Conflict Resolution in AGV Systems

Spyros A. Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology

Abstract

Currently, conflict-free routing in AGV systems is established by means of one of the following three approaches: (i) the problem elimination through the adoption of a segmented path flow or tandem queue configuration, (ii) the identification of imminent collisions through forward sensing and their aversion through vehicle backtracking and/or rerouting, or (iii) the imposition of zone control and extensive route pre-planning, typically based on deterministic timing of the vehicle traveling and docking stages. Among these three approaches, the segmented path flow-based approach presents the highest robustness to the system stochasticities/randomness, but at the cost of restricted vehicle routings and the need for complicated handling operations. This paper proposes an alternative conflict resolution strategy that will ensure robust AGV conflict resolution, while maintaining the operational flexibility provided by free vehicle travel on arbitrarily structured guidepath networks. Specifically, the approach advocated in this paper also employs zone control, but it determines vehicle routes incrementally, one zone at a time. Routing decisions are the result of a sequence of safety and performance considerations, with the former being primarily based on structural/logical rather than timing aspects of the system behavior. The resulting control problem is characterized as the AGV structural control. After defining the notion of AGV structural control, the paper proceeds to the formal characterization and analysis of the problem, and to the development of a structural control policy appropriate for the class of AGV resource allocation systems. The paper concludes with some discussion on the accommodation of emerging AGV operational features in the proposed modeling and analysis framework, and the integration of AGV structural control with the broader control of material-flow among the shop-floor workstations.

1 Introduction

Automated Guided Vehicle (AGV) systems are considered as one of the most appropriate modes for material handling support of contemporary flexibly automated production environments. In general, such a system consists of a set of cooperating driver-less vehicles which transport goods and materials among the different workstations and storage sites of a production facility, usually by following a set of predetermined, physical or virtual guidepaths embedded in the facility layout, and coordinated by a centralized or distributed computer-based control system. Some of the primary advantages attributed to these environments are increased routing flexibility, space utilization and safety, and reduced overall operational cost [13].

The research on the design and operation of AGV systems was initiated with the publication of the seminal paper of Maxwell and Muckstadt in 1982 [25], and it has been an ongoing activity, of increasing intensity, till the present time. To a large extent, this long-standing openness of the AGV design and control problem is due to the dynamic evolution of (i) the underlying technological capabilities, and (ii) the operational requirements posed by the emerging manufacturing paradigms. A good account of the current state of art in AGV hardware can be found in [33], while an excellent survey of all existing results on the AGV design and control problem, as defined by various system configurations, is provided in [13].

According to [13], the key issues to be addressed for the successful deployment of an AGV system are: (i) the flowpath design, (ii) the determination of the fleet size, (iii) the development of operational policies for job scheduling, and AGV scheduling and dispatching, and (iv) the establishment of conflict-free routing. The first two of these four issues are typically addressed during the system (re-)design level, while the last two are issues that concern and drive the system real-time operation. This paper provides an analytical treatment and offers a novel perspective to the last of the aforestated problems, i.e., establishing conflict-free vehicle routing.

In the past, the problem of establishing conflict-free routes in AGV systems has been addressed in a number of different ways. Among them, the simplest and most robust addresses the problem at the design level, by dividing the operational space into a number of non-overlapping domains, each served by a single vehicle. A number of transfer buffers, interfacing a number of strategically preselected domain pairs, establishes the interconnectivity of the entire system. Depending on the detailed lay-out of each domain, these systems are known either as *segmented flow path* [35, 36] or *tandem* [6, 7] AGV configurations. In AGV systems with more general routing schemes, (physical)

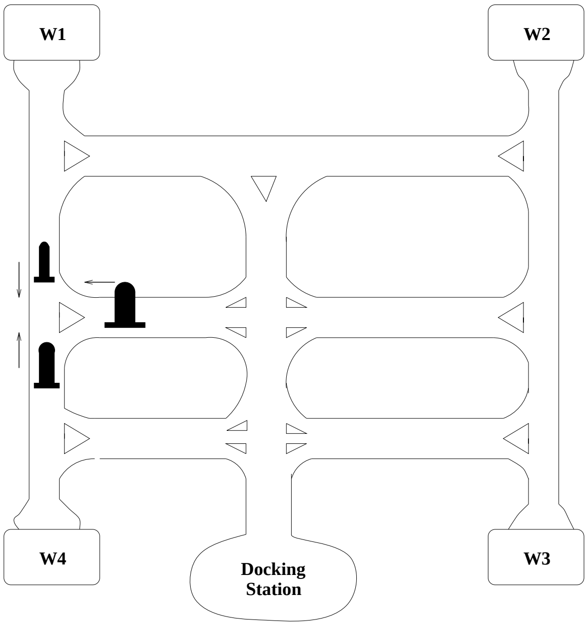


Figure 1: An example of an AGV system deadlock

vehicle collisions are avoided by (i) *zone control*, i.e., the logical segmentation of the system guidepath network into a number of *zones*, and the control of the vehicle traffic so that each zone is occupied by only one vehicle at a time, or by (ii) *forward sensing*, i.e., the deployment of proximity sensors on each vehicle, that will alert about the close presence of another vehicle and will force it to stop. Notice, however, that even though physical collision is avoided, both approaches can give rise to situations similar to that depicted in Figure 1: In this case, none of the three vehicles can advance, and unless corrective action is taken, all three vehicles will be permanently blocked; this situation is characterized as an AGV system *deadlock*. In current AGV control practices, deadlocks are either (i) detected and resolved by appropriate backtracking and/or re-routing of vehicles to space buffers, shaped as *loops, sidings* or *spurs* [11], or (ii) they are predicted and avoided, by extensive route pre-planning. These route plans, in turn, (i) can be the results of analytical formulations borrowed from the literature of vehicle routing with time constraints (e.g., [18, 21]), (ii) might be developed through some search-oriented techniques (e.g., [9, 19]), or (iii) they can be obtained and validated through (PN-based) simulation (e.g., [41, 24]). Furthermore, we note that there have been attempts to address the AGV control problem and the establishment of conflict-free routes as part of the broader problem of *FMS scheduling*; such a monolithic *Mixed Integer Programming (MIP)* formulation is presented in [5].

Except from their very high – usually non-polynomial – complexity, all the route planning schemes mentioned above are very sensitive to the system randomness/stochasticities, and they are non-robust to any contingencies arising during the system operation [37]. This increased sensitivity and non-robustness results from the fact that all these planning strategies (i) are essentially *open-loop* control policies, and (ii) they are based on *deterministic* timing of the vehicle traveling and docking stages. Given the significantly stochastic nature of the operations of contemporary manufacturing systems, the eventual applicability of these open-loop control plans is highly questionable. On the other hand, approaches based on deadlock detection and recovery are more robust to the stochasticity of the system operation, but they will suffer the disruption of vehicle backtracking and re-routing. In general, AGV systems in which route conflicts and deadlocks are resolved by detection and recovery, will be less predictable in their behavior, and therefore, less measurable and controllable w.r.t. performance. Furthermore, these systems will have to deal with the extra cost of buffer space requirements. Finally, tandem configurations are the most robust to the AGV collision and deadlock problems, but they establish their robustness on restricted vehicle routes[1] and the need for complicated handling operations (since a unit load transfer might need the cooperation of two or more vehicles).

The work presented in this paper seeks to ensure robust AGV collision and deadlock avoidance of the level established by tandem configurations, while maintaining the *operational* flexibility provided by free vehicle travel on arbitrarily structured guidepath networks. Specifically, the system layouts considered in this paper are similar to that depicted in Figure 1. Following current manufacturing trends and practices [13, 21], we consider motion on the different links of the guidepath network to be *bidirectional*. However, AGV's themselves are assumed to be *unidirectional*, i.e., they must do a U-turn in order to travel in the opposite direction.[2] To avoid physical collision, *zone control* is applied; a natural segmentation associates one zone with each intersection and workstation node, and each path link. In case, however, that there are lengthy path links, they can be further segmented into a series of zones. Finally, the system possesses some *docking* station(s) where idle vehicles park and potentially recharge their batteries. The capacity of the docking station(s) is sufficient to accommodate all the vehicles, e.g., when the system is shut down.

