}, C, \mathscr{P}, D)$, a safe state $s \in R(s_0)$ and an enabled event $e \in \Gamma(s)$, find out whether or not state $s' = f(s, e)$ is safe.

As demonstrated in [11], the Safety problem is NP-complete even if addressed to any of the sub-classes of RAS, FREE-RANGE-k-RAS, $k \in \mathbb{Z}^+$. On the other hand, there exists a polynomial algorithm solving the safety problem for systems $\Phi \in$ FREE-RANGE*-RAS, in which the capacity of each resource $R \in \mathscr{R}$ is $C(R) > 1$ [15]. The high complexity of the first group of problems implies that practically only sub-optimal solutions of the safety problem can be considered in FREE-RANGE-RAS. Such algorithms ensure the reachability of the initial state, but not necessarily in the least restrictive way.

From the viewpoint of the control synthesis for MMRS, most useful appear two of the sub-classes of FREE-RANGE-RAS, each of which has its pros and cons. The first class, FREE-RANGE-1-RAS, assumes that no more than one robot at a time can be present in each particular cell, thus no further local coordination is required. However, the supervisory control employs a sub-optimal algorithm, which is in general overly restrictive and may have some negative impact on the efficiency of the system. The second class, FREE-RANGE*-2-RAS, allows for a maximally permissive supervisor, so no unnecessary event disabling ever happens. Yet, it imposes some constraints on the shape of the paths, which should omit the corner squares of the tesselation structure. Also, as more than one robot can be present in a cell at a time, an additional control is needed to ensure then their collision-free motion. Such a coordination is, however, fairly simple, as the maximal number of robots in a cell is limited to two. In the following we present two supervisors for the two distinguished models, as proposed in [12, 15], respectively.

5.1 Deadlock Avoidance in FREE-RANGE-1-RAS

In these systems, the reversibility of $G'(\Phi)$ can be enforced by constraining the reachability space $Re(s_0)$ of $G(\Phi)$ to the subspace of *p-ordered* states $Re'(s_0) \subseteq Re(s_0)$. The definition of such states, given below, uses the notion of a *private stage*

of process P_i, Ξ_{iq}, which means that the resources required at this stage, $D_{i,q}$, are not required by any other process in order to complete their cycle and reach back their initial state.

Definition 5.6 In system $G(\Phi)$, state $s = (s_1, \ldots, s_n)$ is *p-ordered* iff there exists an order on the set of robots $\mathscr{A}$, $p : \mathscr{A} \to \{1, 2, \ldots, n\}$ that satisfies the following condition: $\forall i, j$ s.t. $p(A_j) > p(A_i)$, $\forall k = s_i..q_i$, $D_{ik} \cap D_{js_j} = \emptyset$, where q_i is the smallest number s.t. $q_i \geq s_i$ and stage Ξ_{iq_i} is private, if such a number exists. Otherwise $q_i = 0$.

Less formally, a state s is p-ordered iff there exists an order of the robots such that no robot with higher order occupies any of the cells that lie on the way of a robot with a lower order to its nearest private stage. A procedure checking whether or not this property is observed by a particular state of a particular system $G(\Phi)$ is presented in Algorithm 1.

Algorithm 1. The function testing the p-ordered property of states in FREE-RANGE-1-RAS

Input : Parameter Φ describing the RAS, state $s \in S$.
Output: *True* if the state is p-ordered, *false* otherwise.

```
1  Function p-ordered(Φ, s) : bool
2      A ⟵ P;
3      occupied ⟵ ∅;
4      for i = 1, ..., n do
5          occupied ⟵ occupied ∪ D_{i s_i};
6          q_i ⟵ minpriv(i, s_i); remain[A_i] ⟵ ⋃_{j=s_i}^{q_i} D_{i,j};
7      repeat
8          A' ⟵ A;
9          for A_i ∈ A do
10             if remain[A_i] ∩ occupied = D_{i s_i} then
11                 A ⟵ A \ {A_i};
12                 occupied ⟵ occupied \ D_{i,s_i};
13         if A = ∅ then
14             p-ordered ⟵ TRUE;
15         else
16             p-ordered ⟵ FALSE;
17     until A' = A ∨ A = ∅;
```

As in the above algorithm, the operations on the set $\mathscr{A}$ are $O(n)$ complex, the complexity of the whole function is $O(n^2)$, which qualifies it for online applications. It is also not hard to notice the following property.

