An electronic supplement to the manuscript "Assessing and restoring "traffic-state order" in open, irreversible, dynamically routed, zone-controlled guidepath-based transport systems"

Spyros Reveliotis

***Abstract*—This document constitutes an "electronic supplement" to the manuscript entitled "Assessing and restoring "traffic-state order" in open, irreversible, dynamically routed, zone-controlled guidepath-based transport systems", that is authored by the same author, providing formal proofs for the technical results that are presented in that original document, and a supporting example for one of the developed algorithms.**

I. Proof of Proposition 2

It is clear from the construction of the state s' that is presented in the main document, that both states s and s' involve the same set of traveling agents, $\mathcal{A}$, but the guidepath networks that are implied by the corresponding PDGs $\hat{G}(s)$ and $\hat{G}(s')$, are different. Next, we show that any ordering $[\cdot] : \{1, \ldots, |\mathcal{A}|\} \rightarrow \mathcal{A}$ that satisfies the requirements of Definition 5 for state s is also a valid ordering with respect to the same requirements for state s', and vice versa. We establish this result by distinguishing the following two cases.

Case 1: In the PDG $\hat{G}(s)$, there is no maximal connected subgraph of undirected edges that contains a cycle. Suppose that the original traffic state s is h-ordered, and let o_1 denote an ordering of the agent set $\mathcal{A}$ that satisfies the requirements of Definition 5 with respect to this state. Under the working assumption of Case 1, each maximal connected subgraph Ξ_i of the PDG $\hat{G}(s)$ is a tree, and this tree is collapsed into a single vertex in the PDG $\hat{G}(s')$. Also, let σ be any event sequence that is defined in the FSA Φ_1 which models the traffic dynamics of the ZC-GBTS corresponding to the PDG $\hat{G}(s)$, and takes all agents to the "home" edge h while abiding to the ordering o_1; furthermore, the events in sequence σ are defined so that each of them corresponds to the advancement of a single agent a from its current zone to an adjacent free zone. Then, the event sequence σ also induces an event sequence σ' with the same properties but defined in the FSA Φ_2 that corresponds to the PDG $\hat{G}(s')$; sequence σ' is obtained from sequence σ by removing all these events that correspond to transitions into an undirected edge of the PDG $\hat{G}(s)$ by a traveling agent.

A similar argument can also establish that if the constructed traffic state s' is h-ordered, with a corresponding ordering o_2 for the agent set $\mathcal{A}$ and a sequence σ' that takes all agents $a \in \mathcal{A}$ to the "home" edge h of the underlying guidepath network, then, we can obtain a sequence σ with the same properties for the original traffic state s. More specifically, sequence σ is obtained from sequence σ' by inserting in σ' event subsequences corresponding to traversals of paths of undirected edges for any transition in σ' concerning a pair of edges that are not adjacent in the PDG $\hat{G}(s)$. Furthermore, the tree structure of the maximal subgraphs Ξ_i of the PDG $\hat{G}(s)$ in the considered case implies that the inserted paths will be uniquely defined.

Case 2: In the PDG $\hat{G}(s)$, there is some maximal connected subgraph of undirected edges that contains a cycle. In this case, the presence of a cycle $\mathcal{C}$ of undirected edges in the PDG $\hat{G}(s)$ enables any agent a that is located on a directed edge $\big(v_1(a), v_2(a)\big)$ connected to cycle $\mathcal{C}$ by a path of undirected edges, to revert the direction of its motion on its current edge. This possibility would not be recognized if the maximal connected subgraphs of undirected edges, $\Xi_1, \ldots, \Xi_l$, of the PDG $\hat{G}(s)$ were simply reduced to single nodes. In the construction procedure of Figure 2, this issue is addressed through the introduction of the new undirected edges e' in the PDG $\hat{G}(s)$, according to the corresponding logic that is detailed in Step 2 of that procedure. The addition of the edges e' before the compression of the subgraphs $\Xi_1, \ldots, \Xi_l$ into single nodes has the effect of converting the aforementioned edges $\big(v_1(a), v_2(a)\big)$ into self-loops connected to the rest of the obtained PDG $\hat{G}''$ at the corresponding node that represents, both, $v_1(a)$ and $v_2(a)$. Furthermore, it is also possible that the modification of the original PDG $\hat{G}(s)$ through the addition of the edges e' in Step 2 of the construction procedure of Figure 2, and the subsequent reduction of the resulting PDG $\hat{G}''$ by means of Step 3 of the same procedure, will render more explicit the accessibility of certain parts of the underlying guidepath network by some traveling agents that is not as obvious from the structure of the original PDG $\hat{G}(s)$; this possibility is demonstrated very vividly in the example construction of Figure 3 in the main manuscript.