Vehicle traveling in the considered AGV system is conceptualized as a "mission" starting from

[1]As it was pointed out by one of the paper reviewers, the significance of this effect is currently mitigated by the relatively easy reconfiguration of the underlying guidepath network topology, through the employment of free ranging vehicle technology.

[2]Except from expressing the current capabilities of AGV technology, this constraint also disables *backtracking*, and therefore, it enforces a sense of intention in vehicle motion.

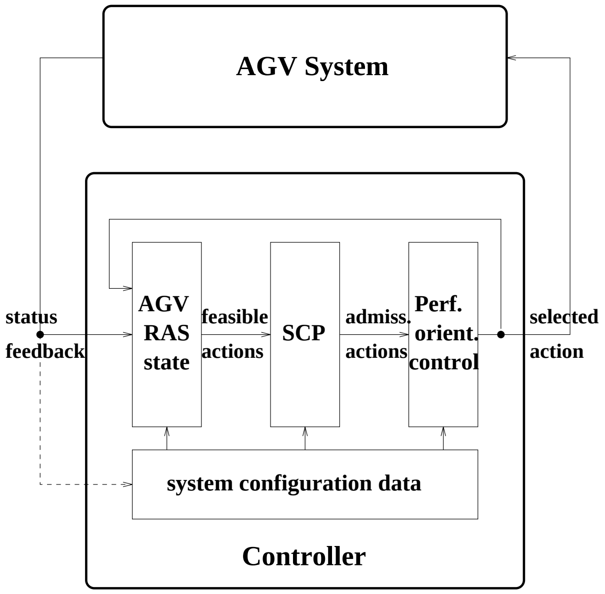


Figure 2: The proposed AGV control architecture

the docking station, and visiting successively a *"source"* and a *"destination"* workstation, in order to conclude again to the docking station. It is possible, however, – in fact, quite probable – that a vehicle having completed its trip up to the "destination" point, will be re-assigned and redirected to another trip, before reaching the docking station. The conceptual requirement that an *idle* vehicle should (be able to) eventually return to the docking station is introduced mainly for analysis purposes, that are explained in the next section.

Contrary to the route planning schemes discussed above, the control framework proposed in this paper determines *dynamically* the detailed path that a vehicle will follow while carrying out its "mission" trip, one leg (*zone*) at a time.[3] Specifically, a vehicle having finished the traversal of its currently allocated zone, is either (i) allocated a new zone or (ii) commanded to wait in its current zone. This decision is based on *safety* and *performance* considerations, and it is the result of the decision-making process depicted in Figure 2: First, the set of *feasible* control actions – i.e., the potential allocations of the vehicle to a new free zone neighboring its currently allocated zone – is determined based on (i) real-time feedback regarding the AGV traffic status, and (ii) the applied zone control scheme. To ensure, however, ultimate collision and deadlock-free operation,

[3]Such a dynamic vehicle routing scheme has been suggested previously in [37], and it was characterized as *incremental*. Our approach extends the notion of incremental vehicle routing by further discriminating among *structural* and *performance-oriented* control.

feasible actions must be filtered through a *structural control policy (SCP)*, which determines the set of *admissible* actions (vehicle zone allocations) [29, 31]. Finally, a *performance-oriented control / scheduling* policy selects a specific admissible action to be executed next – i.e., the assignment of the vehicle to a particular new free zone among the set of admissible zones, or the commandment of the vehicle to wait in its current zone till the occurrence of a new event.

Notice that the proposed control scheme is an instantiation of *closed-loop* or *state-feedback* control logic, since it considers explicitly the current system status in its reasoning, and therefore, it is more robust and reactive to any randomness and/or contingency arising during the system operation. Furthermore, by synthesizing the vehicle routes incrementally, in real-time, the proposed approach can take full advantage of the routing flexibility provided by the topology of the guidepath network. Finally, it is important to notice that the proposed approach decomposes the routing conflict resolution problem from the system performance considerations. Specifically, collision and deadlock-free operation is established on appropriate logical constraining of the system operational space through the application of a structural control policy (SCP), whereas performance-oriented control seeks to optimize the system behavior w.r.t. time-based criteria like throughput, waiting times and resource utilizations, over the operational space admitted by the applied SCP. We believe that this decoupling of the conflict resolution scheme from the detailed timing of the system operations is essential for the final robustness of the proposed control logic.

Stated in the context of the AGV control framework depicted in Figure 2, the main objective and contribution of this paper is the development of effective and efficient SCP's for the AGV systems modeled by Figure 1.[4] To this end, we model the AGV system as a (sequential) *resource allocation system (RAS)* – to be called the AGV RAS – and formally characterize its behavior and deadlock-related properties. Resource allocation systems and the underlying problem of deadlock avoidance arising from conflicting resource requests, were initially studied in the context of computer operating systems, back in the late '60's and early '70's; the work presented in, e.g., [15, 8, 16], is indicative of that effort. Subsequently, the problem complexity was formally characterized in late '70's [2, 14], while in the 80's, deadlock resolution was addressed in the context of distributed databases and computation, e.g., [34]. Since early '90's there is a resurgent interest in the problem of deadlock in the manufacturing system context: the effective management of flexibility inherent in modern automated production systems requires automated resolution of the deadlock problem,

[4] The complementary problem of *AGV performance-oriented control* for incremental AGV route planning is an open research problem, currently under investigation, and it will constitute the topic of a sequel paper.

while the operational characteristics of modern production systems and efficiency considerations render impractical the immediate transfer of past results. Hence, results expressing the current state of art in the manufacturing deadlock-related research can be found in, e.g., [3, 38, 39, 40, 12, 31]. Finally, we notice that a modeling and analysis perspective similar to that proposed in this paper, was recently advocated for the broader class of traffic control systems (e.g., railway system control) in [27]. However, contrary to the work presented in [27], this work proceeds to the complete development of a centralized conflict resolution controller, and develops structural control policies which are *provably correct* – i.e., (under the considered modeling assumptions) will *always* prevent collisions and the occurrence of deadlock – and *scalable* – i.e., of *polynomial* complexity.

The detailed organization of the rest of the paper is as follows: Section 2 models the considered AGV system as a resource allocation system, and investigates its deadlock-related properties. This analysis follows and extends some recently developed theory for deadlock avoidance in sequential RAS, appropriate for flexibly automated production systems [31]. Section 3 develops a variation of *Banker's* algorithm [15] which is appropriate for the AGV RAS. Our past experience with the manufacturing system deadlock has shown Banker's algorithm to be a very efficient deadlock avoidance policy for sequential RAS [23]. Section 4 considers a series of variations of the AGV RAS model, and discusses how the results of Section 3 can be modified and extended to cover these cases. It also considers the integration of AGV structural control to the broader framework controlling the entire shop-floor material flow. Finally, Section 5 concludes the paper discussion, and proposes potential extensions of this work.

