Assessing and restoring "traffic-state order" in open, irreversible, dynamically routed, zone-controlled guidepath-based transport systems

Spyros Reveliotis

***Abstract*—The notion of the "h-ordered" traffic state provides an efficient approach for maintaining liveness in an open, irreversible, dynamically routed zone-controlled guidepath-based transport system. The restriction of these transport systems in their class of h-ordered states can be performed with polynomial complexity with respect to the size of these systems, while the resulting supervisory control policy retains high levels of operational latitude. The work presented in this paper provides novel efficient algorithms for the following two problems: (i) assessing whether a given traffic state is h-ordered, and (ii) bringing the underlying transport system from some general traffic state to the class of its h-ordered states in a way that minimizes a certain measure of "operational disruption". The developed algorithms are motivated by, and find immediate applicability, in the Model Predictive Control (MPC) scheme for the considered transport systems that was developed in [1].**

***Note to Practitioners* – Open and irreversible, zone-controlled, guidepath-based transport systems is a natural abstraction of the traffic dynamics taking place in many unit-load material handling systems (MHSs), like the automated guided vehicle (AGV) systems and the overhead monorail systems that are used in many industrial facilities. In these environments, vehicles are circulating in a "guidepath network" that is defined either by the physical structure of the corresponding MHS (as in the case of the overhead monorail systems) or more artificially, in an effort to isolate the traffic of these vehicles from the surrounding environment (as in the case of the AGV systems). Furthermore, in order to ensure collision-free motion for the traveling vehicles, the edges of this guidepath network are divided into zones, and it is stipulated that each zone is allocated to at most one vehicle at any time. This restriction renders the considered transport systems susceptible to deadlock, and therefore, their traffic controller must control the generated traffic for ensuring a high productivity, but also for ensuring "traffic liveness", i.e., the ability of every vehicle to complete successfully its current assignment and engage in similar assignments in the future. An effective and computationally efficient manner to maintain traffic liveness is by restricting the considered transport systems within a particular subclass of traffic states that is known as h-ordered. The considered paper provides streamlined, customized algorithms for effecting this restriction.**

Keywords: Guidepath-based transport systems; traffic liveness enforcement; deadlock avoidance; discrete event systems; model predictive control

S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology, email: spyros@isye.gatech.edu. He was partially supported by NSF grant ECCS-1707695.

I. Introduction

The *zone-controlled guidepath-based transport system (ZC-GBTS)* is an established abstraction for modeling the traffic dynamics that take place in many popular unit-load material handling systems (MHS), like the automated guided vehicle (AGV) systems and the overhead monorail systems that are used in many production and distribution facilities [2]. The vehicles traveling in such a system serve transport requests between different pairs of pick-up and drop-off locations by moving through a network of interconnected corridors that is known as the underlying "guidepath network". Furthermore, in order to avoid collisions among these vehicles, the corridors of the guidepath network are split into zones, and it is stipulated that these zones cannot be occupied simultaneously by two or more vehicles at any timepoint. Hence, a vehicle can move from its current zone to a neighboring one only when the claimed zone is currently free, and such a zone transition by any given vehicle must be authorized by a traffic coordinator.

Some additional features that characterize the ZC-GBTSs considered in this work, and are crucial for the presented results, are as follows: (i) The considered transport systems avail of a "home" zone that accommodates all idle vehicles, and provides additional maintenance services to them, like the recharging of their batteries [3], [4]. From the standpoint of the operational analysis that is pursued in this work, vehicles located in this "home" zone can be perceived as being outside the primary guidepath network, and therefore, the corresponding ZC-GBTSs are characterized as "open". (ii) In addition, the narrowness of the aisles that define the various zones of the guidepath network and other safety considerations prevent the reversal of the motion of a vehicle within its current zone. Hence, the considered ZC-GBTSs are characterized as "irreversible". (iii) Finally, the routing of the vehicles to their destinations is resolved by the traffic coordinator in an incremental manner that accounts for the currently experienced congestion within the network and the routing flexibility that is provided by the overall connectivity of this network. Hence, the considered ZC-GBTSs are also characterized as "dynamically routed".

The considered ZC-GBTSs must be controlled in a way that maximizes the executed transports and minimizes the experienced delays. The resulting problem is a hard combinatorial optimization problem that becomes further aggravated by the dynamic arrival of the transport requests [4], [5], [6]. In view of this very high complexity, the work of [1] has

proposed a Model Predictive Control (MPC) [7] framework for an effective and efficient resolution of this traffic management problem. This MPC framework assumes that, at any timepoint, every agent is assigned a (possibly empty) sequence of transport tasks, and decomposes the overall routing and scheduling problem to a series of subproblems that seek to efficiently route the system agents to their most immediate destinations. Every time that an agent reaches its current destination, it is reassigned to its next destination (possibly the "home" zone, if it currently has no further assignments), and the routing plans of all agents are revised in order to accommodate this new development, by formulating and solving a new subproblem.

An issue that arises in the above MPC scheme, but also in any other traffic management scheme for the considered class of transport systems, is the preservation of the traffic "liveness", i.e., of the ability of every agent to complete successfully its current transport assignment and engage in similar transport requests in the future. Loss of liveness in the operations of the considered ZC-GBTS can result from the formation of deadlocks and livelocks within a subset of the traveling agents [8].

In [8] it is shown that traffic liveness can be attained by modeling the qualitative dynamics of the considered ZC-GBTSs through a finite state automaton (FSA) [9], and restricting the system traffic within a particular class of states of this FSA. These states satisfy a well-defined property with respect to (w.r.t.) the dynamics of the underlying FSA, and are characterized as "live".[1] Also, the resulting supervisor is the *maximally permissive* "liveness-enforcing supervisor (LES)". In the MPC scheme of [1], maximally permissive liveness-enforcing supervision can be attained by ensuring that the traffic states that are defined by the immediate destinations of the traveling agents as the target traffic states for the subproblems that are formulated and solved in the context of this MPC scheme, are live.

However, the computational complexity of assessing the liveness of any given traffic state from the considered class of ZC-GBTS is an open problem. In view of this reality, the literature has tried to develop non-maximally permissive LES for ZC-GBTS that are computationally efficient. Some of the most indicative examples of these endeavors can be found in [10], [11], [12], [13], [14]. Among these approaches, a method of particular interest to this work tries to restrict the considered ZC-GBTS within a sub-class of their live states that is polynomially recognizable w.r.t. the size of the corresponding ZC-GBTS, and is known as "h-ordered" [10]. The corresponding LES essentially constitutes an adaptation of Banker's algorithm that has been proposed as an efficient LES for more general sequential resource allocation systems (RAS) [15], [16], [17]. Collective past experience with the application of Banker's algorithm in the liveness-enforcing supervision of sequential RAS indicates that the algorithm can provide a significant coverage of the behavioral space of the corresponding maximally permissive LES. On the other hand, the adaptation of this algorithm in the operational context of the considered ZC-GBTS, in a way that preserves the computational efficiency of the original algorithm, is a nontrivial task due to the implicit representation of the available routes to the traveling vehicles by means of the underlying guidepath network [10].