Property 5.1 *In system $G(\Phi)$, the final state can be reached from any state s that is p-ordered.*

Proof The condition defining the p-ordered state provides the robots A_i, $i = 1..n$, with the ability to progress one-by-one, in the order given by $p(A_i)$, to their respective closest private stages $\Xi_i^{q_i}$. Since then no robot occupies a cell that can be required by any other robot on its way to complete the cycle, the robots can one by one reach back their initial state. ■

Definition 5.7 A p-controlled MMRS $G'(\phi)$ is a restriction of DFSA $G(\Phi) = (S, E, \Gamma, f, s_0, S_M)$ obtained by:

- substituting the *feasible-event function* $\Gamma(s)$ by *admissible-event function* $\Gamma'(s) = \{e : e \in \Gamma(s) \ \wedge \ s' = f(s, e)$ is p-ordered$\}$, and
- substituting the transition function f with f' such that $f'(s, e) = f(s, e)$, but it is only defined for admissible pairs (s, e), i.e., such that $e \in \Gamma'(s)$.

The following theorem proves that $G'(\Phi)$ is reversible and thus its operation satisfies Requirement (1.a).

Theorem 5.1 *In a p-controlled MMRS $G(\Phi)$, the initial state s_0 is reachable from each state s reachable from the initial state s_0.*

Proof Based on Definition 5.7, each state reachable in a p-controlled $G(\Phi)$ is p-ordered. Thus, by Property 5.1, the theorem holds. ■

5.2 Deadlock Avoidance in FREE-RANGE*-2-RAS

Since in this class of models the robot paths omit the square corners of the tesselation grid, the motion process of each robot consists of a sequence of stages that correspond alternately to the travel in a cell and the transition from one cell to the next one. At a stage of the latter type, a robot occupies both cells, yet it eventually passes to a stage of the former type and its disk no more occupies the previous cell. Thus, from the viewpoint of deadlock avoidance, the transitions between the cells can be considered as transient states, and FREE-RANGE*-2-RAS can be viewed as a system of processes, whose each stage Ξ_{ij} requires a single resource $D(Xi_{ij}) = D_{ij} \in \mathscr{R}$.

The supervisor defined for this class of MMRS control models employs a graphical representation of a state that has the form of *resource allocation graph*.

Definition 5.8 The resource allocation graph representing a state $s \in S$ of $G(\Phi)$ is a graph $F(s) = (V, H)$ such that:

- The set of vertices is defined by the extended set of resources $V = \mathscr{R} \cup \{R_\infty\}$, where R_∞ is a dummy resource of infinite capacity allocated to each process $P_i \in \mathscr{P}$ at its stage Ξ_{i0}.

Algorithm 2. The function testing the safety of state transition $s' = f(s, e_i)$ in FREE-RANGE*-2-RAS.

Input: Parameter Φ describing the RAS, state s, and the index i of the process, whose potential advancement to the next stage is considered in the context of the safety of the resulting state s'.

Function $safe(\Phi, s, i) : bool$
  if $s_i = l(i)$ **then**
    return *true*
  if $a(s, D_{is_{i+1}}) = |\{s_k : D_{k,s_k} = D_{is_{i+1}}\}| = 0$ **then**
    return *true*
  $s' \longleftarrow f(s, e_i)$;
  if $\exists t = R^1, R^2, \ldots, R^q$, $q \geq 1$, *such that* $R^1 = D_{is_{i+1}}$ *and* $R^q = R_\infty$ *or* $a(s', R^q) = |\{s'_k : D_{k,s'_k} = R^q\}| < 2$ **then**
    return *true*
  return *false*

- The set of edges is defined by the set of robot processes $H = \mathscr{P}$. The edge (corresponding to process) P_i goes from vertex $R \in \mathscr{R}$ to vertex $R' \in \mathscr{R}$ *iff*, at state s, process P_i has been allocated resource R and for its next stage it requires resource R'. Edge P_i goes from vertex $R \in \mathscr{R}$ to vertex R_∞ iff $s_i = l(i)$, i.e., at state s, process P_i executes the last stage of its cycle, $\Xi_{i,l(i)}$. Edge P_i goes from vertex R_∞ to a vertex $R \in \mathscr{R}$ iff $s_i = 0$, that is, process P_i is in its initial state.

The following theorem [14] provides a property that allows the construction of a maximally permissive supervisor.

