Optimized Multi-Agent Routing for a Class of Guidepath-based Transport Systems

Greyson Daugherty, Spyros Reveliotis and Greg Mohler

***Abstract*—This paper presents a heuristic algorithm for minimizing the makespan required to route a set of agents inhabiting a shared guidepath network from their initial locations to their respective destinations, while observing a set of regulations that seek to ensure the safety and the integrity of the generated traffic. From an application standpoint, the presented developments are motivated by the traffic coordination challenges that arise in the context of many automated unit-load material handling systems, and also in the transport of the ionized atoms that takes place in the context of quantum computing. From a methodological standpoint, our developments constitute a customization of the general "local-search" framework of combinatorial optimization theory to the traffic management problem that is considered in this work. Hence, the presented results include a rigorous characterization of the considered problem and its solution space, detailed algorithms for the construction of the necessary initial solutions and the improving step for the pursued search, a complexity analysis of these algorithms, and a set of computational experiments that reveal and assess the computational efficiency of the presented algorithms and the efficacy of the derived solutions. The paper concludes with some suggestions for potential extensions of the presented results.**

***Note to Practitioners* – In many contemporary applications of automation science and engineering, a number of entities – or "agents" – must be transported expediently from their initial locations to certain destinations using a set of links that define the underlying "guidepath network". Furthermore, various safety considerations require that the agents must be adequately separated during these transports, and the imposed restrictions turn the corresponding traffic coordination problem into a complex resource allocation problem where the contested resources are the guidepath-network links. This paper presents a set of algorithms that can provide high-quality schedules for the resulting traffic-scheduling problems in a computationally efficient manner. These properties of our algorithms are established through the necessary theoretical analysis, but they are also demonstrated through a series of numerical experiments where they are shown capable to provide near-optimal solutions for some very complex problem instances in no more than a few seconds. In addition, our algorithms are "complete", i.e., they will always provide a feasible schedule for any instantiation of the traffic-scheduling problem considered in this work. Hence, they can effectively address the needs for "real-time" traffic management that arise in the context of the considered applications.**

***Index Terms*—Guidepath-based transport systems, multi-agent routing, multi-robot path planning, combinatorial scheduling, congestion control, conflict management.**

G. Daugherty and S. Reveliotis are with the School of Industrial & Systems Engineering, Georgia Institute of Technology; emails: `sdaugherty3@gatech.edu`, `spyros@isye.gatech.edu`.
G. Mohler is with the Georgia Tech Research Institute; email: `Greg.Mohler@gtri.gatech.edu`.
This work was partially supported by NSF grants ECCS-1405156 and ECCS-1707695, and by an IRAD program of the Georgia Tech Research Institute.

I. Introduction

In this work, we consider a class of traffic scheduling problems that concern the circulation of a set of "agents" on the edges of a connected graph, which is known as the "(supporting) guidepath network". The edges of the guidepath network define "zones" for the supported traffic that can be occupied by at most one agent at a time. Access to the guidepath zones is granted by a central controller, and in this way, the trip of any given agent from an "origin" location to a "destination" location becomes a "resource allocation process" where the necessary zones that support the various legs of this trip, are requested and acquired one zone at a time. Additional constraints regulate the agent transitioning between zones, and the behavior of the agents upon reaching their destinations. Our main concern is to enable all traveling agents to reach their destinations in a way that (i) respects the imposed regulations, and (ii) minimizes the "(time) makespan" of the corresponding traffic schedule.[1]

From a practical standpoint, the aforementioned problem arises in a broad spectrum of applications, that includes various robotics [1] and automated industrial material-handling [2] applications, the design and the analysis of various classical games [3] and animated computer-game [4] applications, and the efficient processing of the ionized atoms that are the elementary information carriers in the context of quantum computing [5]. Each of these application contexts gives rise to different detailed formulations of the considered traffic scheduling problem, with the corresponding formulations being differentiated in terms of, both, (i) the zone-allocation constraints that define the dynamics of the generated traffic, and (ii) the performance objective that characterizes the notions of the "traffic expediency" and "efficiency". Furthermore, as it happens with many other classes of combinatorial optimization / scheduling problems, each of these formulations can vary substantially in terms of the computational complexity of the corresponding optimization problem, but also in terms of the feasibility of its various instantiations and the complexity of the construction of a feasible – or "satisficing" – solution for it.