2 A formal characterization of the AGV RAS model and its deadlock properties

The operation of the AGV system under consideration (cf. Figure 1 and the corresponding discussion in Section 1) is formally abstracted to a *resource allocation system (RAS)* as follows: The system topological structure is modeled by the AGV *guidepath graph*, $G = (N, E)$; in the following, it is also assumed that G is *undirected* and *connected*. The arc set E of this graph is defined by the AGV zones corresponding to the path links of the network, and it constitutes the *resource set* of the underlying RAS model. The node set N corresponds to the intersections, workstations and the docking station of the actual AGV network, as well as the interconnecting points of artificially segmented links. In particular, the system workstations are modeled by a set of terminal nodes N_W

connected to the rest of the network by a single arc. This arc corresponds to the operational zone for vehicles loading and unloading parts to/from the station, but it also models the fact, that in many contemporary AGV system configurations, a vehicle visiting a certain station, can leave it in any of the two possible directions. This effect is modeled by assuming that an AGV entering the link leading to some workstation node $n_i \in N_W$, immediately changes its direction on that link, heading back towards the rest of the network. Similarly, the docking station is modeled by another terminal node, d_s, connected to the rest of the network with a single arc. However, a vehicle entering this node, vanishes. This effect results from the ability of the docking station to accommodate the entire AGV fleet, which renders it equivalent to an *infinite* capacity job *initiation/termination* node, in the AGV RAS operation. Except from these two singularities characterizing the operation of the vehicles at the system workstations and the docking station, vehicle motion on the rest of the guidepath graph is *unidirectional*. The guidepath graph of the AGV layout depicted in Figure 1 is given in Figure 3.

Processes in this RAS model correspond to "mission" trips executed by vehicles $v_i \in V$. They are formally defined by triplets: $< n_s,\ n_d,\ d_s >$, where $n_s, n_d \in N_W$ define the *"source"* and the *"destination"* nodes, respectively. The trips are initiated either at node d_s, the docking station, or by re-assigning and re-directing an *idle* vehicle performing the last part of its trip, i.e., returning from node n_d to node d_s. Furthermore, at any point in time, the number of processes (vehicles) in the system is limited to $m \equiv |V|$, i.e., the number of vehicles available in the system.

For structural analysis purposes, vehicles are assumed to always reside in one of the guidepath arcs $e \in E$. Specifically, even though in the physical operation of the system, a vehicle can find itself in one of the system intersections, it is assumed that this situation is only *transient*, corresponding to transitions from link to link of the system guidepath; vehicles cannot stay in an intersection while waiting for the acquisition of the next link. Hence, intersections are only facilitators of the motion from link to link, and they do not constitute resources of the underlying RAS model. The only control needed for the allocation of the intersection zones, is the localized prioritization of the vehicles waiting to cross them, so that collisions in the intersections are avoided.

A formal characterization of the AGV RAS *state* is as follows:

Definition 1 *The* state s *of the AGV RAS model is determined by the status of each of the m vehicles circulating in the system. Specifically, any vehicle $v_i \in V$ can be either* `PARKED` *at the docking station d_s, or* `TRAVELING`*, executing a "mission" trip. In the latter case, the vehicle status is further specified by: (i) an* ordered list L_{v_i} *containing the "milestone" nodes remaining to be visited, and (ii) the guidepath edge, $e_{v_i} \equiv (k, l)$, currently allocated to the vehicle. Moreover, edge (k, l)*

should be interpreted as a directed *edge, with the direction defining the current vehicle motion on that edge.*

From Definition 1, it follows that the system state space, S, is *finite*. Furthermore, *transitions* between these states correspond to the following events:

1. A parked vehicle is assigned to a new "mission" and allocated its first zone.
2. A vehicle waiting on edge (i, j) is allocated a neighboring edge (j, k). In case that edge (j, k) is a link leading to a workstation terminal node, it is automatically assumed that the vehicle has veered its direction to (k, j). If $k \equiv d_s$, the vehicle status is switched to `PARKED`.
3. An idle vehicle returning to the docking station d_s, is re-assigned and redirected to a new "mission" trip.

The RAS state space armed with the state transition function described above, defines a *Finite State Automaton (FSA)* [17, 29]. The *initial* and *final* states of this automaton correspond to state s_0, in which the system is at rest, i.e., all vehicles are parked at d_s. Hence, the *language* accepted by this automaton corresponds to sets of completed transport tasks.

However, we have already seen (cf. Fig. 1) that under uncontrolled operation, the completion of some tasks might not be possible due to the occurrence of deadlocks. A formal definition of the AGV deadlock is as follows:

Definition 2 *An AGV* deadlock *is a RAS state such that there exists a* cut *of the guidepath graph for which: (i) one of the two defined subgraphs is allocated to capacity, and (ii) all the cut edges are allocated to AGV's moving towards the fully allocated subgraph.*

Notice that if one of the subgraphs defined by the cut considered in Definition 2 is a single node, condition (i) is trivially satisfied. For instance, the cut corresponding to the AGV system deadlock depicted in Figure 1 consists of edges (A, H), (I, H) and (G, H), with the sub-graph allocated to capacity being the single node H (cf. Figure 4).

We have also seen that the AGV control paradigm proposed in this work employs the notion of a *structural control policy (SCP)*, that guards against the occurrence of AGV deadlocks. Specifically, the role of the applied SCP is to screen the tentative state transitions resulting from (i) zone allocations to vehicles having completed the traversal of their currently allocated zone, or (ii) (re-)

assignment of free vehicles to new mission trips, and to disable these transitions that lead to deadlocking situations. It can be shown that deadlock states in the AGV RAS are polynomially detectable [30, 32]. However, effective deadlock avoidance requires the detection and disabling of transitions leading not only to deadlock states, but also to a broader class of states from which deadlock is unavoidable. This class of states is characterized as *unsafe* states, and its existence for the AGV RAS model is demonstrated in Figure 5.

A thorough analysis of unsafety in the considered class of RAS and its complexity is presented in [2, 14], where it is shown that, in the general case, detection of unsafety for a given RAS state is an NP-complete problem.[5] From a practical perspective, the NP-completeness of the state safety problem implies that, for the general class of the considered AGV systems, the implementation of the *optimal* deadlock avoidance policy – i.e., the *minimally restrictive* policy which prevents a state transition if and only if (*iff*) it results in an unsafe state – is computationally intractable. This computational intractability of the optimal SCP, combined with the real-time nature of the sought control policies, further imply that for all practical purposes, we have to compromise optimality, and thus, operational flexibility, for computational tractability. A methodology for developing correct and computationally tractable (*scalable*) SCP's has been proposed in [22]. The underlying idea is to identify a property P such that: (i) $P \Rightarrow$ `SAFETY`, (ii) the computational cost of testing P on a given RAS state is *polynomial* with respect to the system size, and (iii) the subgraph of the RAS state transition diagram defined by the subspace of states presenting property P, $S(P)$, is *strongly connected, containing the initial state* s_0. Then, it is clear that confining the system operation in the subspace $S(P)$ through one step-lookahead is computationally tractable, and the resulting operation of the system is deadlock-free. Furthermore, condition (iii) ensures that property P does not *induce* any additional deadlocks. A final requirement arising from efficiency considerations, is that property P should be selected in such a way that the system operation is not over-constrained with respect to the maximal capabilities/capacity determined by the optimal SCP.

Past experience with sequential RAS modeling the operation of flexibly automated production systems, indicates that a quite efficient, correct and scalable SCP for them can be obtained from the implementation of the idea underlying the “classical” *Banker's algorithm*, originally developed for deadlock avoidance in computer operating systems [23]. This finding is further corroborated by other researchers in the field, e.g., [20, 27]. However, the highly flexible routing scheme supported by

[5] To apply this complexity analysis in the AGV RAS context, the *problem size* is defined based on (i) the size of the AGV guidepath graph, and (ii) the number of vehicles available in the system.