Theorem 5.2 *Consider a RAS $\Phi \in$ FREE-RANGE*-2-RAS, a DFSA $G(\Phi)$, a reachable safe state s, and an event e_i such that the next state $s' = f(s, e_i)$ is defined. Then, state s' is safe iff in the graph $F(s')$, there exists a path $t = R^1, R^2, \ldots, R^q$, $q \geq 1$, from resource $R^1 = D_{is_{i+1}}$, to a resource $R^q \in V$ that in state s' is allocated to fewer processes than its capacity.*

It is clear that the restriction of any DFSA $G(\Phi)$, $\Phi \in$ FREE-RANGE*-2-RAS, to $G'(\Phi)$ obtained by substituting function $\Gamma(s)$ with function $\Gamma'(s) = \{e : e \in \Gamma(s) \wedge s' = f(s, e)$ is safe$\}$ yields a model of MMRS that is reversible and maximally permissive, i.e., it captures all the trajectories of $G(\Phi)$ that reach the initial state and no state $s \in Re(s_0)$ from which the initial state is not reachable.

When verifying the safety condition, there is no need to construct graph $F(s')$, from scratch at each state change $s' = f(s, e)$, as it can be directly obtained by a small update of $F(s)$—removing one edge and adding another. Moreover, it is possible to distinguish two special cases of state s when the safety condition holds: (i) $s_i = l(i)$, as then $D_{is_{i+1}} = R_\infty$ and its capacity is infinite, and (ii) resource $R^1 = D_{is_{i+1}}$ is not allocated to any process in state s. Thus, checking the safety of a state transition can be done with the function specified in Algorithm 2. The most complex part of the

calculations is testing the existence of the required path in graph $F(s')$, which can be done with, e.g., the depth first search, that has the $O(|V| + |H|)$ computational complexity.

6 Implementation of the MMRS Control Logic

The logic described in the previous sections can be implemented both in a centralized and distributed manner. In the former case, all the robots communicate with the central RAS controller. In the latter case, each robot is equipped with a local RAS-controller, a higher control level that interacts with its lower control level in the same way as in the centralized case, but, additionally, it has to communicate with other robots in order to be aware of their state. The specifics of the distributed implementation of the FREE-RANGE-RAS based control can be found in [15], whereas here we focus on the common features of both approaches.

6.1 Interaction of Robots with Their RAS-Controller

The interaction of the robots and the RAS-controller is event-based, as depicted in Fig. 3. The controller generates only one type of events, $next_i$, $i = 1, \ldots, n$, which is a permission for robot A_i to proceed to the next path segment. Having received this message, the robot confirms it with the ack_i signal. Next, each robot A_i informs the controller about the occurrence of three types of events, ap_i, cp_i, and rp_i, corresponding to reaching three types of characteristic points on its path: ap, cp, and rp.

- Event ap_i is generated when robot A_i passes an *approach point* ap, which signals that A_i is approaching its next stage. These points are distinguished at the end of the path segments p_{ij} such that the transition to the next path segment $p_{i,j+1}$ requires allocation of additional resources $D_{i,j+1} \setminus D_{ij} \neq \emptyset$.
- A *critical point* cp is set at a safe-braking distance from the end of each path segment p_{ij} in which appears point ap. If by arriving at point cp, robot A_i has not received the signal $next_i$, it generates event cp_i and starts decelerating. At the

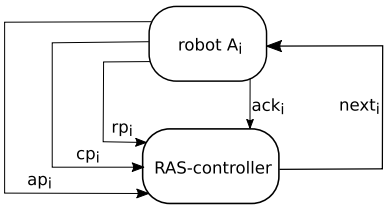

Fig. 3 Interaction of the RAS-controller with a robot

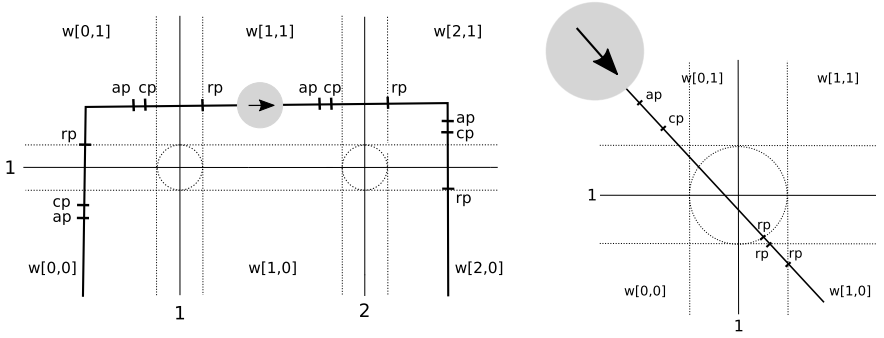

Fig. 4 Two exemplary paths with their characteristic points

arrival of signal $next_i$, which can occur after the robot has come to the standstill or is still decelerating, it accelerates again and proceeds to the next section.