This work exploits a graphical representation of the traffic state of the considered transport systems, in order to efficiently assess whether a given traffic state from the considered ZC-GBTS class is h-ordered. Furthermore, it provides additional algorithms that can construct an h-ordered traffic state from a traffic state that does not possess this property; the h-ordered state is obtained from the original traffic state by relocating some of the traveling agents to the "home" zone, in way that some "disruption" cost that will result from this relocation is minimized. Both of these problems are motivated by, and have immediate application in the MPC scheme of [1], since they enable a pertinent evaluation, and the potential redefinition, of the target traffic states that are employed in the subproblems that are solved in the context of this MPC scheme.

In view of the above positioning of the paper content and its intended contribution, the rest of it is organized as follows: The next section provides a more formal description of (i) the considered ZC-GBTS, (ii) the problem of the liveness-enforcing supervision for its generated traffic, and (iii) the notion of the h-ordered traffic state. Section III introduces the novel algorithm for the decision problem of assessing whether a given traffic state of an open, irreversible, dynamically routed ZC-GBTS is h-ordered. Section IV introduces the additional problem of bringing an open and irreversible ZC-GBTS in its class of h-ordered states with minimal disruption for the underlying operation, and the corresponding algorithms for its solution. Finally, Section V concludes the paper and suggests some directions for potential future work. Furthermore, due to the imposed limits on the length of this manuscript, we are providing the formal proofs of the technical results of this paper, and some of the supporting examples, in an electronic supplement that is accessible at: `https://www2.isye.gatech.edu/~spyros/ho-sup.pdf`.

II. The considered ZC-GBTS and the problem of the liveness-enforcing supervision of its generated traffic

This section provides a formal description of the open, irreversible, dynamically routed ZC-GBTS, the notion of liveness of the corresponding traffic, and the problem of the liveness-enforcing supervision that is defined in this context. Due to the imposed length limitations for this document, the provided material is the minimum necessary for ensuring the integrity of this document; more expansive treatments can be found in [8], [1].

A. Formal characterization of the structure and the traffic dynamics of the considered ZC-GBTS

An open, irreversible, dynamically routed ZC-GBTS is formally represented by a pair $(\mathcal{A}, G)$, where: (a) $\mathcal{A}$ denotes the set of the system vehicles – or, more generally, "agents" –

[1] We provide more formal characterizations of all these concepts and results in the next section.

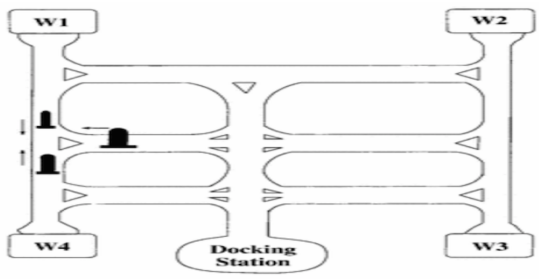

Fig. 1: An abstracting representation of an AGV system and an AGV deadlock.

circulating in it, and (b) $G = (V, E \cup \{h\})$ is a multi-graph[2] representing the guidepath network that is traversed by these agents.

Graph G is assumed to be undirected and connected. Also, the minimum vertex degree of G is 2, since, under the presumed irreversibility of the considered transport systems, an agent a reaching a vertex v of degree 1 would deadlock at that vertex.

The edges $e \in E$ of G model the "zones" of the underlying guidepath network. These edges can be traversed by a traveling agent $a \in \mathcal{A}$ in either direction, but they can hold no more than one agent at a time.

On the other hand, edge h models the "home" zone of the guidepath network. Edge h is connected to the rest of the guidepath network through a single vertex v_h (i.e., edge h is a self-loop of G), and it can hold an arbitrary number of agents.

A "mission" trip for an agent $a \in \mathcal{A}$ is defined by a sequence of edges $\Sigma_a = \langle e_i \in E \setminus \{h\} \rangle$ that must be visited by agent a in the specified order. The edges e_i in sequence Σ_a are the successive destinations for agent a, and the agent can follow any feasible walk[3] on guidepath graph G when moving from edge e_i to edge e_{i+1}. Furthermore, "home" edge h is an implicit last edge in every sequence Σ_a, since each agent a that has completed its mission trip, must retire at this location.

An agent a traversing an edge $e \in E$ with $e = \{v_i, v_j\}$ has a certain direction of motion on this edge that is indicated by the corresponding ordered pair (v_i, v_j) or (v_j, v_i). Furthermore, the presumed irreversibility of the agent motion stipulates that an agent a entering edge $e = \{v_i, v_j\}$ from vertex v_i must leave this edge through vertex v_j, and vice versa.

Finally, an agent a cannot move in an edge e from a neighboring edge e', unless edge e is currently empty. This stipulation seeks to establish adequate separation among the traveling agents even during the transitional phases that an agent is moving between two different but neighboring edges. It also implies that two agents cannot "swap" the occupation of two neighboring edges.

[2]In general, two vertices v_1, v_2 of graph G may be connected by more than one zones. But this feature does not impact substantially our subsequent developments, and we shall keep referring to G as a graph in the sequel.

[3]We remind the reader that a walk in an undirected graph G is a sequence $\langle v_0, e_1, v_1, \ldots, v_{i-1}, e_i, v_i, \ldots, v_{k-1}, e_k, v_k \rangle$ where, for all $i = 1, \ldots, k$, edge e_i is incident upon the vertices v_{i-1} and v_i.

B. Traffic deadlock, traffic state liveness, and the need for liveness-enforcing supervision

The motion irreversibility and the other traffic restrictions for the system agents that were defined in the previous subsection, when combined with the arbitrary topology of the guidepath graph G and the bidirectional traversal of its edges by the traveling agents, can give rise to the formation of deadlocking situations similar to that depicted in Figure 1, where each of the three depicted AGVs is blocked in its further advancement by the presence of the other two vehicles.

The problem of preventing these deadlock formations is known as the problem of the *liveness-enforcing supervision (LES)* for the considered transport systems. Next, we overview some important concepts and results for this supervisory control problem following the corresponding developments of [8], which constitutes an effort to collect and organize the major results that are currently available for this problem. Central to all these developments, is the following characterization of the "traffic state":

Definition 1: For the needs of the subsequent developments, the *state* $s(t)$ of any open, irreversible, dynamically routed ZC-GBTS, at some timepoint t, is defined by (i) the distribution of the agents $a \in \mathcal{A}$ to the zones $E \cup \{h\}$ of the underlying guidepath network G, together with (ii) the information about the direction of the agents $a \in \mathcal{A}$ that are located in zones $e \in E$.

It is further assumed that state $s(t)$ is *valid*, i.e., every edge $e \in E$ is allocated to no more than one agent.

Finally, the *"home" state* s_h is the state where every agent $a \in \mathcal{A}$ is located in the "home" zone h. □

In the following, we shall use the notation s instead of $s(t)$ for the traffic state, and we shall denote the entire set of valid traffic states s by S. Clearly, state set S is finite. Also, state s will be represented graphically by a *labeled partially directed digraph (PDG)*, $\hat{G}(s)$. This PDG is induced from the original undirected graph G through (i) the labeling of each zone $e \in E$ that is allocated to some agent $a \in \mathcal{A}$ by the name of the corresponding agent, and (ii) the turning of edge e into a directed edge with its sense of direction indicating the direction of motion of agent a in the corresponding zone.