The particular problem version that is considered in this work arises in the operational context of a very broad class of automated material handling systems (MHS), known as "unit-load, zone-controlled" MHS [2], and also in the aforementioned operational context of quantum computing. More specifically, in many industrial settings, the necessary material

[1]This high-level description of the considered traffic systems becomes much more detailed in the subsequent, more technical parts of this work.

handling operations are supported by a fleet of autonomous agents that transport material among a set of locations while moving on a set of very detailed pathways that link these locations and define the corresponding "guidepath network". Particular instantiations of this basic description include the automated guided vehicle (AGV) systems that are used in many production and distribution environments [2], the overhead monorail systems that have been the typical MHS in semiconductor manufacturing [6], [7], and the complex gantry crane systems that are used in some large ports and railway yards [2]. In all these environments, in order to avoid collisions among the system agents, the links of the guidepath network are split up into "zones", and it is required that, at any time point, each zone is occupied by at most one vehicle. This requirement is enforced by a traffic supervisor that monitors the zone occupancy by the traveling vehicles and grants accessibility to these zones. Additional safety concerns and other practical considerations define further conditions that must be satisfied by an admissible transition of an agent from its current zone to a neighboring one on the guidepath network. A typical such condition is that an agent can move from its current zone to a neighboring one only if the requested zone is currently unoccupied; this requirement (i) ensures the agent separation in the face of the uncertainty that is introduced by the asynchronous nature of the agent transitions between their consecutive zones, and (ii) it also prevents zone "swapping" by agents that occupy neighboring zones, since in the context of the considered environments such zone swaps are not physically possible.

In the aforementioned settings, the agent traveling can be perceived as a set of "mission trips" corresponding to a transport operation between an origin and a destination node of the guidepath network. The transport requirements arise in a dynamic manner, and they are assigned to the system agents according to certain logic that will be taken as pre-specified and outside the scope of the considered work. Furthermore, the system configuration possesses a "resting area" where agents that have completed their previously assigned missions are retired and potentially recharge their batteries. Hence, in this operational setting, the task of the traffic controller is to enable each agent to complete its current mission in an expedient manner, and reach successfully the resting area.[2] In particular, under an arbitrary topology for the underlying guidepath network, and certain further assumptions about the agent maneuverability while traversing the various zones of this network, the aforementioned restrictions on the zone allocation process can also give rise to "deadlock" formations where a subset of agents blocks each other's advancement towards the completion of their mission trips in a permanent manner [8]; in these cases, the traffic controller must also effectively predict and prevent such potential deadlocks.

The traffic coordination problem for the unit-load, zone-controlled MHS that was described in the previous paragraphs, arises essentially identical in the physical operations that take place in the context of quantum computing. In this case, the traveling agents are the ionized atoms that hold the elementary information that is processed in those computational platforms, and are known as "qubits" [5]. These qubits must have their quantum state altered in a controlled manner by bringing them in certain locations where they will interact with certain local fields, and possibly with each other. The qubit circulation among these locations is facilitated by a "maze" of "ion traps" that contain physically the qubits and isolate them from their surrounding environment and from each other. Hence, the ion traps play the role of the zones in the MHS that were described in the previous paragraphs, and their allocation to the traveling qubits obeys restrictions and concerns similar to those that arise in those earlier cases. Furthermore, the role of the resting area is played in this case by the physical medium implementing the "memory" that holds qubits that are not currently participating in the running computation, while the mission trips for the traveling qubits are defined by the underlying program code and its decomposition to the elementary computational operations that are recognized by quantum computing. Finally, in this new application setting, traveling agents might also need to satisfy "rendezvous" requirements at certain locations, a fact that raises additional synchronization concerns and implicit precedence constraints for the underlying "zone allocation" problem.