Using the insights that are provided in the previous paragraph, we can proceed to establish the validity of Proposition 2 for Case 2 through an argument that is very similar to the argument that established the validity of this proposition in Case 1; the details are quite straightforward and they are left to the reader.

Finally, since Cases 1 and 2 cover exhaustively all the different possibilities regarding the structure of the maximal subgraphs Ξ_i of the PDG $\hat{G}(s)$, the proof of Proposition 2 has been completed.

S. Reveliotis is with the School of Industrial & Systems Engineering, Georgia Institute of Technology; email: {spyros@isye}.gatech.edu.
This work has been partially supported by NSF grant ECCS-1707695.

II. Complexity Analysis of Algorithm 1

Steps 1 and 2 of the construction procedure of Figure 2 can be implemented by a simple exploratory (also known as "reaching") algorithm that identifies the targeted subgraphs $\Xi_i,\ i = 1, \ldots, l$, of the underlying PDG $\hat{G}(s)$, one at a time. This algorithm will start a new maximal subgraph Ξ_i by inserting in it some undirected edge e of $\hat{G}(s)$ that is currently "unexplored", and it will also include in Ξ_i all those additional undirected edges of $\hat{G}(s)$ that can be reached from edge e through a path of undirected edges. Furthermore, if some vertex v of the identified subgraph Ξ_i is reached through more than one paths during the aforementioned exploration, it can be inferred that the subgraph Ξ_i also contains a cycle. Hence, Steps 1 and 2 of the construction procedure of Figure 2 can be executed in time $O(|E|)$. Furthermore, the complexity of Step 3 in this procedure is $O(|E|)$, as well. Finally, the second part of Algorithm 1 – i.e., the part that executes on the digraph $\hat{G}(s')$ – has complexity $O(|\mathcal{A}'|)$, where $\mathcal{A}'$ denotes the set of agents a with $\gamma(a; s') \neq h$. Since $|\mathcal{A}'| \leq |E|$, the complexity of the entire Algorithm 1 is $O(|E|)$.

III. Proof of Theorem 2

We begin by noticing that under the presumed structure of the guidepath network G of the underlying transport system, every traffic state s containing only one agent with $\gamma(a; s) \neq h$ is h-ordered. Furthermore, Proposition 2 implies that this property is preserved for the traffic states s' that are constructed during the execution of Algorithms 1 and 2. Hence, any subset $\hat{\mathcal{A}}$ of $\mathcal{A}''$ with cardinality $|\hat{\mathcal{A}}| = |\mathcal{A}''| - 1$ will be a feasible solution for the considered problem.

Next, consider a sequence of candidate entries $\epsilon_1, \epsilon_2, \epsilon_3, \ldots$ for QUEUE that is generated as follows: Entry ϵ_1 is one of the candidate solutions that enter QUEUE during the initialization of this queue in phase 2 of Algorithm 2. Entry ϵ_2 is an entry that is spawned from the processing of entry ϵ_1; entry ϵ_3 is an entry that is spawned from the processing of entry ϵ_2; etc. Then, the aforestated logic that generates each entry $e_i,\ i = 2, 3, \ldots$, from entry e_{i-1}, together with the opening remarks of this proof regarding the feasibility of all candidate solutions with cardinality $|\mathcal{A}''| - 1$, implies that there is an entry e_k with $k \leq |\mathcal{A}''| - 1$ in the considered sequence that is a feasible solution for the considered problem. This entry will enter QUEUE unless its generation is preempted by the generation and the processing of another feasible solution of smaller cost. Hence, Algorithm 2 will terminate in a finite number of steps.

The fact that the returned solution is indeed a feasible solution to the considered problem, is implied by the correctness of Algorithm 1 that is used for the testing of the various candidate solutions. Furthermore, the returned solution will also be optimal due to the initialization of QUEUE by the singleton sets of $\mathcal{A}''$, and the cost-based priority logic that is used in the maintenance of this queue.

IV. Proof of Proposition 4

Let $\hat{\mathcal{A}} \subset \mathcal{A}''$ be any feasible solution to the considered problem instance. Also, consider the PDG $\hat{G}(\hat{\mathcal{A}})$ that is obtained from the digraph $\mathcal{G}$ by converting the directed edges of $\mathcal{G}$ corresponding to the agents $a \in \hat{\mathcal{A}}$ into undirected. Then, according to the opening remarks of Section IV.C in the main document, the PDG $\hat{G}(\hat{\mathcal{A}})$ must belong to least one of the following two cases:

Case 1: One of the maximal directed paths of the digraph $\mathcal{G}$ that emanate from vertex v_h and have the in-degrees and out-degrees of their internal vertices equal to one, has been converted into a path of undirected edges in the PDG $\hat{G}(\hat{\mathcal{A}})$. In this case, it is clear that the considered feasible solution $\hat{\mathcal{A}}$ satisfies the condition of Proposition 4.