We shall also use the notation $\gamma(a; s)$ to denote the zone of agent $a \in \mathcal{A}$ in state s. In the PDG-based representation of state s, $\gamma(a; s)$ should be understood as a directed edge, if agent a is located in a zone other than the "home" zone h; in the opposite case, $\gamma(a; s)$ corresponds to an undirected edge.

Finally, state s evolves by advancing a subset of agents $\hat{\mathcal{A}} \subseteq \mathcal{A}$ from their current zones, $\gamma(a; s)$, to some neighboring zones that are empty in s. Furthermore, the advancing agents must be selected in a way that ensures the validity of the resulting state s'.

In the FSA that results from the above characterizations, the notion of "traffic liveness" for the considered class of transport systems implies the preservation of the ability of every agent $a \in \mathcal{A}$ to reach each edge $e \in E \cup \{h\}$ of G *ad infinitum*. In [8] it is shown that, for these systems, traffic liveness can be equivalently characterized through the following result:

Proposition 1: An open ZC-GBTS is live if and only if (*iff*) every reachable traffic state s is co-reachable[4] to the "home" state s_h. □

Proposition 1 also motivates the following definition.

Definition 2: A traffic state s of an open ZC-GBTS is characterized as *live iff* it is co-reachable to the "home" state s_h. The entire set of live traffic states will be denoted by S_l. □

At any traffic state s of an open ZC-GBTS, the *maximally permissive LES* will allow the transition of an agent $a \in \mathcal{A}$ from its current zone $\gamma(a; s)$ to a neighboring free zone e *iff* the traffic state s' that will result from this transition is live. But, as discussed in the introductory section, in the ZC-GBTSs considered in this work, the deployment of the maximally permissive LES is challenged by the lack of an efficient algorithm for assessing state liveness. Hence, we usually seek the deployment of a suboptimal – i.e., non-maximally permissive – LES, based on some alternative property that will define the admissibility of any given traffic state s; this property (i) must preserve traffic liveness, and (ii) its assessment on any given traffic state $s \in S$ must incur a polynomial computational cost w.r.t. the size of the underlying transport system. Such a property is provided by the notion of the "h-ordered" traffic state for open and irreversible ZC-GBTS, which is defined as follows [1]:

Definition 3: A traffic state s of an open, irreversible, dynamically routed ZC-GBTS is *"h-ordered" iff* there exists an ordering $[\cdot] : \{1, \ldots, |\mathcal{A}|\} \rightarrow \mathcal{A}$, of the agent set $\mathcal{A}$, such that, for each agent $a_{[i]},\ i = 1, \ldots, |\mathcal{A}|$, there is a feasible route $\mathcal{R}_{a_{[i]}}$, under the agent-separation rules of Section II-A, that can take agent $a_{[i]}$ from its original zone $\gamma(a_{[i]}; s)$ to the "home" zone h, while agents $a_{[j]},\ j = i + 1, \ldots, |\mathcal{A}|$, maintain their original positions in state s. Also, the entire set of h-ordered traffic states of any given ZC-GBTS will be denoted by S_{ho}. □

In more plain terms, a traffic state s of an open, irreversible, dynamically routed ZC-GBTS is h-ordered *iff* it is possible to order the system agents that are not located in the "home" zone h, in a way that each agent a can advance from its current zone $\gamma(a; s)$ to the "home" zone h using only edges that are free in state s or have been freed by the advancement of the previous agents in this ordering.

It is clear from the above definition that every h-ordered state is live. Also, any state s with only one agent a having $\gamma(a; s) \neq h$ is h-ordered, and, therefore, the restriction of the operation of any open, irreversible, dynamically routed ZC-GBTS in the corresponding set S_{ho} will preserve traffic liveness. On the other hand, [1] outlined only very briefly an algorithm for assessing whether $s \in S_{ho}$, for any given traffic state $s \in S$. The next section employs the PDG-based representation of the traffic state of Definition 1 in order to provide a complete and much more streamlined algorithm for this decision problem.

[4]We remind the reader that in the FSA modeling framework, a state s is co-reachable to a state s' *iff* state s' is reachable from state s through a feasible event sequence.

Algorithm 1 An efficient algorithm for testing whether $s \in S_{ho}$, for any traffic state s of an open, irreversible, dynamically routed ZC-GBTS.

Input: The corresponding PDG $\hat{G}(s)$.
Output: $ORDERED$: a Boolean variable indicating whether $s \in S_{ho}$.

Construct the digraph $\hat{G}(s')$ from the PDG $\hat{G}(s)$ according to the construction procedure of Figure 2;
$\mathcal{G} := \hat{G}(s')$;
while $\mathcal{G}$ contains an edge $e = (v_1(e), v_2(e))$ with $v_2(e) = v_h$ **do**
  Merge vertex $v_1(e)$ with vertex v_h and remove edge e from $\mathcal{G}$;
end while
$ORDERED := I_{\{\mathcal{G} \text{ is reduced to the single vertex } v_h\}}$;
return $ORDERED$.

III. An efficient algorithm for recognizing h-ordered states

In this section we present a novel efficient algorithm for the decision problem '$s \in S_{ho}$?', that was introduced in the previous section. The corresponding pseudocode is presented in Algorithm 1.

The execution of Algorithm 1 on any given traffic state s evolves in two major phases: First, the algorithm constructs from the input state s another state s' that is h-ordered *iff* state s is h-ordered. In addition, the new state s' is constructed in a way that (a) guarantees that state s' is h-ordered *iff* it is live, and (b) enables an efficient assessment of its liveness based on some existing results; hence, the evaluation of the liveness of state s' constitutes the second phase of Algorithm 1. Next we detail these two phases.

The construction of the aforementioned traffic state s' from the input state s must guarantee the following three properties:

1) State s' involves the same set of traveling vehicles $\mathcal{A}$ with the original state s.
2) It is h-ordered *iff* the original state s is h-ordered.
3) It has a PDG $\hat{G}(s')$ with no undirected edges (i.e., the PDG $\hat{G}(s')$ is a digraph).

Properties 1 and 2 imply that we can assess whether the original traffic state s is h-ordered by actually assessing the same property for the induced traffic state s'. On the other hand, Property 3 of the induced traffic state s' enables the more efficient assessment of this new traffic state according to the logic that was discussed in the opening part of this section.

Algorithm 1 computes the sought traffic state s' by constructing the corresponding PDG $\hat{G}(s')$ from the original PDG $\hat{G}(s)$, according to the procedure that is described in Figure 2. The next example demonstrates this construction.

Example: This example demonstrates the state construction of Figure 2 by means of the PDG $\hat{G}(s)$ that is presented at the left part of Figure 3. The state s that corresponds to this PDG involves five agents, $a_1, \ldots, a_5$, located on the directed edges of this graph that are labeled by the corresponding labels. Each agent $a_i,\ i = 1, \ldots, 5$, is heading in the direction that

1) Compute the maximal connected subgraphs of the PDG $\hat{G}(s)$ that consist of undirected edges only. Let these subgraphs be denoted by $\Xi_1, \ldots, \Xi_l$.
2) For every subgraph Ξ_i that contains a cycle, identify the directed edges $e = \big(v_1(e), v_2(e)\big)$ of the PDG $\hat{G}(s)$ that have their end-vertex $v_2(e)$ located on Ξ_i, and for every such identified edge e, add in the original PDG $\hat{G}(s)$ an undirected edge $e' = \big\{v_1(e), v_2(e)\big\}$. Let $\hat{G}''$ denote the PDG that results from this augmentation.
3) Compute the digraph $\hat{G}(s')$, that is the graphical representation of the constructed traffic state s', from the PDG $\hat{G}''$, by collapsing into a single node each maximal connected subgraph of $\hat{G}''$ that consists of undirected edges only.