It should be clear from all the above discussion that the traffic coordination problem arising in the context of the described operational settings is pretty complex. Indeed, the agent contest for the zones of the guidepath network and the corresponding sequential resource allocation process possess all the operational characteristics of the notorious "job shop scheduling" problem [9]. At the same time, this new problem version is further complicated by (i) the extensive routing flexibility that is defined by the underlying guidepath network, (ii) the new spatio-temporal constraints that must be observed by the zone allocation process, (iii) the potential "rendezvous" requirements among the traveling agents, and (iv) the dynamic specification of the various mission trips. In view of all this complexity and the dynamic evolution of the underlying requirements, we propose to address the resulting traffic scheduling problem through a "rolling horizon" scheme that decomposes the overall problem into a number of subproblems seeking to transport all the traveling agents to their immediate destinations as fast as possible. More specifically, these subproblems will be specified each time that an agent reaches its current destination or it is assigned to a new mission trip, and they will seek to move all the traveling agents from their current locations to their next immediate destination in their mission trips, while minimizing the time that is necessary for all these transports; this time is known as the "makespan" of the corresponding traffic schedule in the relevant terminology [9]. Additional care must be taken to ensure that the specification of the intermediate destinations of the traveling agents, and the resulting problem decomposition, will guarantee the "liveness" of the generated traffic, i.e., the ability of the traveling agents to complete successfully their mission trips and retire to the provided resting area while avoiding potential deadlocks. Finally, it is also important to notice that the subproblems defined by the outlined rolling-horizon scheme remain pretty hard, as they maintain all four features of the overall traffic scheduling problem that were

[2]In fact, agents that have completed their mission and are heading to the resting area, can also be re-assigned to a new mission before actually returning to the resting area.

enumerated at the beginning of this paragraph.

In an effort to provide a complete characterization of the subproblems that arise in the context of the aforementioned rolling-horizon scheme, in [10], [11] we presented a mixed integer programming (MIP) formulation for these subproblems, and we also formulated and investigated a Lagrangian "dual" problem for that formulation. This "dual" problem can provide some good lower bounds for the performance of any optimal traffic schedule, but the computation of such an optimal traffic schedule through the solution of the MIP formulation itself will not be tractable for many practical problem instances. Furthermore, the MIP nature of this formulation implies that the optimal solutions of the "dual" problem will not specify any solutions for the original MIP formulation [12], [13].[3] Hence, there is a remaining need for some methodology that will provide efficient, near-optimal solutions for the traffic scheduling problem that was formulated in [10], [11]. This need is addressed in this paper.

More specifically, in this paper we provide a heuristic algorithm for the traffic scheduling problem that constitutes the core subproblem of the aforementioned rolling-horizon framework, by adapting to this problem ideas and techniques borrowed from the broader area of combinatorial optimization [14]. In more technical terms, the presented algorithm can be perceived as a "local-search" scheme that starts with the construction of a feasible routing schedule, and subsequently it searches for improved solutions over pertinently defined "neighborhoods" of the underlying solution space. It is well known that the effective implementation of such a local-search scheme depends significantly upon the employed representations of the underlying solution space and the imposed "neighborhood" structures. We provide these representations as well as the procedures that will effect the search for improved solutions.

Furthermore, in the context of the considered traffic scheduling problems, the construction of an initial feasible traffic schedule can be a challenging task in itself. In fact, for many instantiations of these problems, the decision problem of assessing the existence of a feasible routing schedule is NP-complete [15]. However, the particular class of guidepath-based traffic systems that is the primary focus of this work is defined by a set of operational assumptions regarding the maneuverability of the traveling agents that (i) ensure the feasibility of all the instantiations of the considered traffic scheduling problem, and (ii) enable the design of effective and efficient computational algorithms for the construction of the necessary initial solutions. When viewed in the context of the aforementioned rolling-horizon framework, these properties further imply the liveness of the generated traffic, and therefore, the provided solution is complete in that sense, as well. Moreover, in the closing discussion of the paper, we also outline some ideas and guidelines for extending the presented methodological framework to guidepath-based traffic systems that might not satisfy the complete set of the operational assumptions that are considered in this work.