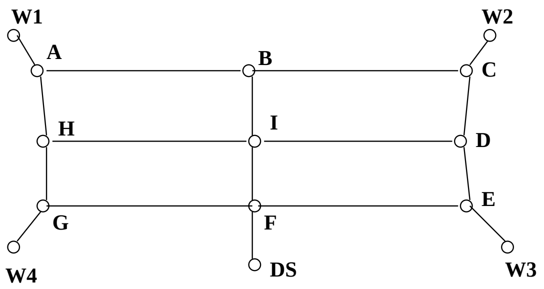


Figure 3: The guidepath graph of the AGV system in Figure 1

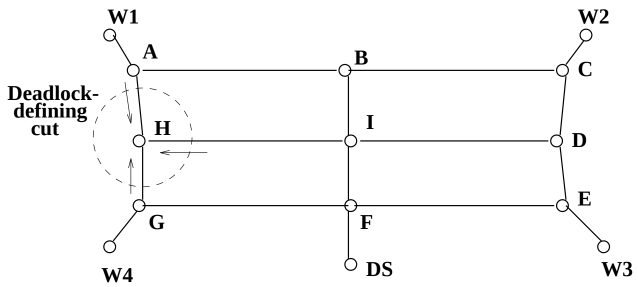


Figure 4: The cut of the guidepath graph corresponding to the deadlock of Figure 1

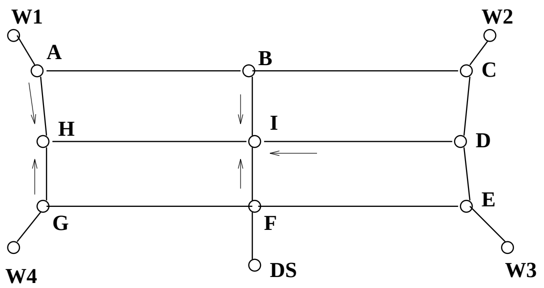


Figure 5: An AGV system *unsafe* state: The arrows along the edges indicate the existence of a vehicle on the edge, moving along the indicated direction. Hence, for the depicted situation, the only feasible movement for these five vehicles is to edge (HI). However, no matter which of the five vehicles is allocated this edge, the system will end up in deadlock.

the considered AGV RAS model, renders past implementations of the Banker's algorithm computationally intractable. Hence, in the next section, a new variation of Banker's algorithm is developed, which is able to handle efficiently the intricacies posed by the AGV RAS model, and provides a correct and scalable SCP for this environment.

3 The AGV Banker's Algorithm

3.1 The generic logic of Banker's Algorithm

Banker's algorithm is a generic scheme for developing *correct* and *scalable* SCP's, which applies to a broad class of RAS characterized by resource *re-usability*. Re-usability implies that the utilization of the system resources by the various processes does not degrade or perish them in any way; hence, the number of units from each resource type circulating in the system, is a system *invariant*. For the class of re-usable RAS, Banker's algorithm establishes a correct and scalable SCP by restricting the system operation to a particular subset of its states, characterized as *ordered*. The detailed definition of the *ordered state* concept is as follows:

Definition 3 *A (re-usable) RAS state s with running processes $q_i \in s(Q)$ is* ordered*, if there exists a bijective mapping (ordering) $o() : Q \rightarrow \{1, \ldots, |Q|\}$, such that each process $q_i \in Q$ can run to completion by using its currently allocated and free resources, as well as resources allocated to any process q_j with $o(q_j) < o(q_i)$.*

In other words, a state s is ordered iff there exists at least one feasible sequence of transitions from it to the final state s_0, which can be perceived as a *concatenation* of transition subsequences corresponding to *complete* executions of single processes. Let S_o denote the subset of ordered states, for a given RAS. The following statements hold true:

Proposition 1 *[23] The subspace induced by S_o constitutes a strongly connected component of the system state space, containing the initial state s_0. As a result, constraining the system operation in the class of ordered states, through one-step lookahead, constitutes a* correct *SCP.*

Proposition 2 *Testing $s \in S_o$, i.e., whether state s is ordered, can be performed efficiently by the following* greedy *procedure:*

Algorithm 1: the generic logic of Banker's Algorithm

1. `Initialize:`

(a) $U = P(s)$, *where* $P(s)$ *is the set of process instances running in* s;

(b) $R = F(s)$, *where* $F(s)$ *the set of free resources in* s;

(c) $i = 0$;

(d) $ORDERED = \texttt{TRUE}$.

2. `While`$(ORDERED \wedge (U \neq \emptyset))$ `do`:

(a) $i = i + 1$;

(b) *Try to find a process* $p \in U$ *such that it can terminate by using its currently allocated resources and the resources in set* R;

(c) `If` *no such process can be found,* `then` $ORDERED = \texttt{FALSE}$;

(d) `else`

i. $o(p) = i$;

ii. $U = U \backslash \{p\}$;

iii. $R = R \cup \texttt{res}(p)$;

3. `return` $ORDERED$;

Proof: Clearly, if the algorithm returns $ORDERED = \texttt{TRUE}$, state s is ordered, with a valid ordering being defined by the values of function $o()$, computed in step (2di), above. To show that the algorithm does not mis-classify any ordered state as unordered, we must show that the greedy scheme applied by the algorithm logic in the search for a valid process ordering is correct, i.e., no *backtracking* is necessary in that search. The truth of this last statement results immediately from the truth of the following implication:

$$ORDERED(s) \Longrightarrow ORDERED(s' : \ P(s') \equiv P(s) \backslash \{p\}) \tag{1}$$

In Equation 1, s' denotes the state that results from state s by deleting process instance p, and returning the resources allocated to it, to the pool of free resources. To see the validity of this equation, simply notice that a valid ordering for $P(s')$ can be obtained by "projecting" any ordering of $P(s)$ on $P(s') \equiv P(s) \backslash \{p\}$, i.e.,

$$o'(q : q \in P(s')) \equiv \begin{cases} o(q), & \text{if } o(q) < o(p) \\ o(q) - 1, & \text{if } o(q) > o(p) \end{cases} \tag{2}$$

□

A closer look at the generic Banker's logic reveals that the details of the algorithm implementation on a specific RAS type, and the entailed complexity, concern mainly Step (2b) in the algorithm description. In the next section we develop an efficient implementation of that step for the AGV RAS model.

3.2 Implementing Step (2b) of the generic Banker's Algorithm in the AGV RAS

In the following discussion, let $V(s)$ denote the set of "TRAVELING" vehicles in state s (cf. Def. 1). Then, interpreted in the context of the AGV RAS, Step (2b) of the generic Banker's algorithm requires the identification of a "TRAVELING" vehicle $v \in V(s)$ which can complete its "mission" trip, i.e., reach all the "milestone" nodes remaining in the associated trip-defining list L_v, by using only (i) free edges in the guidepath graph and (ii) its allocated edge $e_v \equiv (k,l)$. Given the (assumed) *unidirectional* motion of the system AGV's, this requirement can be further detailed as follows:

Proposition 3 *A vehicle $v \in V(s)$, currently situated on edge $e_v \equiv (k,l)$ of the guidepath graph, and with remaining "mission" trip defined by the ordered list $L_v \equiv < n_1, (n_2 \vee \texttt{null}), (n_3 \vee \texttt{null}) >$, can complete its "mission" trip according to Step (2b) of the generic Banker's algorithm, iff:*

Algorithm 2: Testing whether a vehicle v can complete its "mission" trip

1. *there exists a path of currently free edges connecting node l to n_1, and*

2. *there exist paths of currently free edges connecting nodes l or k to nodes n_2 and n_3, assuming that $L_v(3)$ and/or $L_v(2) \neq \texttt{null}$; o.w., this part is trivially satisfied.*

Proof: The truth of this statement results from the following three observations on the formal description of the AGV RAS:

1. For any "TRAVELING" vehicle, list L_v will contain at least one remaining "milestone" node.

2. According to the state definition of the AGV RAS, a vehicle $v \in V(s)$ associated to guidepath edge $e_v \equiv (k,l)$, has traversed this link from k to l, and it is currently waiting next to node l, for the allocation of its next link. Furthermore, given the vehicle unidirectional motion, the next "milestone" node, n_1, must be reached without using edge e_v, itself.