Fig. 2: The construction of the traffic state s' from the original traffic state s.

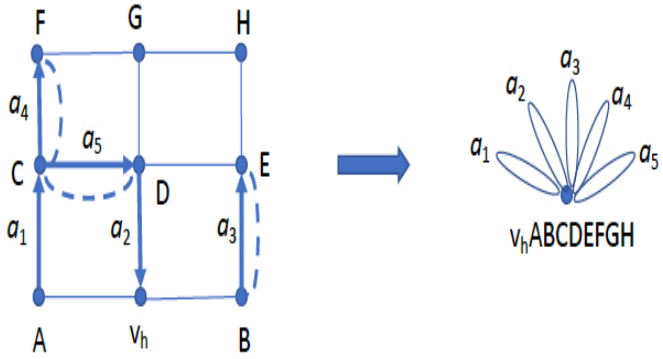

Fig. 3: An example demonstrating the construction of Figure 2.

is defined by the corresponding edge. PDG $\hat{G}(s)$ also has two maximal connected subgraphs consisting of undirected edges only, that are respectively defined by the two vertex sets $\{v_h, A, B\}$ and $\{D, E, F, G, H\}$, and the undirected edges that are incident upon these vertices.

The PDG depicted in the left part of Figure 3 also depicts, in dashed lines, the undirected edges that will be added to the PDG $\hat{G}(s)$ by the second step in the procedure that is defined in Figure 2. The addition of these undirected edges accounts for the fact that the agents a located in the corresponding directed edges e, can reverse the direction of their motion in their current zones by using the cycles of free edges in the corresponding subgraphs of free edges Ξ_i. A more detailed explanation of the necessity and the role of these additional edges in the pursued computation can be found in the proof of Proposition 2 that is provided in the electronic supplement.

In the PDG that results from the addition of the undirected edges, the maximal connected subgraph that consists of undirected edges (including the added ones) involves all the graph vertices. Therefore, Step 3 of the construction procedure in Figure 2 will collapse the entire graph into a new digraph $\hat{G}(s')$ that consists of (i) a single vertex labeled $v_hABCDEFGH$, since it represents all of the original vertices, and (ii) the directed edges of $\hat{G}(s)$, that correspond to the traveling agents $a_1, \ldots, a_5$; in the resulting digraph $\hat{G}(s')$ each of these directed edges is a self-loop of the vertex $v_hABCDEFGH$. □

The next proposition establishes the claimed Property 2 for the state s' that results from the construction procedure of Figure 2. The proof of this proposition can be found at the electronic supplement of this paper.

Proposition 2: Consider the traffic state s' that is obtained from a given traffic state s of an open and irreversible ZC-GBTS through the construction procedure of Figure 2. Then, the original traffic state s is h-ordered *iff* the constructed traffic state s' is h-ordered. □

Since the guidepath network of the constructed state s' has no free edges, state s' is "totally congested" according to the terminology of [18]. But then, we can apply to state s' the following proposition that was established in [18].

Proposition 3: Consider a traffic state $s \neq s_h$ of an open, irreversible, dynamically routed ZC-GBTS such that the corresponding graph $\hat{G}(s)$ contains no undirected edges. Then, state s is live *iff* every vertex $v \in V \setminus \{v_h\}$ of the digraph $\hat{G}(s)$ is co-reachable to vertex v_h. □

Proposition 3 provides the following efficient algorithm for assessing the liveness of a totally congested traffic state s: We start with the corresponding digraph $\hat{G}(s)$, and iteratively we merge into its node v_h every edge $e = (v, v_h)$ in this digraph, and also in all the digraphs that result from these mergers. If this merging process manages to collapse the entire digraph $\hat{G}(s)$ into a single vertex, then the considered state s is live; otherwise, state s is not live.

It is clear from the semantics of the above algorithm that every time that an edge (v, v_h) is merged in the node v_h, the agent a that occupies this edge in the considered state s is brought to the "home" edge h using the edges that have been released from the previous mergers. Hence, the merging process that is effected by this algorithm also defines an ordering of the system agents that satisfies the requirements of Definition 3. This remark implies the following corollary of Proposition 3.

Corollary 1: The state s' that is constructed by the procedure of Figure 2 is h-ordered *iff* it is live. □

The combination of Propositions 2 and 3 and of Corollary 1 implies that we can address the problem '$s \in S_{ho}$?' by running the above liveness-assessing algorithm for totally congested traffic states on the state s' that is obtained through the construction of Figure 2; this computation constitutes the second phase of Algorithm 1.

The next theorem recapitulates all the above discussion, stating formally the correctness of Algorithm 1.

Theorem 1: When applied on any given traffic state s coming from an open, irreversible, dynamically routed ZC-GBTS, Algorithm 1 will execute in finite time, and it will determine correctly whether $s \in S_{ho}$. □

Examples: It is obvious that the execution of the second part of Algorithm 1 on the digraph that is depicted in the right part of Figure 3, will remove all the five edges of this digraph, leading to a graph $\mathcal{G}$ that will consist only of the vertex $v_hABCDEFGH$. Hence, the traffic state s that is defined by the PDG that is depicted in the left part of Figure 3, is h-ordered.

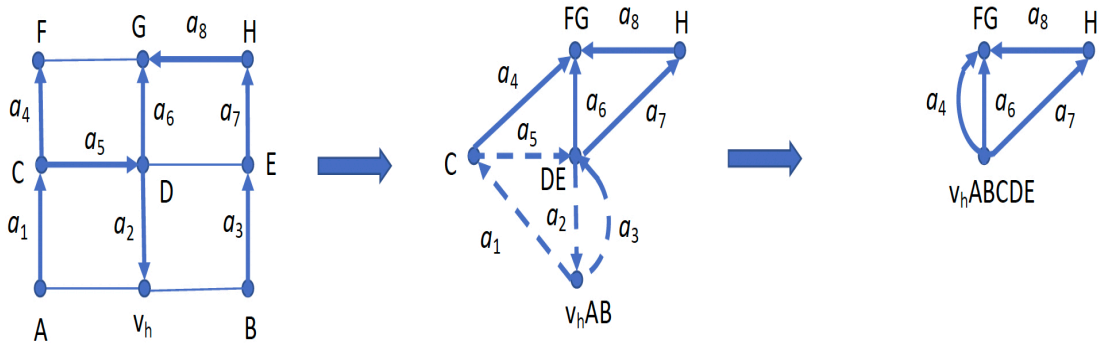

Fig. 4: The execution of Algorithm 1 on a traffic state $\tilde{s}$ that is not h-ordered.