The aforementioned theoretical developments are complemented by a series of computational experiments. The corresponding results reveal that (i) the traffic schedules generated by the presented algorithm are very efficient in terms of the specified objective of minimizing the corresponding makespan, and (ii) they can be obtained very fast. Both of these properties are very important for the practical applicability of the proposed algorithm in the aforementioned application contexts. Furthermore, additional discussion provided in the last part of the paper suggests various modifications for the presented algorithm that can further enhance the quality of the derived solutions, and facilitate a more explicit trade-off between (a) the representational and the computational complexity of this algorithm, and (b) the operational efficiency of the generated traffic schedules.

In view of all the above discussion, the rest of the paper is organized as follows: Section II provides a systematic review of the current literature on the control of guidepath-based traffic systems, and positions the developments presented in this work in the context of that literature. Section III presents a formal characterization of the considered traffic system, the generated traffic dynamics, and the particular traffic scheduling problem addressed in this work. Subsequently, Section IV presents the key results of the paper, i.e., the proposed algorithm, together with a formal analysis of its correctness and its computational complexity. Section V reports the computational results that demonstrate and assess the aforementioned efficacies of the presented algorithm. Section VI discusses potential extensions of the results that are presented in Section IV, in an effort to expand the applicability of these results and to further control the trade-off between the complexity of the presented algorithm and the efficiency of the derived solutions. Finally, Section VII concludes the paper. We also notice, for completeness, that an abridged version of this material was presented at the IFAC World Congress 2017.[4]

II. Literature review

Because of their very broad applicability that was discussed in the previous section, guidepath-based transport systems have drawn attention in a number of scientific and engineering communities. In fact, some of the first studies regarding the dynamics of this class of traffic systems have been performed by the computer science (CS) community, in the context of the, so-called, 15-puzzle. The objective of this puzzle is to re-organize 15 labeled pieces, that are positioned on a 4×4 grid, in a row-major arrangement, by using the single empty slot of the grid. A first feasibility study of this problem was presented in [3], which studied a generalized version of the problem that involved $n-1$ labelled pebbles located on the vertices of an n-vertex 2-connected graph G. Using permutation group theory, [3] established that, for non-bipartite graphs G, the original pebble configuration can be re-arranged to any target configuration, while in the case of bipartite graphs, the corresponding reachability requirement divides all possible pebble configurations into two equivalence classes.

[3]In general, the solution of the Lagrangian "dual" problem might not provide even a feasible solution for the original MIP formulation.

[4]That manuscript was developed in a more casual style, and it lacks (i) the detailed positioning of the work and the expansive literature review that are provided in the current manuscript, (ii) the more formal arguments of Section IV, (iii) the more extensive computational experiments that are reported in Section V, and (iv) the second and the third parts of the discussion that is provided in Section VI.

This feasibility study was subsequently extended in [16], which addressed a more general problem version involving (i) more general connectivity conditions for the underlying graph G, and (ii) pebble distributions with more than one empty vertices. This last paper also provided an $O(n^3)$ algorithm that either returns a feasible solution for the considered problem instance, in the form of a pebble move-sequence, or determines its infeasibility.

More recently, guidepath-based transport systems similar to those studied in [3], [16] have been revisited by the artificial intelligence (AI) and the robotics communities under the theme of "multirobot path planning (MPP)". These studies have introduced additional assumptions regarding (i) the topologies of the supporting graph G, (ii) the allowed moves that transform the pebble allocation (or, in this case, the robot positioning) in this graph, and (iii) they have have also converted the original feasibility studies to optimization problems that seek to optimize the generated move-sequence in terms of some of its attributes. Some interesting works along the aforementioned lines (i) and (ii) are those presented in [17], [18], [19], [20], [21], [22]. More specifically, when viewed from a collective standpoint, these works have specialized the original results of [3], [16] for tree topologies of the underlying graph G, and they have also considered variations of the original problem versions that allow for robot substitutability in the specification of the target configuration, synchronized robot moves that allow their repositioning and advancement towards their destinations even in totally congested graphs, and robot collaboration for the transport of their assigned payloads to the corresponding destinations. These studies have also provided feasibility conditions for the corresponding problem instances, rigorous complexity analyses of the decision problems that are defined by these feasibility tests, and also complexity analyses of the various optimization formulations that are defined for all the above problem versions. Perhaps not surprisingly, most of these optimization problems turn out to be NP-hard [23].