3. However, once the vehicle has reached its milestone node n_1, the previously held edge e_v becomes another free edge available for the subsequent travel of the vehicle to the remaining

"milestone" nodes. Given that node n_1 has already been shown to be interconnected to node l through a path of free edges, it suffices to establish the existence of free-edge paths from the terminal nodes of edge e_v to the remaining "milestone" nodes, in order to conclude that the vehicle can complete its trip.

□

The connectivity of node $i \in \{k,\ l\}$ to node $n_j,\ j = 1, 2, 3$, can be efficiently checked through a *"reaching"* scheme, which starts from node i and tries to identify a path to node n_j, by successively processing "already reached" nodes, and trying to "reach out" from them to additional nodes, through unused, free edges. The details of such a algorithm can be found in ([1], Sect. 3.4).

3.3 The AGV Banker's algorithm

Algorithm 2, supported by the "reaching" scheme discussed at the end of the previous section, provides an implementation of Banker's algorithm appropriate for the AGV RAS. Notice, however, that under this implementational scheme, the node inter-connectivity test employing the suggested "reaching" algorithm must be carried out for each vehicle $v \in V(s)$, and each pair of nodes implied by Proposition 3. Here, we show that it is possible to organize the computations of the AGV Banker's algorithm in such a way, that additional computational economies are obtained. The increased efficiency of this new computational scheme stems from the recognition that as vehicles are driven to completion and their currently allocated edges are freed for further use by the remaining vehicles, the connectivity of the underlying guidepath graph is improved. The additional computational gains result from exploiting this monotonicity.

Specifically, in the proposed AGV Banker's algorithm, we maintain the node inter-connectivity information in a separate data structure, which is updated every time another guidepath graph edge is freed. The key observation for defining this data structure is that, due to the *bi-directional* motion of the AGVs on the system path links, inter-connectivity through free edges defines an *equivalence* relationship, $\mathcal{E}$, on the node set, N, of the guidepath graph, G. The next proposition states more formally this result, and details an algorithm for the computation of the node equivalence classes induced by $\mathcal{E}$.

Proposition 4 *Let $F(s)$ denote the set of* free *guidepath graph edges, for a given AGV RAS state s. Then,*

I. *the relationship $\mathcal{E}$, defined on the guidepath graph node set N, by:*

$$\mathcal{E}: \; N \times N \; \equiv \{(i,j): \; i, j \textit{ are inter-connected by at least one path of } \text{free} \textit{ edges}\} \qquad (3)$$

is an equivalence *relationship.*

II. *The equivalence classes of $\mathcal{E}$, C_i, can be computed by the following algorithm:*

Algorithm 3: Computing the node inter-connectivity equivalence relationship

(1) *Initially, each node $n \in N$ is assumed to define a separate equivalence class, i.e., $\mathcal{E} = \{C_n \equiv \{n\}, \; \forall n \in N\}$.*

(2) *For every free edge $e \equiv (k,l) \in F(s)$, if $C_k \neq C_l$, the equivalence classes of nodes k and l are* merged*, i.e., $\mathcal{E} = (\mathcal{E} \backslash \{C_k, C_l\}) \cup \{C_k \cup C_l\}$.*

Proof: Clearly, $\mathcal{E}$ is reflexive and transitive, while the assumed possibility for bi-directional motion on a guidepath graph edge, makes it also symmetric. This establishes Part (I) of the proposition.

The correctness of Algorithm 3 can be formally established by an induction on the cardinality of set $F(s)$. Specifically, if $|F(s)| = 0$, then, graph G is totally disconnected with respect to paths consisting of free edges. Indeed, in this case, only Step 1 of Algorithm 3 will be executed, establishing one equivalence class for each guidepath graph node. Furthermore, consider that the algorithm works correctly for $F(s)$ with $|F(s)| = q$. Then, for $F'(s) = F(s) \cup \{e \equiv (k,l)\}$, we first process edges in $F(s)$, leaving edge e to be processed last. Clearly, for this edge, there are two options:

1. The terminal nodes of edge e, k and l, belong to the same equivalence class, $C_k = C_l \equiv C_{kl}$, in the relationship defined by the processing of edges in $F(s)$. This implies that nodes k, l are already interconnected by free edges in $F(s)$, and therefore, the addition of edge e in the set of free edges does not alter the inter-connectivity of graph G.

2. The terminal nodes of edge e, k and l, belong to different equivalence classes, $C_k \neq C_l$, in the equivalence relationship defined by $F(s)$. Then, the addition of edge e in the set of free edges introduces a unique new path between nodes k and l, bringing them in the same equivalence class C_{kl}. Obviously, all nodes previously interconnected to any of these nodes are now members of C_{kl}. Hence, $C_k \cup C_l \subseteq C_{kl}$. Furthermore, $C_{kl} \subseteq C_k \cup C_l$, since any node in C_{kl} other than nodes k and l must be connected to them through edges in $F(s)$. As a result, $C_{kl} = C_k \cup C_l$, and the correctness of the *merging* operation of Step 2 of Algorithm 3 is established.

□

The merging operation defined in Step 2 of Algorithm 3 is also all that is required in order to update the node inter-connectivity status every time a vehicle completes its trip and frees its currently allocated edge. The next two theorems show that the availability of the guidepath graph node inter-connectivity relationship, $\mathcal{E}$, supports a specialized, efficient implementation of Banker's algorithm in the AGV RAS context.

Theorem 1 *The following algorithm is a* correct *implementation of Banker's logic in the AGV RAS context:*

Algorithm 4: AGV Banker's Algorithm

(Given state s, return `TRUE` *if $s \in S_o$, i.e., it is an* ordered *state;* `FALSE` *o.w.)*

1. `Initialize:`
   - (a) $U = V(s)$;
   - (b) *use Algorithm 3 to compute the initial inter-connectivity relationship, $\mathcal{E}$, for the guidepath graph node-set N;*
   - (c) $i = 0$;
   - (d) $ORDERED =$ `TRUE`.
2. `While`$(ORDERED \wedge (U \neq \emptyset))$ `do:`
   - (a) $i = i + 1$;
   - (b) *Try to find a vehicle $v \in U$ able to complete its "mission" trip, by using the logic of Algorithm 2, and data structure $\mathcal{E}$;*
   - (c) `If` *no such vehicle can be found,* `then` $ORDERED =$ `FALSE`;
   - (d) `else`
     - i. $o(v) = i$;
     - ii. $U = U \backslash \{v\}$;
     - iii. *Update data structure $\mathcal{E}$, by using Step 2 of Algorithm 3;*
3. `return` $ORDERED$;

The correctness of Algorithm 4 results immediately from the truth of Propositions 1 - 4. Next we show that the algorithm is also efficient.

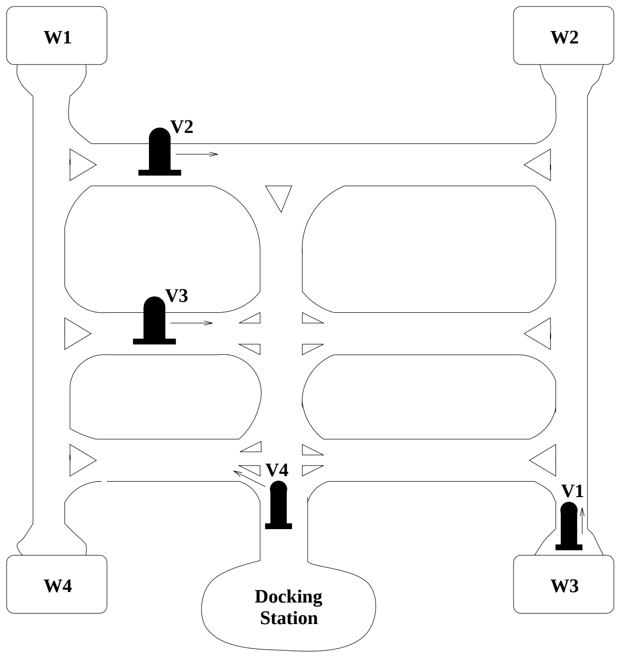
Theorem 2 *The complexity of the AGV Banker's algorithm is $O(|N| + |E| + |V|^2)$, where $|N|$, $|E|$ denote the cardinalities of the node and edge sets, N, E, of the guidepath graph, G, and $|V|$ denotes the cardinality of the vehicle set, V.*