On the other hand, Figure 4 presents the execution of Algorithm 1 on another PDG, $\hat{G}(\tilde{s})$, where the induced digraph $\hat{G}(\tilde{s}')$ cannot be reduced to a single vertex by the second phase of this algorithm. Hence, $\tilde{s} \notin S_{ho}$ in this case. The complete parsing of Figure 4 is as follows: The left part of the figure presents the original PDG $\hat{G}(\tilde{s})$ and its augmentation with the undirected edges e', according to the representational conventions that are also followed in Figure 3. The middle part of Figure 4 depicts the digraph $\hat{G}(\tilde{s}')$ that will be returned by the first phase of Algorithm 1. The edges that are depicted by dashed lines in this digraph are those that will be removed through the execution of the second phase of the algorithm. The final outcome of this second phase is the digraph $\mathcal{G}$ that is depicted in the right part of the figure. □

We conclude the presentation of Algorithm 1 by noticing that its worst-case computational complexity is $O(|E|)$; this result is systematically established in the electronic supplement.

IV. Restoring "traffic-state order" in the considered ZC-GBTS

In this section we consider the second problem that is addressed in this work. This problem concerns the construction of an h-ordered traffic state s' from a given traffic state $s \notin S_{ho}$ by relocating some of the traveling agents in state s to the "home" zone h of the underlying guidepath network. The selection of the agents to be relocated to the "home" zone h must be done in a way that minimizes some measure of "disruption" of the operation of the underlying ZC-GBTS. The first part of the section provides the formal characterization of the resulting optimization problem, while the remaining parts provide some algorithms for its resolution.

A. The "optimal order restoration" problem for the considered ZC-GBTS

A formal statement of the problem that is considered in this section is as follows:

Definition 4: The problem of "optimal restoration of traffic-state order": Consider a traffic state s of an open and irreversible ZC-GBTS that is not h-ordered. Also, let $\mathcal{A}'$ denote the set of agents $a \in \mathcal{A}$ with $\gamma(a;s) \neq h$, and further assume a function $c: \mathcal{A}' \to \mathbb{R}_0^+$, where $\mathbb{R}_0^+$ denotes the set of nonnegative reals. Finally, let $P(\mathcal{A}')$ denote the set of strict subsets of the set $\mathcal{A}'$ with the following property: For any $\mathcal{A}'' \in P(\mathcal{A}')$, the traffic state s' that is induced from the traffic state s by setting $\big(\gamma(a;s') := h,\ \forall a \in \mathcal{A}''\big) \wedge \big(\gamma(a;s') := \gamma(a;s),\ \forall a \in \mathcal{A} \setminus \mathcal{A}''\big)$, belongs in S_{ho}. We want to compute $\mathcal{A}^* \in P(\mathcal{A}')$ such that $\mathcal{A}^* = \arg\min_{\mathcal{A}'' \in P(\mathcal{A}')} \big\{\sum_{a \in \mathcal{A}''} c(a)\big\}$. □

In the above problem statement, function $c(\cdot)$ defines a "disruption" cost for every agent that is relocated to the "home" zone h. Hence, the considered problem seeks to restore traffic-state order in a way that minimizes the total "disruption" cost that will result from the effected relocations. In the rest of this section, we shall refer to this optimization problem as the *"optimal order restoration"* problem, for brevity.

In the MPC scheme of [1], that has provided the primary motivation for this problem, the input traffic states s for the addressed instances of the "optimal order restoration" problem will be those target states s of the various subproblems that are formulated by this MPC scheme, that are not h-ordered. Furthermore, the agent relocations that will be determined by the solution of the "optimal order restoration" problem in this particular context, will not take place in an immediate, physical sense, but they have the meaning of the redirection of the corresponding traveling agents from their next destinations to the "home" zone h, in the formulation of the corresponding subproblem.

Then, setting the function $c(\cdot)$ uniformly equal to 1.0 implies that we seek to obtain an h-ordered target state for the formulated subproblems while minimizing the number of the agents that must be redirected from their immediate destinations to the "home" zone h. Alternatively, function $c(\cdot)$ can also assign more general values to the traveling agents, that might express (a) the criticality of the various transport tasks that are executed by these agents, (b) a time-based priority that might be assigned to certain agents in an effort to prevent the indefinite postponement of the corresponding tasks, (c) the current proximity of the different agents to their prevalent destinations, and/or (d) other similar considerations.

Algorithm 2 A solution algorithm for the "optimal order restoration" problem of Definition 4.

Input: The corresponding PDG $\hat{G}(s)$; the cost function $c(\cdot): \mathcal{A}' \rightarrow \mathbb{R}_0^+$.
Output: $ORDERED$: a Boolean variable indicating whether $s \in S_{ho}$; $\mathcal{A}^*$: an agent set defining an optimal solution for the considered "optimal order restoration" problem.

$ORDERED$:= Algorithm 1$\big(\hat{G}(s)\big)$;
if $ORDERED$ **then**
  $\mathcal{A}^*$:= $\emptyset$;
  return $ORDERED$ and $\mathcal{A}^*$.
end if
QUEUE := NIL;
for each $e \in \mathcal{G}$ computed by Algorithm 1 **do**
  create the corresponding candidate solution $\langle\{a(e)\}, c(a(e)), \hat{G}(a(e))\rangle$ and enter it in $QUEUE$ preserving the desired ordering of the queue entries;
end for
while $TRUE$ **do**
  $\big\langle\hat{\mathcal{A}}, c(\hat{\mathcal{A}}), \hat{G}(\hat{\mathcal{A}})\big\rangle$:= Extract head entry from $QUEUE$;
  $ORDERED$:= Algorithm 1$\big(\hat{G}(\hat{\mathcal{A}})\big)$;
  if $ORDERED$ **then**
    $\mathcal{A}^*$:= $\hat{\mathcal{A}}$;
    return $\neg ORDERED$ and $\mathcal{A}^*$.
  else
    for each $e \in \mathcal{G}(\hat{\mathcal{A}})$ computed by Algorithm 1 **do**
      create the corresponding candidate solution $\big\langle\hat{\mathcal{A}} \cup \{a(e)\}, c(\hat{\mathcal{A}}) + c(a(e)), \hat{G}(\hat{\mathcal{A}}(e))\big\rangle$ and enter it in $QUEUE$ preserving the desired ordering of the queue entries and avoiding duplication;
    end for
  end if
end while

B. A general algorithm for the "optimal order restoration" problem

In this subsection, we consider the "optimal order restoration" problem with an arbitrary cost function $c(\cdot)$. The pseudocode for the basic algorithm that we propose for this general version of the "optimal order restoration" problem of Definition 4, is presented in Algorithm 2. This algorithm executes in two phases: (i) In the first phase, Algorithm 2 executes Algorithm 1 on the input PDG $\hat{G}(s)$ in order to identify the agent set $\mathcal{A}'' \subseteq \mathcal{A}'$ containing those agents that are in actual conflict in terms of the notion of the h-ordered state; more specifically, the set $\mathcal{A}''$ contains all those agents that label the edges of the digraph $\mathcal{G}$ that is computed by Algorithm 1. (ii) Subsequently, the second phase of Algorithm 2 effects a search over the elements of $P(\mathcal{A}'')$ for an optimal solution.