Case 2: There is a maximal directed path of the digraph $\mathcal{G}$ that emanates from vertex v_h and has the in-degrees and out-degrees of their internal vertices equal to one, this path has only one directed edge $\tilde{e}$ in the PDG $\hat{G}(\hat{\mathcal{A}})$, and furthermore, in the PDG $\hat{G}(\hat{\mathcal{A}})$ there is also a path of undirected edges leading from edge $\tilde{e}$ to a cycle $\mathcal{C}$ of undirected edges; hence, the agent $\tilde{a}$ located on the edge $\tilde{e}$ can use the cycle $\mathcal{C}$ of undirected edges in order to revert the direction of its motion on the considered path and reach vertex v_h, clearing, thus, this path for the remaining agents in the PDG $\hat{G}(\hat{\mathcal{A}})$ (this is the case depicted in the example of Figure 7 in the main document).

In this case, consider the agent set $\tilde{\mathcal{A}}$ that is induced from the considered feasible solution $\hat{\mathcal{A}}$ by adding agent $\tilde{a}$ into set $\hat{\mathcal{A}}$ and removing from this set the agent $\hat{a}$ that occupies one of the edges, say $\hat{e}$, of the aforementioned cycle $\mathcal{C}$ in the digraph $\mathcal{G}$. Clearly, $|\tilde{\mathcal{A}}| = |\hat{\mathcal{A}}|$. Furthermore, the agent set $\tilde{\mathcal{A}}$ defines another feasible solution for the considered instance of the "optimal order restoration" problem, since, (i) in the PDG $\hat{G}(\tilde{A})$, the re-inserted agent $\hat{a}$ can use the path of undirected edges that established the addition of the agent $\tilde{a}$ in the set $\hat{\mathcal{A}}$ in order to reach vertex v_h, and (ii) the traffic state that results from this advancement of agent $\hat{a}$ is exactly the same traffic state that will result from the aforementioned advancement of agent $\tilde{a}$ to vertex v_h in the PDG $\hat{G}(\hat{\mathcal{A}})$. Finally, it is clear that the new feasible solution $\tilde{\mathcal{A}}$ satisfies the condition of Proposition 4.

Collectively, the above analyses of Cases 1 and 2 imply that for any feasible solution $\hat{\mathcal{A}}$ for the considered instances of the "optimal order restoration" problem, there exists another feasible solution $\tilde{\mathcal{A}}$ that satisfies the condition of Proposition 4 and has the same cost with $\hat{\mathcal{A}}$. But then, the validity of Proposition 4 results immediately from this last remark.

TABLE I: The evolution of the content of QUEUE in the example of Section V.

Iter.	QUEUE
0	$\langle a_6a_7a_8, a_1a_2, a_5\rangle$
1	$\langle \mathbf{a_5a_6a_7a_8}, a_6a_7a_8, \mathbf{a_3a_5}, a_1a_2\rangle$
2	$\langle \mathbf{a_1a_2a_6a_7a_8}, a_5a_6a_7a_8, a_6a_7a_8, \mathbf{a_1a_2a_5}, \mathbf{a_1a_2a_4}, \mathbf{a_1a_2a_3}, a_3a_5\rangle$

V. An example on the execution of Algorithm 3

In this example, we consider the execution of (the second phase of) Algorithm 3 on the digraph $\mathcal{G}$ that is depicted in the left side of Figure 1 of this document. The corresponding evolution of QUEUE is presented in Table I, while the digraphs $\mathcal{G}(\hat{\mathcal{A}})$ that are generated by the invocations of Algorithm 3 during the processing of the various entries that are extracted from QUEUE, are also depicted in Figure 1.