Proof: The two main operations performed by Algorithm 4 are (i) the (incremental) computation of the node inter-connectivity relationship, $\mathcal{E}$, and (ii) the testing of whether an unordered vehicle can complete its "mission" trip, given the current inter-connectivity status. Regarding the computation of $\mathcal{E}$, its initialization step is of complexity $O(|N|)$ (cf. Algorithm 3), while the merging step (Step 2 of Algorithm 3) is performed once for every free (or freed) edge, and therefore, it is of complexity $O(|E|)$. The second major operation, i.e., testing whether a given unordered vehicle can complete its "mission" trip with the current inter-connectivity status, has a complexity $O(1)$, thanks to the availability of the data structure $\mathcal{E}$. However, these tests are performed within the loop of Step 2 in Algorithm 4, and it should be easy to see that in the worst case, $|V| + (|V| - 1) + \ldots + 1$ of them might have to be carried out. Therefore, the cumulative computational cost incurred by this second operation is $O(|V|^2)$. $\square$

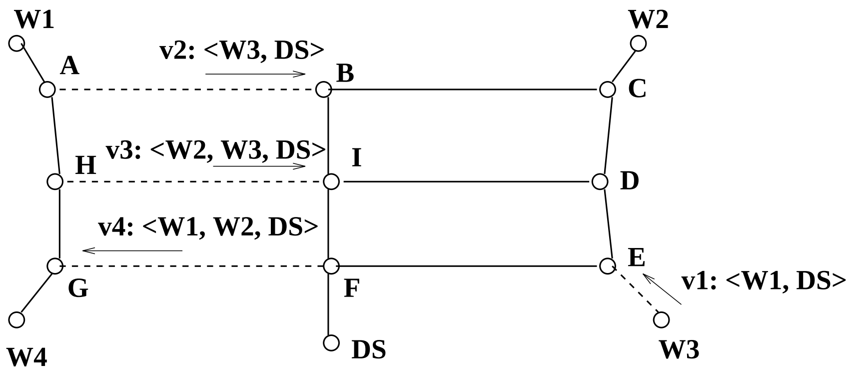
3.4 Examples

We conclude this section by presenting two examples which demonstrate the role of the applied SCP w.r.t. the conflict resolution in AGV systems, the notion of ordered AGV RAS states, and the detailed application of the AGV Banker's logic.

First, consider the situation depicted in Figure (6,a). In this case, vehicle v_1 has just unloaded a part at workstation W_3 and it is heading towards the docking station, vehicle v_2 is crossing the indicated zone of the guidepath graph transferring a part to workstation W_3, vehicle v_3 is heading towards workstation W_2 in order to pick a part that is to be transferred to workstation W_3, while vehicle v_4 is idling at the docking station. At the considered point, a new transfer request is posed to the AGV system controller, concerning the transfer of a part from workstation W_1 to workstation W_2. The controller considers allocating vehicle v_4 to this transfer, in which case, the vehicle should depart on the corresponding trip by traversing the guidepath zone between the docking station, and the workstation W_4. Before, however, the initiation of this motion is actually commanded/communicated to the vehicle controller, the system controller must evaluate its safety w.r.t. the arising of any potential deadlocks. To this end, the AGV system controller must determine whether the AGV RAS state that would result from the allocation of the aforestated zone to vehicle v_4 is *ordered*, i.e., whether it is accepted by the Banker's algorithm detailed in Theorem 1.



(a)



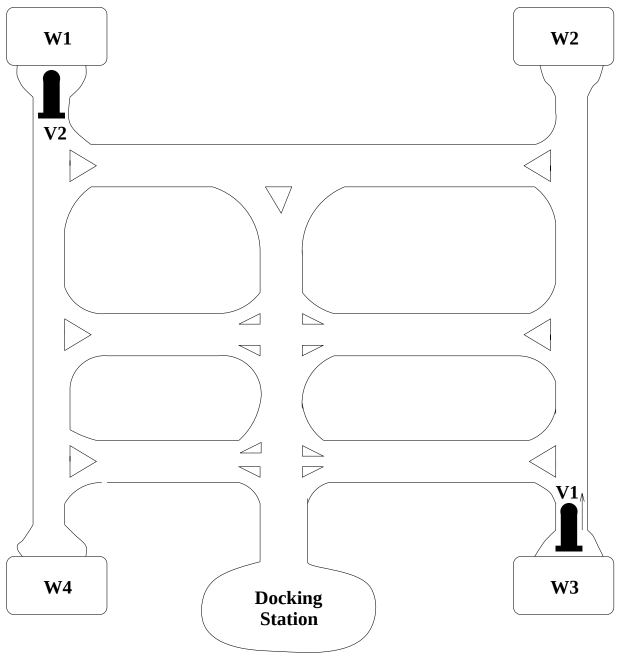
(b)

Figure 6: Example: Running the AGV Banker's algorithm on an ordered state

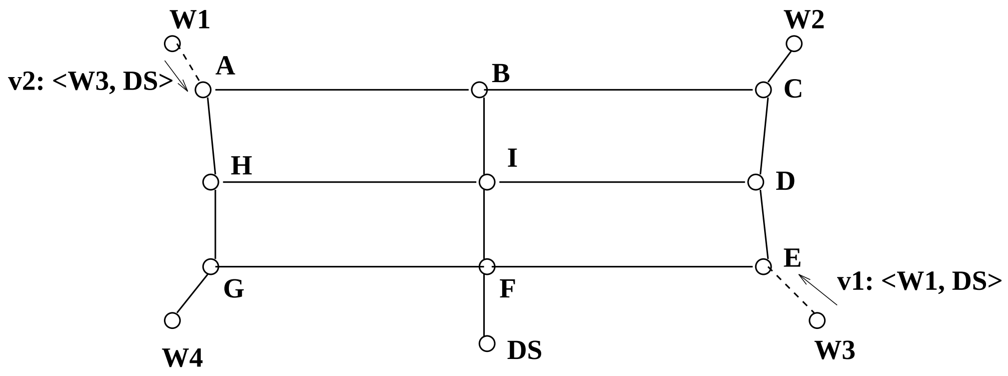
The characterization of the AGV RAS state s to be evaluated by the Banker's algorithm in the formalism of the AGV RAS developed in Section 2, is provided in Figure (6,b). Under the vehicle positioning depicted in this figure, the guidepath subgraph induced by the system free links presents three inter-connected components, defining the node equivalence classes: $C_1 = \{W_1, A, H, G, W_4\}$, $C_2 = \{W_2, C, D, E, F, I, B, DS\}$, and $C_3 = \{W_3\}$. Application of Algorithm 2 indicates that vehicle v_4 can complete its "mission" trip, visiting first node W_1, and then returning to node W_2 through the freed edge (G, F). Subsequently, vehicle v_1 can use the freed edge (G, F) to reach W_1 and the docking station, DS, in that order. It is easy to see that, by the time vehicle v_1 completes its trip, the guidepath subgraph induced by free and *freed* edges is fully connected, and therefore, the remaining vehicles v_2 and v_3 can complete their trip in any order. Hence, state s is ordered, and therefore, vehicle v_4 can proceed with the acquisition and traversal of zone FG.