This search is organized according to a "best first" scheme that is facilitated by the following observations: Any given element $\hat{\mathcal{A}}$ of $P(\mathcal{A}'')$ is associated with the corresponding cost $c(\hat{\mathcal{A}}) \equiv \sum_{a \in \hat{\mathcal{A}}} c(a)$. In addition, the agent set $\hat{\mathcal{A}}$ induces a PDG $\hat{G}(\hat{\mathcal{A}})$; this PDG is obtained from the digraph $\mathcal{G}$ that was computed during the first phase of the algorithm, by turning the edges of $\mathcal{G}$ corresponding to the agents $a \in \hat{\mathcal{A}}$ into undirected edges. Finally, if the execution of Algorithm 1 on the PDG $\hat{G}(\hat{\mathcal{A}})$ results in a digraph $\mathcal{G}(\hat{\mathcal{A}})$ consisting of a single vertex only, then the considered set $\hat{\mathcal{A}}$ constitutes a feasible solution to the "optimal order restoration" problem, with corresponding cost $c(\hat{\mathcal{A}})$.

In view of the above remarks, Algorithm 2 maintains QUEUE, a priority queue of candidate solutions where each candidate solution is represented by a triplet $\langle\hat{\mathcal{A}}, c(\hat{\mathcal{A}}), \hat{G}(\hat{\mathcal{A}})\rangle$ and the stored candidates are ordered in increasing cost $c(\hat{\mathcal{A}})$. Entries of equal cost $c(\hat{\mathcal{A}})$ can be stored in any arbitrary order without compromising the correctness of the algorithm, but for complete specificity, we shall further assume that entries of equal cost are ordered "lexicographically", according to some order that is imposed on the elements of $\mathcal{A}''$.

At the beginning of phase 2 of Algorithm 2, QUEUE is initialized with the triplets that correspond to the singleton sets of the powerset $P(\mathcal{A}'')$. Subsequently, at each iteration of the conducted search process, Algorithm 2 extracts the head element of QUEUE and tests whether the corresponding set $\hat{\mathcal{A}}$ constitutes a feasible solution, as explained above. If $\hat{\mathcal{A}}$ is feasible, the algorithm will exit, returning the set $\hat{\mathcal{A}}$ as an optimal solution for the considered instance of the "optimal order restoration" problem. Otherwise, the corresponding digraph $\mathcal{G}(\hat{\mathcal{A}})$ contains a number of directed edges. Each of these edges, e, generates a candidate entry for QUEUE that is defined as follows: (i) the agent set $\hat{\mathcal{A}}(e)$ for this candidate entry is $\hat{\mathcal{A}}(e) = \hat{\mathcal{A}} \cup \{a(e)\}$, where $a(e)$ is the label of e in $\mathcal{G}(\hat{\mathcal{A}})$. (ii) $c(\hat{\mathcal{A}}(e)) = c(\hat{\mathcal{A}}) + c(a(e))$. (iii) The PDG $\hat{G}(\hat{\mathcal{A}}(e))$ is obtained from the digraph $\mathcal{G}(\hat{\mathcal{A}})$ by turning edge e into undirected.

The next theorem establishes the correctness of Algorithm 2; its proof can be found in the electronic supplement.

Theorem 2: When executed on an instance of the "optimal order restoration" problem of Definition 4, Algorithm 2 will terminate in finite time, and it will return a correct solution for this problem instance. □

Example: As an example on the algorithmic developments of this section, we consider the execution of Algorithm 2 on the traffic state $\tilde{s}$ that is defined by the PDG on the left side of Figure 4. As already discussed in Section III, the digraph $\mathcal{G}$ that will result from the execution of Algorithm 1 – or, equivalently, the first phase of Algorithm 2 – on this state, is that depicted on the right side of Figure 4. Hence, next we discuss the execution of the second phase of Algorithm 2, making use of the information that is provided in the digraph $\mathcal{G}$ of Figure 4. Furthermore, in this example we assume that $c(a) = 1, \forall a$; i.e., we try to restore order while minimizing the number of the agents that will be relocated to the "home" zone h by the derived solution.

In the considered case, QUEUE will be initialized with the entries $\big\langle\{a_i\}, 1.0, \hat{G}(a_i(e))\big\rangle$, for $i \in \{4, 6, 7, 8\}$. Furthermore, since all these entries are of equal cost, we shall assume that they are ordered lexicographically in QUEUE w.r.t. the agent index i.

Hence, the first entry to be picked for processing from this list is the entry $\big\langle\{a_4\}, 1.0, \hat{G}(a_4(e))\big\rangle$. The execution of Algorithm 1 on the corresponding PDG $\hat{G}(a_4(e))$ is depicted in Figure 5. As indicated in this figure, Algorithm 1 will reduce

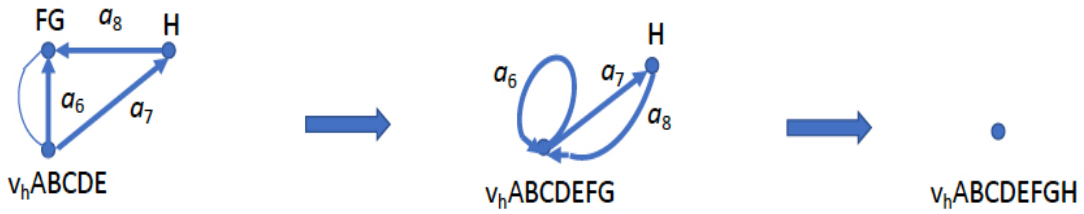

Fig. 5: The execution of Algorithm 1 on the PDG $\hat{G}(a_4(e))$ during the second phase of Algorithm 2, when Algorithm 2 is applied on the traffic state $\tilde{s}$ of Figure 4.

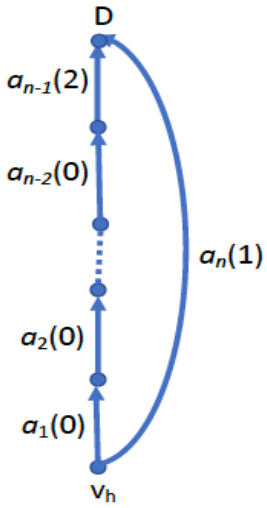

Fig. 6: A digraph $\mathcal{G}$ for which the execution of the second phase of Algorithm 2 will be of super-polynomial complexity w.r.t. its size.

the original PDG $\hat{G}(a_4(e))$ into a single vertex, and therefore, it can be deduced that the agent set $\{a_4\}$ constitutes an optimal solution for the instance of the "optimal order restoration" problem that is considered in this example. □

Complexity considerations: From the description of Algorithms 1 and 2, it is clear that the (a) the first phase of Algorithm 2 and also (b) the processing of any single entry that is extracted from QUEUE, can be performed efficiently w.r.t. the required computational time and memory. As a result, Algorithm 2 can exhibit fast execution times on most practical instantiations of the considered "optimal order restoration" problem.