The negative complexity results for the aforementioned optimization problems have subsequently determined the algorithmic approaches that have been pursued for their solution. In the context of the MPP literature, these approaches are broadly classified into "coupled" and "decoupled" methods [24]. Decoupled methods seek to control the underlying problem complexity by decomposing the overall path-planning problem across the different agents, and addressing the resulting subproblems sequentially. Path plans that are generated earlier in this sequence define constraints to be observed by the remaining subproblems. However, as observed in [4], [25], [24], such a decomposition scheme is not guaranteed to generate a feasible traffic schedule, even if such a schedule exists, and the algorithm's ability to generate a feasible solution, as well as the quality of this solution, will generally depend on the particular sequence that was adopted for the solution of the single-agent path-planning problems. In many cases, these issues can be (partially) addressed through the iterative execution of the employed algorithm with pertinently modified sequences for the solution of the single-agent problems, and/or a localized perturbation of the generated agent plans at their segments with experienced conflicts [4]. In some other works of the MPP literature, the aforementioned limitations of the decoupled methods are partially addressed through the specification of the entire traffic schedule as a concatenation of "single-agent [motion] primitives" that seek to attain certain positions for judiciously selected agents while possibly incurring the relocation of the remaining agents in this process. The effectiveness and the completeness of the resulting algorithms usually depends on the presumed topology of the underlying guidepath network. Also, the typical objective of these algorithms is only the synthesis of a satisficing solution. Some of the most sophisticated examples of this last line of work are presented in [24], [26].

Coupled methods of the MPP literature seek to provide (near-)optimal solutions to the corresponding scheduling problems by taking a more holistic view of these problems and their solution spaces. These methods essentially represent the dynamics of the underlying traffic as a finite state automaton (FSA) [27] where (i) the system states are defined by the various distributions of the traveling robots to the vertices (or, in some other cases, the edges) of the guidepath network, and (ii) the transitions of the automaton among its various states are defined by the various "elementary moves" that are allowable in the considered operational context. Under such a representation, the resulting optimization problem can be expressed either as a (mixed) integer program (MIP) [21], or as a dynamic programming (DP) problem [25], or even as a "satisfiability (SAT)" problem [28]. The resulting methods are complete, i.e., they can provide, in principle, an optimal solution to the considered traffic scheduling problem, if such a solution exists. But their applicability is severely limited by the state space explosion of the underlying FSA and/or the NP-hardness of the employed formulations.[5]

On the other hand, some works, like those of [25], [29], [30], [31], have tried to develop in the middle ground between the coupled and decoupled methods, coming up with what has been characterized as a "hybrid" method. This alternative method uses a complete algorithm (frequently an adaptation of the A^* algorithm [32] to the considered problems) in order to determine optimized routing plans for each single agent, and progressively it couples these agents into larger groups every time that the original decoupled approach fails to generate a complete solution for the underlying subgroups. The aforementioned papers also propose a number of additional

[5]To circumvent these computational limitations, the coupled methods that were mentioned in the previous paragraph are typically embedded in a decomposing scheme that seeks to mitigate the computational complexity of the eventually solved formulations. Hence, when it comes to the developments of [28] and [21], it is proposed to cope with the potential intractability of the respective SAT and MIP formulations by specifying a sequence of intermediate destinations for each traveling agent, and employing a sequence of simpler SAT and/or MIP-based formulations that will seek to route the traveling agents to the various configurations that are defined by these intermediate destinations. By employing a dense set of intermediate destinations and keeping these destinations sufficiently apart from each other, it is possible to establish a "locality" for the intermediate agent trips that decouples the corresponding routing problems, simplifies substantially the empirical complexity of the involved SAT and MIP formulations in terms of the numbers of variables and constraints involved, and reduces the experienced solution times. But these simplifications come with a sub-optimality for the resulting traffic schedule that is incurred by the myopic nature of the eventually employed formulations. Even more importantly, the works of [28] and [21] essentially "hide" a substantial part of the underlying problem complexity in the specification of the intermediate destinations, for which they fail to provide systematic algorithmic procedures.

heuristics that seek to trade off the optimality of the generated solution for some control of the complexity of the generated subproblems and their solution through the employed variations of the A^* algorithm. The resulting hybrid schemes will work effectively for fairly sparse problem instances, where it is possible to identify optimized nonconflicting robot paths across small groups of robots; but they will not scale up to more congested environments where the pursued decomposition is not possible [21]. Furthermore, in these harder cases, the premature termination of the executed algorithm will fail to provide any feasible solution for the overall optimization problem.