Next consider the situation depicted in Figure (7,a). In this case, vehicle v_1 has just been loaded with a part destined to workstation W_1, while vehicle v_2 has just unloaded a part at workstation W_1. Workstation W_1 possesses also a part that needs to be transferred to workstation W_3, and the AGV system controller contemplates assigning vehicle v_2 to this transfer request. However, before this assignment takes place, the system controller must evaluate its safety. To this end, it uses Banker's algorithm in order to test the admissibility of the resulting AGV RAS state, s'. The formal representation of this state is depicted in Figure (7,b). Applying the logic of Algorithm 2, it should be clear that under the depicted nodal inter-connectivity status of the system guidepath graph, no vehicle can complete its trip; hence, state s' is not ordered.

Notice, however, that state s' is safe, since advancing any of the two vehicles to one of its neighboring edges results in an ordered state. From a theoretical standpoint, the existence of unordered safe states, demonstrated by this example, implies that, in the general case, the AGV Banker's algorithm will be a *suboptimal* solution to the AGV deadlock problem, i.e., some vehicle movements will be unnecessarily restricted by the SCP. This is the cost to be paid for the attained computational efficiency. Furthermore, in the considered situation, the apparent inefficiency of Banker's SCP, indicated by the non-admissibility of the (safe) state s', can be effectively overcome at the "performance-control" level. Specifically, an efficient performance-oriented control policy should recognize that, under the described situation, the best move for vehicle v_2 would perhaps be to wait on edge (W_1, A), until vehicle v_1 is safely moved out of edge (W_3, E), and then undertake the new assignment. So, this is a characteristic case where the *"no move"* option seems to be the most efficient.



(a)



(b)

Figure 7: Example: Running the AGV Banker's algorithm on an unordered state

4 Discussion

In this section, we present some extensions and modifications of the basic AGV RAS theory of Sections 2 and 3, which accommodate and/or exploit additional operational features of contemporary AGV systems. Specifically, we show that: (i) Banker's logic can be applied as an effective structural control policy even in cases that the vehicle trips are a priori constrained on specific routes, (ii) the proposed AGV RAS modeling and control logic can be easily extended to AGV systems in which certain links/zones can accommodate more than one vehicle at a time, and (iii) AGV system control can be easily integrated with higher-level policies controlling the material-flow among the system workstations, to ensure smooth and uninterrupted product flow.

AGV structural control with a priori constrained vehicle routes The development of Banker's algorithm in Section 3 was based on the assumption that in order to exploit the benefits of dynamic routing, a vehicle can be routed through any available path able to support the implementation of a given leg of its "mission" trip. The detailed selection of the path to be taken, in case that more than one options are available, is left to the performance-oriented control policy.

It might be the case, however, that in some practical implementations, the actual vehicle routings between a given pair of nodes are restricted to a subset of the physically possible routes, due to efficiency or other operational considerations. Then, Algorithm 4 is not applicable any more. All the same, the generic Banker's logic can still be applied, by evaluating node inter-connectivity in the tests of Algorithm 2, independently for each vehicle $v \in V(s)$. Specifically, the new Banker's implementation will not use data structure $\mathcal{E}$ anymore, but it will employ a *static* data structure, $\mathcal{A} :\ N \times N$, which indicates for each pair of guidepath graph nodes, (i, j), the subgraph induced by the admissible paths connecting node i to node j. The inter-connectivity of nodes i, j can be determined by checking whether there exists in subgraph $\mathcal{A}(i, j)$, a path from i to j, consisting of currently free edges. This last test can be performed through the "reaching" algorithm suggested at the end of Section 3.2. Hence, the entire scheme remains polynomial in the size of the AGV RAS.

AGV RAS models with multi-lane zones In some large AGV facilities, it is possible that (some of) the AGV aisles are wide enough to support more than one lane of traffic [19]. The most straightforward way to model this situation is by introducing in the AGV RAS model one guidepath graph edge for every single lane. Under this representation, the algorithms of Section 3 apply without any further modification. One problem with this modeling approach, however, is that it increases the size of the guidepath graph, and therefore, the complexity of the Banker's algorithm. A more efficient approach is to model each multi-lane link by a single edge, e, in the RAS guidepath graph, but further associate with it a non-unit capacity, C_e, equal to the number of the link lanes. Under this approach, the logic of the algorithms of Section 3 must be slightly modified, so that an edge is treated as *free*, iff it is not allocated to capacity.

A case of special interest is AGV systems in which every guidepath link has at least two lanes. Then, according to a result presented in [32], all unsafe states in the AGV operational space are deadlock states. Since deadlock states in AGV RAS are *polynomially* recognizable [30, 32], it follows that for this special class of AGV RAS, the optimal SCP is polynomially computable through one-step lookahead for deadlock. The implementational details of this optimal control scheme can be found in [30, 32].

Co-ordinating AGV and Workstation-level material flow An additional type of deadlock that can arise in AGV material-handling systems is between a set of vehicles and a workstation filled to capacity with WIP. Such a situation is discussed in [10], where a vehicle transporting a part to a workstation, finds it filled to capacity with WIP. As a result, the vehicle cannot unload its part, and furthermore, it blocks any other vehicle that might come to resolve the problem by picking some part(s) from the workstation output buffer.

An approach to deal with this problem is through *hierarchical decomposition* of the resource allocation system modeling the entire manufacturing shop-floor, to a sub-RAS modeling the material flow between the system workstations, and a sub-RAS modeling the AGV material handling system. These two sub-RAS's interact through a *master-slave* relationship. Specifically, buffering capacity of the system workstations is managed independently from the AGV system, by means of one of the existing correct and scalable SCP's for flexibly automated production systems (e.g., one of the policies presented in [31]). According to these policies, a condition for allowing a part to move to its next required station, is the existence of free buffering capacity. Hence, a part entering the AGV system has already reserved capacity at the receiving station. In other words, from the perspective of the RAS modeling the workstation resource management, the AGV system is only the *facilitator* of the between-the-station part movements. On the other hand, for the AGV modeling RAS, the workstation RAS model is the *generator* of the transport tasks executed by it. The logical consistency and the feasibility of the part flow between the various workstations is beyond the control domain of this system. The role of the AGV performance and structural control is to ensure that all the transport tasks commanded to this RAS from above, are met promptly and in an orderly fashion. An extensive investigation of the problem of hierarchical decomposition and control of resource allocation systems, resulting from the need to simultaneously manage the buffering capacity of the shop-floor workstations and the interconnecting material handling system, as well as more general cellular structures, can be found in [26].

5 Conclusions

The starting point and motivation for this work has been the observation that most existing AGV conflict resolution strategies are either non-robust to the system stochasticities/randomness, or they establish their robustness at the cost of restricted vehicle routings and the need for complicated handling operations. In this paper, we have proposed a dynamic, *closed-loop* control scheme, which manages the AGV dispatching and routing on the basis of real-time feedback on the system traffic. A particular characteristic of the proposed approach is that it further emphasizes robustness by discriminating between logical/structural and performance-oriented control. Specifically, a structural control policy determines the set of logically admissible moves out of the entire set of physically feasible vehicle movements, and then, the performance-oriented control selects the most efficient option among the admissible ones. A formal analysis of the AGV structural control problem, and the synthesis of effective and efficient structural control policies, were the main contributions of this paper.

Part of our future work will seek the simulation and lab-based testing of the developed structural control schemes. This implementational work will be particularly useful in studying the second part of the proposed control architecture, i.e., the development of performance-oriented, closed-loop, AGV control policies, and their interfacing to AGV structural control. To this end, the theory of *(Semi-)Markovian Decision Processes* [28] and its associated field of *Neuro-Dynamic Programming* [4] seem to provide an interesting modeling framework. Results obtained along these lines will be the content of a sequel paper.

Finally, we would like to indicate that the results developed herein have some additional significance and value, beyond the AGV control context:

1. The formal AGV RAS modeling and analysis, as well as the application of Banker's algorithm in this context, extends the existing Resource Allocation System theory [31] to a new RAS class, in which processes are characterized by the sequence of certain *"milestone"* events that

must be achieved, without any further specification of the intermediate steps. In this new RAS context, a challenging problem has been the effective management of the complexity resulting from the extensive set of options associated with the execution of each individual process. It turns out that the AGV RAS formalism can be a very effective tool for the visualization and analysis of such a RAS. Also, the version of the Banker's algorithm developed herein is an effective and efficient SCP for this broader class of systems.