- A *release point* rp is set at the border of two path segments p_{ij} and $p_{i,j+1}$ such that $D_{i,j+1} \subset D_{ij}$. On passing a point rp, robot A_i generates event rp_i to inform the controller that it can deallocate from the robot the resources $D_{ij} \setminus D_{i,j+1}$.

An example of a robot path with its characteristic points is given in Fig. 4. If a path segment p_{ij} is very short, in particular if it is equal or shorter than the safe-braking distance, then such a segment can be merged with the next segment $p_{i,j+1}$ or the previous stage $p_{i,j-1}$, creating a process stage that requires the union of the resources required by both stages, $D_{ij} \cup D_{i,j+1}$ or $D_{i,j-1} \cup D_{ij}$, respectively. In the right part of Fig. 4, three path segments preceding the robot were merged: the segment between the dashed horizontal and vertical line, the segment between the vertical line and the 'square', and the segment between the 'square' and the 'circle'. The resulting path segment starts in cell $[w[0, 1]$ at the cross of the path with the dashed horizontal line, ends in cell $w[0, 1]$ at the cross of the path with the 'circle', and its resource requirement is $\{w[0, 0], w[0, 1], w[0, 1], w[1, 1]\}$. All these cells should be allocated before the robot crosses the horizontal line, so one pair of points ap and cp is sufficient. There is no need, however, to merge the symmetric path segments in cell $w[0, 1]$, between the 'circle' and the 'square', and between the horizontal and vertical lines, as the resources are only released here. Since no new resource allocation is needed, there is no risk that the robot will have to brake, thus there is no lower limit on the path segment length.

6.2 Priority Control

The decisions made by the RAS controller concern the selection of events that will be activated in a particular state through signals $next_i$. To make such decisions, the controller must follow the state of the robots, updating it at the occurrence of

each event that is sent/received to/from any robot. Then it checks the admissibility conditions established in Sects. 4, 5, and solves the conflicts among the robots using some *priority policy* in order to decide which robots should be allowed to continue their motion, and which should be temporarily suspended in the case when concurrent operation of all of them is not admissible. By a priority policy we understand an algorithm that, based on some heuristic priority criterion, selects a subset of events $\Delta(s) \subseteq \Gamma'(s)$ such that:

(i) Each pair of events $e_i, e_k \in \Delta(s)$ is non-conflict, that is, occurrence of one of them does not make inadmissible the other. Hence, robots A_i and A_k can transit to their next sectors concurrently.
(ii) For each event $e_i \notin \Delta(s)$, there exists event $e_k \in \Delta(s)$ that is in conflict with e_i, i.e., granting robot A_k the permission to transit to its next sector makes the transition of robot A_i inadmissible.

The set $\Delta(s)$ is recalculated at each state change, and command *next* is sent to all robots A_i that have passed their approaching point ap and $e_i \in \Delta(s)$. The algorithms to calculate $\Delta(s)$ can range from simple priority rules to optimization algorithms based on predicted time parameters of the robot motion, yet, since executed online, they must feature low computational complexity. Since the proposed RAS controller can serve both a real MMRS and its simulator, the selection of the most efficient policy, with respect to an assumed criterion, can be done experimentally.

7 Remarks and Conclusions

Although it is possible to view the operation of a group of mobile robots as a set of processes sharing common resources, there is a number of features that seriously differ this system from other systems typically represented by a similar abstraction, e.g., job shops. The motion processes of the robots and their shared resource—the motion space are not inherently discrete, but have a continuous character, and when viewed as a resource sharing system require discretization. The system is deadlock-prone, and the problem of deciding whether or not a particular resource allocation is safe in a particular state is NP-complete. The robots operate asynchronously and their travel time between distinguished points can be only roughly estimated. The actual time can substantially vary from the initially assumed one. This is due to inaccurate time calculation based, e.g., on the robot control model as well as possible on-line modification of its path or velocity profile necessary for collision avoidance with other robots. Moreover, vehicle transport systems, are often subordinate to other systems, e.g., a machining system in a manufacturing plant. That is, the start moments of particular transport operations depend on the activity of the latter system, which makes the prediction of the time behavior of the former system still more complex.