On the other hand, since Algorithm 2 conducts a "best-first" search over the elements of the set $P(\mathcal{A}'')$ for an optimal solution, and the cardinality of this set is $O(2^{|\mathcal{A}''|})$, its worst-case computational complexity is super-polynomial w.r.t. the number of the system agents. A configuration of the digraph $\mathcal{G}$ for which Algorithm 2 will exhibit such a super-polynomial complexity, is depicted in Figure 6. The digraph $\mathcal{G}$ depicted in Figure 6 involves n traveling agents, $a_1, \ldots, a_n$, organized into two directed paths leading from vertex v_h to a terminal vertex D. The first path consists of the agent sequence $\langle a_1, \ldots, a_{n-1} \rangle$, while the second path involves only agent a_n. Furthermore, the relocation cost $c(a_i)$, for each agent a_i, $i = 1, \ldots, n$, is the number that is quoted in parentheses in the corresponding label. The reader can easily check that the optimal solution for this problem instance is the singleton $\{a_n\}$ with a corresponding optimal cost of 1.0. On the other hand, even though Algorithm 2 will enter the set $\{a_n\}$ in QUEUE, as a candidate solution, during the initialization of this list in phase 2, it will shift attention to this entry only after it has processed all the elements of the powerset of the agent set $\{a_1, a_2, \ldots, a_{n-2}\}$, since all these agent subsets have an associated cost of 0.0. Hence, in this case, Algorithm 2 will consider $2^{|\mathcal{A}''|-2}$ candidate solutions before it identifies the optimal one.

C. A streamlined version of Algorithm 2 for the case of uniform cost functions c.

As a potential "remedy" for the super-polynomial complexity that is demonstrated by the example problem instance of Figure 6, one can consider the possibility of synthesizing the sets $\hat{\mathcal{A}}$ that define the entries of QUEUE so that they contain at least one maximal directed path of the underlying digraph $\mathcal{G}$ that emanates from vertex v_h and has the in-degrees and the out-degrees of its internal vertices equal to 1.0. The rationale for such a restriction of the search process over the elements of the powerset $P(\mathcal{A}'')$ would be that the clearing of at least one of these paths from its occupying agents is a necessary condition for accessing the target vertex v_h by any remaining agents in digraph $\mathcal{G}$. This is certainly true in the example case of Figure 6, where the aforementioned maximal paths are the two paths $\langle a_1, \ldots, a_{n-1} \rangle$ and $\langle a_n \rangle$ leading from vertex v_h to the terminal vertex D. But Figure 7 provides a counter-example to the above conjecture.

Nevertheless, in the rest of this subsection we show that the above conjecture is true, and it will result in a more efficient version of Algorithm 2, in the particular case where the cost function $c(\cdot)$ is uniform over its domain set $\mathcal{A}'$.[5] The key result that underlies all these developments is established in the following proposition.

[5]Without loss of generality, we shall assume that $c(a) = 1.0, \ \forall a \in \mathcal{A}'$.

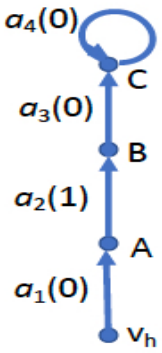

Fig. 7: A digraph $\mathcal{G}$ for which the optimal solution $\mathcal{A}^*$ for the corresponding "optimal order restoration" problem will not clear any maximal directed path that emanates from vertex v_h and possesses internal vertices with in-degree and out-degree equal to one. In this case, the capability of the "expensive" agent a_2 to reach vertex v_h is established by relocating to the "home" zone h the "cheaper" agent a_4.

Proposition 4: Consider an instance of the "optimal order restoration" problem with the corresponding cost function being $c(a) = 1.0,\ \forall a \in \mathcal{A}''$, and with the agent set $\mathcal{A}''$ being the labels of the edges of the digraph $\mathcal{G}$ that is obtained from the execution of Algorithm 1 on the corresponding PDG $\hat{G}(s)$. Then, there exists an optimal solution $\mathcal{A}^* \subset \mathcal{A}''$ for this problem instance that contains all the agents that are located on a maximal directed path of $\mathcal{G}$ that emanates from vertex v_h and has in-degree and out-degree equal to 1.0 for all of its internal vertices. □

The proof of Proposition 4 is provided in the electronic supplement. For those instances of the "optimal order restoration" problem that possess a uniform cost function $c(\cdot)$, Proposition 4 enables a much more focused and streamlined search process for an optimal solution $\mathcal{A}^*$; this search process is implemented by the pseudocode of Algorithm 3.

The overall structure of Algorithm 3 is similar to that of Algorithm 2. In particular, Algorithm 3 consists of two phases, with phase 1 consisting of the execution of Algorithm 1 on the input PDG $\hat{G}(s)$, and phase 2 conducting a "best-first" search over the powerset $P(\mathcal{A}'')$ that is defined from the result of phase 1. Furthermore, this search is conducted by means of a priority list QUEUE that is arranged in increasing cardinality of the agent subsets that constitute the stored candidate solutions.

The main difference of Algorithm 3 w.r.t. Algorithm 2 is the way that it generates the various entries to be stored in QUEUE. In particular, every time that the processing by Algorithm 1 of the PDG $\hat{G}(\hat{\mathcal{A}})$, that corresponds to a candidate solution set $\hat{\mathcal{A}}$, results in a digraph $\mathcal{G}(\hat{\mathcal{A}})$ that is not a single vertex, Algorithm 3 generates new candidate solutions for QUEUE by emptying from their occupying agents the maximal directed paths of the digraph $\mathcal{G}(\hat{\mathcal{A}})$ that emanate from vertex v_h and possess internal vertices with in-degrees and out-degrees equal to one. The same logic also applies to the initialization of QUEUE, based on the information that is provided by the digraph $\mathcal{G}$ that is obtained from phase 1.

Algorithm 3 A solution algorithm for instances of the "optimal order restoration" problem of Definition 4 with uniform cost function $c(a) = 1.0,\ \forall a \in \mathcal{A}'$.

Input: The corresponding PDG $\hat{G}(s)$.
Output: $ORDERED$: a Boolean variable indicating whether $s \in S_{ho}$; $\mathcal{A}^*$: an agent set defining an optimal solution for the considered "optimal order restoration" problem.

$ORDERED$:= Algorithm 1$\big(\hat{G}(s)\big)$;
if $ORDERED$ **then**
  $\mathcal{A}^*$:= $\emptyset$;
  return $ORDERED$ and $\mathcal{A}^*$.
end if
QUEUE := NIL;
for each maximal path p of the digraph $\mathcal{G}$ computed by Algorithm 1 that emanates from vertex v_h and has all its internal vertices possessing an in-degree and an out-degree equal to one **do**
  create the corresponding candidate solution $\langle \hat{\mathcal{A}}(p) \equiv \{a(e) : e \in p\}, |\hat{\mathcal{A}}(p)|, \hat{G}(\hat{\mathcal{A}}(p))\rangle$ and enter it in $QUEUE$ preserving the desired ordering of the queue entries;
end for
while $TRUE$ **do**
  $\big\langle \hat{\mathcal{A}}, |\hat{\mathcal{A}}|, \hat{G}(\hat{\mathcal{A}})\big\rangle$:= Extract head entry from $QUEUE$;
  $ORDERED$:= Algorithm 1$\big(\hat{G}(\hat{\mathcal{A}})\big)$;
  if $ORDERED$ **then**
    $\mathcal{A}^*$:= $\hat{\mathcal{A}}$;
    return $\neg ORDERED$ and $\mathcal{A}^*$.
  else
    for each maximal path p of the digraph $\mathcal{G}(\hat{\mathcal{A}})$ computed by Algorithm 1 that emanates from vertex v_h and has all its internal vertices possessing an in-degree and an out-degree equal to one **do**
      create the corresponding candidate solution $\big\langle \hat{\mathcal{A}} \cup \hat{\mathcal{A}}(p)\}, |\hat{\mathcal{A}}| + |\hat{\mathcal{A}}(p)|, \hat{G}(\hat{\mathcal{A}}(p))\big\rangle$ and enter it in $QUEUE$ preserving the desired ordering of the queue entries and avoiding duplication;
    end for
  end if
end while

The correctness of Algorithm 3, that results from the modifications of Algorithm 2 that were described in the previous two paragraphs, can be established through an argument similar to that used for establishing the correctness of Algorithm 2, while also considering the special structure of the optimal solution set for the considered problem instances that is characterized by Proposition 4. The next theorem is a formal statement of this result.