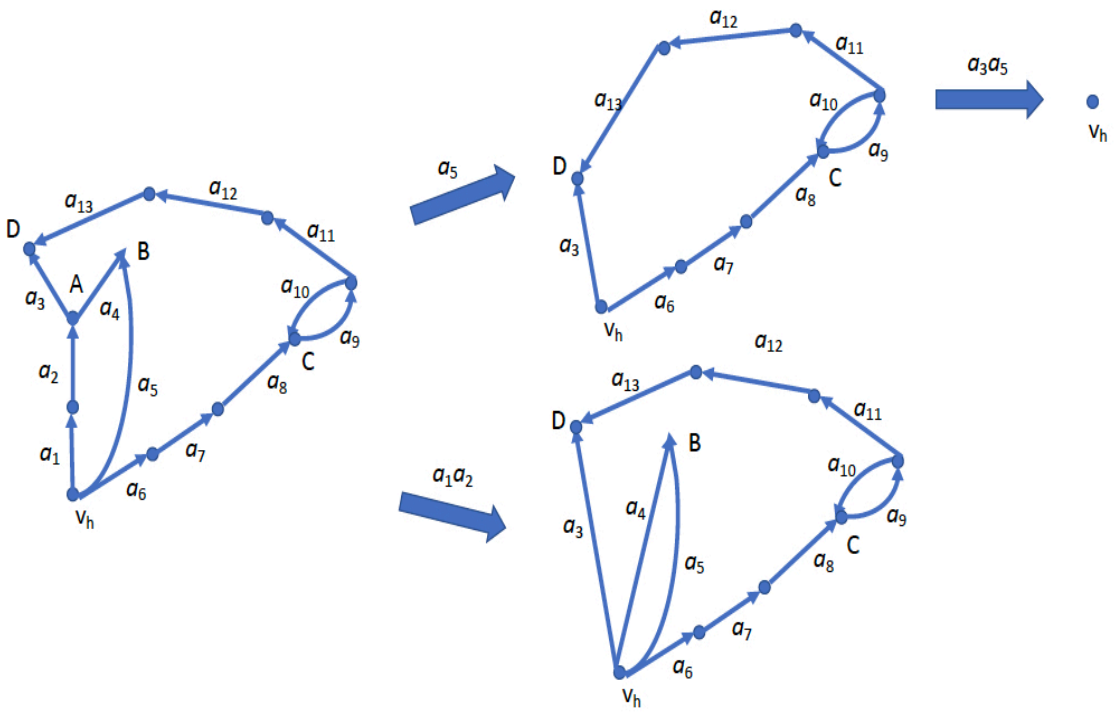

Fig. 1: The digraphs that are generated by Algorithm 1 during the execution of Algorithm 3 on the example problem instance of Section V.

More specifically, Table I reports the first element of the entries stored in QUEUE upon its initialization, and upon the completion of every iteration of Algorithm 3 that takes place during its second phase, except for the last one. Also, in order to attain a more compact representation, in Table I we do not present these first elements of the various QUEUE entries as sets containing the corresponding agents, but as strings of these agents. At every iteration $i = 1, 2, \ldots$, the head element in the QUEUE list that is reported in Table I for iteration $i-1$, is removed for processing, and the new entries for this list that are generated during this iteration, are indicated in bold fonts in the corresponding list of Table I for iteration i.

QUEUE is initialized with the agent subsets occupying each of the three directed paths in the digraph $\mathcal{G}$ that is depicted in the left side of Figure 1, which lead from vertex v_h to one of the three vertices A, B and C of $\mathcal{G}$; each of these three vertices is the first encountered vertex to violate the requirement of having, both, its in-degree and out-degree equal to one, for the corresponding path.

During the first iteration of the "while" loop in Algorithm 3, agent a_5, which is the head entry of QUEUE after its initialization, will be relocated to the "home" zone h. Furthermore, this relocation of agent a_5 will enable agents a_4, a_2 and a_1 to reach vertex v_h, as well (in this particular order). Hence, the execution of Algorithm 1 during this first iteration will return the digraph that is depicted in the top-middle part of Figure 1, and this result will generate the new entries a_3a_5 and $a_5a_6a_7a_8$ for QUEUE; these two entries are defined by (i) the currently processed entry of a_5 and (ii) the two directed paths leading, respectively, from vertex v_h to the vertices C and D in the aforementioned digraph. The resulting state for QUEUE upon the completion of this first iteration is reported in the second row of Table I; the current head element of QUEUE is the agent subset $\{a_1, a_2\}$.

The processing of the current head element of QUEUE during the second iteration of Algorithm 3 will result in the digraph that is depicted in the lower-middle part of Figure 1 and the QUEUE state that is depicted in the third row of Table I.

The head element of QUEUE upon the completion of iteration #2 is the agent subset $\{a_3, a_5\}$. The processing of this queue entry by Algorithm 3 will turn the directed edge (v_h, D) in the digraph at the top-middle part of Figure 1 into undirected. Furthermore, it can be easily checked that, under this conversion, the remaining agents that are present in this digraph can reach vertex v_h, in the sequence $a_{13}, a_{12}, a_{11}, a_9, a_{10}, a_8, a_7, a_6$. Hence, the result of the execution of Algorithm 1 during this iteration will be the single vertex v_h, and Algorithm 3 will return the agent set $\{a_3, a_5\}$ as an optimal solution for the considered instance of the "optimal order restoration" problem.

Concluding this example, it is also interesting to notice that Algorithm 3 was able to identify an optimal solution for the corresponding problem instance in just two iterations, while Algorithm 2, of Section IV.B of the main manuscript, would have to explore at least all the singletons $\{a_i\}$, $i = 1, \ldots, 13$, before getting to this solution.