2. The main structure and operational features presented by contemporary AGV systems are very similar to the operational characteristics of some other transportation networks, like inter-city and airport mono-rail systems, or multi-cell, multi-motion elevator systems recently contemplated for some large building blocks. It is expected that the control approaches proposed herein will be applicable in these additional application contexts. In fact, as it was already mentioned, researchers in these fields have started contemplating the advantages of dynamic, closed-loop control of these systems, and the underlying resource allocation modeling framework(s) [27]. The implementational details for such a "technology transfer", however, remain to be investigated.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] T. Araki, Y. Sugiyama, and T. Kasami. Complexity of the deadlock avoidance problem. In *2nd IBM Symp. on Mathematical Foundations of Computer Science*, pages 229–257. IBM, 1977.

[3] Z. A. Banaszak and B. H. Krogh. Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows. *IEEE Trans. on Robotics and Automation*, 6:724–734, 1990.

[4] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Boston, MA, 1996.

[5] U. Bilge and G. Ulusoy. A time window approach to simultaneous scheduling of machines and material handling system in an fms. *Op. Res.*, 43:1058–1070, 1995.

[6] Y. A. Bozer and M. M. Srinivasan. Tandem configurations for agv systems offer simplicity and flexibility. *Industrial Eng.*, 21:23–27, 1989.

[7] Y. A. Bozer and M. M. Srinivasan. Tandem configurations for automated guided vehicle systems and the analysis of single vehicle loops. *IIE Trans.*, 23:72–82, 1991.

[8] E. G. Coffman, M. J. Elphick, and A. Shoshani. System deadlocks. *Computing Surveys*, 3:67–78, 1971.

[9] K. A. Dowsland and A. M. Greaves. Collision avoidance in bi-directional agv systems. *J. Opl. Res. Soc.*, 45:817–826, 1994.

[10] P. J. Egbelu and J. M. A. Tanchoco. Characterization of automatic guided vehicle dispatching rules. *IJPR*, 22:359–374, 1984.

[11] P. J. Egbelu and J. M. A. Tanchoco. Potentials for bi-directional guide-path for automated guided vehicle based systems. *IJPR*, 24:1075–1097, 1986.

[12] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano. Event-based feedback control for deadlock avoidance in flexible production systems. *IEEE Trans. on Robotics and Automation*, 13:347–363, 1997.

[13] T. Ganesharajah, N. G. Hall, and C. Sriskandarajah. Design and operational issues in agv-served manufacturing systems. *Annals of OR*, 76:109–154, 1998.

[14] E. M. Gold. Deadlock prediction: Easy and difficult cases. *SIAM Journal of Computing*, 7:320–336, 1978.

[15] A. N. Habermann. Prevention of system deadlocks. *Comm. ACM*, 12:373–377, 1969.

[16] R. D. Holt. Some deadlock properties of computer systems. *ACM Computing Surveys*, 4:179–196, 1972.

[17] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley, Reading, MA, 1979.

[18] J. Huang, U. S. Palekar, and S. G. Kapoor. A labeling algorithm for the navigation of automated guided vehicles. *Journ. of Eng. for Industry*, 115:315–321, 1993.

[19] C. W. Kim and J. M. A. Tanchoco. Operational control of bidirectional automated guided vehicle system. *IJPR*, 31:2123–2138, 1993.

[20] C. W. Kim, J. M. A. Tanchoco, and P.-H. Koo. Deadlock prevention in manufacturing systems with agv systems: Banker's algorithm approach. *Manufacturing Science and Engineering Trans. of the ASME*, 1997.

[21] N. N. Krishnamurthy, R. Batta, and M. H. Karwan. Developing conflict-free routes for automated guided vehicles. *Oper. Res.*, 41:1077–1090, 1993.

[22] M. Lawley, S. Reveliotis, and P. Ferreira. Design guidelines for deadlock handling strategies in flexible manufacturing systems. *Intl. Jrnl. of Flexible Manufacturing Systems*, 9:5–29, 1997.

[23] M. Lawley, S. Reveliotis, and P. Ferreira. The application and evaluation of banker's algorithm for deadlock-free buffer space allocation in flexible manufacturing systems. *Intl. Jrnl. of Flexible Manufacturing Systems*, 10:73–100, 1998.

[24] C.-C. Lee and J. T. Lin. Deadlock prediction and avoidance based on petri nets for zone-control automated guided vehicle systems. *IJPR*, 33:3249–3265, 1995.

[25] W. L. Maxwell and J. A. Muckstadt. Design of automatic guided vehicle systems. *IIE Trans.*, 14:114–124, 1982.

[26] J. Park, S. A. Reveliotis, D. A. Bodner, and L. F. McGinnis. A distributed, event-driven control architecture for flexibly automated manufacturing systems. Technical Report (submitted to the IEEE Trans. on Systems, Man & Cybernetics, special issue on Discrete Event Systems and Control), School of Industrial & Systems Engineering, Georgia Institute of Technology, 1999.

[27] G. Parodi, G. Vernazza, and R. Zunino. Stability and deadlock avoidance in distributed systems for traffic control. *IEEE Trans. on Vehicular Technology*, 45:732–743, 1996.

[28] M. L. Puterman. *Markov Decision Processes.* John Wiley & Sons, Inc., NY, NY, 1994.

[29] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77:81–98, 1989.

[30] S. A. Reveliotis. Accommodating fms operational contingencies through routing flexibility. *IEEE Trans. on R&A*, 15:3–19, 1999.

[31] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira. Structural control of large-scale flexibly automated manufacturing systems. In C. T. Leondes, editor, *Computer Aided and Integrated Manufacturing Systems: Techniques and Applications*. GORDON & BREACH (to appear).

[32] S. A. Reveliotis, M. A. Lawley, and P. M. Ferreira. Polynomial complexity deadlock avoidance policies for sequential resource allocation systems. *IEEE Trans. on Automatic Control*, 42:1344–1357, 1997.

[33] G. F. Schwind. Trends in automatic guided vehicle systems. *Material Handling Engineering*, pages 49–52, 1996.

[34] M. Singhal. Deadlock detection in distributed systems. *IEEE Computer*, 22:37–48, 1989.

[35] D. Sinriech and J. M. A. Tanchoco. An introduction to the segmented flow approach for discrete material flow systems. *IJPR*, 33:3381–3410, 1995.

[36] D. Sinriech, J. M. A. Tanchoco, and Y. T. Herer. The segmented bidirectional single-loop topology for material flow systems. *IIE Trans.*, 28:4–54, 1996.

[37] F. Taghaboni-Dutta and J. M. A. Tanchoco. Comparison of dynamic routeing techniques for automated guided vehicle system. *IJPR*, 33:2653–2669, 1995.

[38] N. Viswanadham, Y. Narahari, and T. L. Johnson. Deadlock avoidance in flexible manufacturing systems using petri net models. *IEEE Trans. on Robotics and Automation*, 6:713–722, 1990.

[39] R. A. Wysk, N. S. Yang, and S. Joshi. Resolution of deadlocks in flexible manufacturing systems: Avoidance and recovery approaches. *Journal of Manufacturing Systems*, 13:128–138, 1994.

[40] K. Y. Xing, B. S. Hu, and H. X. Chen. Deadlock avoidance policy for petri net modeling of flexible manufacturing systems with shared resources. *IEEE Trans. on Aut. Control*, 41:289–295, 1996.

[41] L. Zeng, H. P. Wang, and S. Jin. Conflict detection of automated guided vehicles: a petri net approach. *IJPR*, 29:865–879, 1991.