All the above features imply the need of a DES-based robot coordination system that calculates control decisions online, based on the current state of the robots, rather

than execution of an offline calculated schedule. The presented work advocates such an approach.

In this chapter we recaptured our earlier results concerning the supervisory control synthesis for free-ranging mobile robot systems within the framework of the RAS model, showed its application to the supervisory control of cyclic robot motion processes, and discussed implementation of this concept in a RAS-controller. Currently we are focused on the development of a MMRS controller employing the presented theory for a fleet of real mobile robots, which will be followed by experimental research.

Acknowledgements This work was partially supported by grant no. 2016/23/B/ST7/01441 of the National Science Center.

References

1. Barraquand, J., Latombe, J.: Robot motion planning: a distributed representation approach. Int. J. Robot. Res. **10**(6), 628–649 (1991)
2. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Kluwer Academic Publishers, Boston (1999)
3. Chang, C., Chung, J., Lee, B.: Collision avoidance of two robot manipulators by minimum delay time. IEEE Trans. Syst., Man, Cybern. **24**(3), 517–522 (1994)
4. Chang, D., Shadden, S., Marsden, J., Olfati Saber, R.: Collision avoidance for multiple agent systems. In: Proceedings of 42nd IEEE International Conference on Decision and Control, Maui, HI, USA, 9–12 December 2003, pp. 539–543. IEEE (2003)
5. Cirillo, M., Pecora, F., Andreasson, H., Uras, T., Koenig, S.: Integrated motion planning and coordination for industrial vehicles. In: ICAPS'14 Proceedings of the Twenty-Fourth International Conference on International Conference on Automated Planning and Scheduling, Portsmouth, New Hampshire, USA, 21–26 June 2014, pp. 463–471. AAAI Press (2014)
6. Claes, D., Tuyls, K.: Multi robot collision avoidance in a shared workspace. Auton. Robot. **42**(8), 1749–1770 (2018)
7. Draganjac, I., Miklic, D., Kovacic, Z., Vasiljevic, G., Bogdan, S.: Decentralized control of multi-AGV systems in autonomous warehousing applications. IEEE Trans. Autom. Sci. Eng. **13**(4), 1433–1447 (2016)
8. Ferrera, E., Capitán, J., Castaño, A.R., Marrón, P.J.: Decentralized safe conflict resolution for multiple robots in dense scenarios. Robot. Auton. Syst. **91**, 179–193 (2017)
9. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. J. Robot. Res. **5**(1), 90–98 (1995)
10. Kousi, N., Koukas, S., Michalos, G., Makris, S.: Scheduling of smart intra-factory material supply operations using mobile robots. Int. J. Prod. Res. **57**(3), 801–814 (2018)
11. Reveliotis, S., Roszkowska, E.: On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems. IEEE Trans. Autom. Control. **55**(7), 1646–1651 (2010)
12. Reveliotis, S., Roszkowska, E.: Conflict resolution in free-ranging multivehicle systems: a resource allocation paradigm. IEEE Trans. Robot. **27**(2), 283–296 (2011)
13. Reveliotis, S.A.: Real-time Management of Resource Allocation Systems: A Discrete Event Systems Approach. Springer, New York (2005)
14. Roszkowska, E., Goral, I.: Correct-by-construction distributed control for multi-vehicle transport systems. In: IEEE International Conference on Automation Science and Engineering, Madison, Wisconsin, USA, 17–21 August 2013, pp. 156–161. IEEE (2013)

15. Roszkowska, E., Reveliotis, S.: A distributed protocol for motion coordination in free-range vehicular systems. Automatica **49**(6), 1639–1653 (2013)
16. Snape, J., van den Berg, J., Guy, S.J., Manocha, D.: The hybrid reciprocal velocity obstacle. IEEE Trans. Robot. **27**(4), 696–706 (2011)
17. Valle, S.M.L., Hutchinson, S.A.: Optimal motion planning for multiple robots having independent goals. IEEE Trans. Robot. Autom. **14**, 912–925 (1998)