Theorem 3: When executed on an instance of the "optimal order restoration" problem of Definition 4 with uniform cost function $c(a) = 1.0,\ \forall a \in \mathcal{A}'$, Algorithm 3 will terminate in finite time, and it will return a correct solution for this problem instance.

Complexity considerations: The search that is conducted by Algorithm 3 for an optimal solution $\mathcal{A}^*$ over the underlying powerset $P(\mathcal{A}'')$ is much more efficient than the corresponding search that is conducted by Algorithm 2. This efficiency results from (i) the more focused generation of the candidate solutions $\hat{\mathcal{A}}$ by considering, in the corresponding digraphs $\mathcal{G}$, only the maximal directed paths that emanate from the vertex v_h and have in-degrees and out-degrees of their internal vertices equal to one, and (ii) the augmentation of the tested candidate sets $\hat{\mathcal{A}}$ that fail to provide a feasible solution, with an entire subset of agents that is defined by the aforementioned paths. Furthermore, the logic that drives the synthesis of the candidate solution sets $\hat{\mathcal{A}}$ under Algorithm 3, implies that entire subsets of $\mathcal{A}''$ will never be considered as parts of any candidate solution; in particular, it is easy to see that any agent a located in a strongly connected component of the digraph $\mathcal{G}$ that is generated by the first phase of Algorithm 3, will not be part of any set $\hat{\mathcal{A}}$ that is generated during the second phase of this algorithm.

A detailed example that demonstrates the execution of Algorithm 3, and concretizes the above remarks regarding the computational efficiency of the algorithm, can be found in the electronic supplement.

V. Conclusion

This paper has provided a thorough treatment for the two problems of (i) assessing whether a given traffic state of an open, irreversible, dynamically routed ZC-GBTS is h-ordered, and (ii) returning the state of such a ZC-GBTS into its subspace of h-ordered states, in a way that minimizes a measure of the disruption that is incurred in the operation of the underlying system. These two problems were first introduced in [1], and the current paper presents complete and computationally efficient algorithms for their solution.

The algorithmic developments and their supported analyses that were presented in this paper, rely heavily on the graphical structures that have been employed for the representation of the traffic state of the considered transport systems, and of the qualitative dynamics that will evolve this state in the context of the presumed operational policies. When viewed from this more methodological standpoint, the results of this paper also define a novel and powerful framework for reasoning about problems pertaining to traffic-state reachability and liveness assessment and enforcement in the considered transport systems, and parallel the recent developments of [19].

Another pertinent remark regarding the presented developments is that, while Algorithms 1 and 3 are making explicit use of the notion of the h-ordered state, the basic structure of Algorithm 2 is much more generic. In fact, Algorithm 2 can be used for restoring optimally additional properties of the traffic state s of the underlying transport system through the relocation of some agents in the underlying guidepath network. Hence, if we ever availed of an efficient algorithm for assessing the liveness of any given traffic state s, Algorithm 2 can be used for restoring optimally traffic-state liveness by invoking this new algorithm instead of Algotithm 1 during its execution.

Finally, potential future work on the developments of this paper can investigate (i) the worst-case computational complexity of the "optimal oder restoration" problem of Definition 4, and (ii) the pertinent structuring of the cost function $c(\cdot)$ that characterizes the disruption of the agent redirection to the "home" zone h, so that it captures a number of attributes regarding the current "progress" of the "mission" trips that are executed by the traveling agents and the "criticality" of these trips.

References

[1] S. Reveliotis, "An MPC scheme for traffic coordination in open and irreversible, zone-controlled, guidepath-based transport systems," *IEEE Trans. on Automation Science and Engineering*, vol. 17, pp. 1528–1542, 2020.

[2] S. S. Heragu, *Facilities Design (3rd ed.)*. CRC Press, 2008.

[3] T. Le Anh and M. B. M. De Koster, "A review of design and control of automated guided vehicle systems," Erasmus Research Institute of Management, Tech. Rep. ERS-2004-030-LIS, 2004.

[4] I. F. A. Vis, "Survey of research in the design and control of automated guided vehicle systems," *European Journal of Operational Research*, vol. 170, pp. 677–709, 2006.

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

[6] J. Yu and S. M. LaValle, "Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics," *IEEE Trans. on Robotics*, vol. 32, pp. 1163–1177, 2016.

[7] B. Kouvaritakis and M. Cannon, *Model Predictive Control: Classical, Robust and Stochastic*. London, UK: Springer, 2015.

[8] S. Reveliotis, "On the state liveness of some classes of guidepath-based transport systems and its computational complexity," *Automatica*, vol. 113, 2020.

[9] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems (2nd ed.)*. NY,NY: Springer, 2008.

[10] S. A. Reveliotis, "Conflict resolution in AGV systems," *IIE Trans.*, vol. 32(7), pp. 647–659, 2000.

[11] N. Wu and M. Zhou, "Resource-oriented Petri nets in deadlock avoidance of AGV systems," in *Proceedings of the ICRA'01*. IEEE, 2001, pp. 64–69.

[12] M. P. Fanti, "Event-based controller to avoid deadlock and collisions in zone-controlled AGVS," *Intl. Jrnl Prod. Res.*, vol. 40, pp. 1453–1478, 2002.

[13] E. Roszkowska, "Undirected colored Petri nets for modelling and supervisory control of AGV systems," in *WODES'02*. IEEE, 2002, pp. 135–142.

[14] ——, "Liveness enforcing in closed agv systems with dynamic routing," in *Proceedings of ICRA'04*. IEEE, 2004, pp. –.

[15] E. W. Dijkstra, "Cooperating sequential processes," Technological University, Eindhoven, Netherlands, Tech. Rep., 1965.

[16] 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*, vol. 10, pp. 73–100, 1998.

[17] J. Ezpeleta, F. Tricas, F. Garcia-Valles, and J. M. Colom, "A Banker's solution for deadlock avoidance in FMS with flexible routing and multi-resource states," *IEEE Trans. on R&A*, vol. 18, pp. 621–625, 2002.

[18] S. Reveliotis and T. Masopust, "Some new results on the state liveness of open guidepath-based traffic systems," in *27th Mediterranean Conference on Control and Automation (MED 2019)*. IEEE, 2019.

[19] ——, "Efficient liveness assessment for traffic states in open, irreversible, dynamically routed, zone-controlled guidepath-based transport systems," *IEEE Trans. on Automatic Control*, vol. 65, pp. 2883–2898, 2020.