The problem of the real-time traffic management in guidepath-based traffic systems, as it materializes in the context of the automated, unit-load, zone-controlled MHS that are considered in this work, has also been addressed, although rather sporadically, by the IE/OR community. More specifically, the works of [33], [34] have formulated the problem of managing the AGV traffic that takes place over some complex network topologies by seeking to adapt to this problem perspectives and formulations that were originally developed by the IE/OR communities for the more traditional "vehicle routing" problem [35]. Like the coupled methods discussed in the previous paragraphs, the developed approaches provide a complete characterization of the considered problem and its solution space, and they can return an optimal solution when applicable. But their practical applicability, especially in real-time settings, is severely limited by the extensive computational times that are needed for the solution of the pursued formulations. On the other hand, the work presented in [36] is essentially a decoupled method that employs a "DP with time-windows" model for the generation of the traffic schedules of the various traveling vehicles. Hence, it possesses all the advantages and limitations that were discussed for these methods in the previous paragraphs. Furthermore, all three of the aforementioned methods have failed to consider systematically the problem of deadlock that might be encountered in the generated schedules, and they tend to treat it more as a "nuance" that must be faced at the end, in the context of the generated schedules, instead of an integral issue to be systematically addressed during the generation of these solutions.

Deadlock avoidance and liveness-enforcing supervision for the traffic that is generated by the various MHS classes considered in this work have been studied quite thoroughly and extensively by a group of researchers that come from the controls community, and especially a group working in the area of Discrete Event Systems (DES) [37], [38]. Using linguistic modeling frameworks and abstractions coming from qualitative DES theory [37], [38], and its specialization to the problem of the liveness-enforcing supervision of complex resource allocation systems (RAS) [39], [40], this group has studied the "behavioral" – or "untimed" – dynamics of the considered traffic systems, and it has provided (i) a formal characterization of the corresponding traffic coordination problems, (ii) a notion of "optimal control" for these problems in the form of "maximal (behavioral) permissiveness" of the derived solutions, (iii) a formal establishment of the computability but also the NP-hardness of the corresponding optimal control policies, and (iv) a set of suboptimal but computationally tractable "deadlock avoidance" policies (DAPs) that can ensure the liveness of the underlying traffic while retaining extensive levels of the concurrency and the operational flexibilities that are provided by the underlying system. Characteristic samples of these works can be found in [8], [41], [42], [43], [44], while the works of [45], [46] extend the aforementioned results even in "free-ranging" traffic systems where the system agents travel over a compact 2-dim or 3-dim region. But all these results constitute "preventive" control [37]; i.e., the derived DAPs essentially confine the uncontrolled system behavior in order to prevent deadlock formations.[6]

Finally, when it comes to the current industrial practice, the considered traffic management problems have been addressed through the adoption of configurations for the underlying guidepath networks that minimize the need for traffic control and coordination. As a characteristic example of this attitude, one can mention the popular "tandem" AGV systems, which decompose the overall traffic into a set of unidirectional loops that are interconnected by a set of interfacing buffers [53]. Such a layout essentially abolishes the entire traffic coordination problem since, at each loop, vehicles are filing behind each other in a perpetual cyclical motion over that particular loop. But, on the other hand, the resulting operation involves (a) unnecessarily long trips for transfers that take place between closely located stations, (b) an operational speed for each loop that, for the most part, is regulated by the slowest vehicles, and (c) the need for "double-handling" in the case of transfers involving stations that are located on different loops.

All the above discussion has substantiated, in much more concrete terms, the remark that was made in the opening part of this section that the problem of the (real-time) traffic management in guidepath-based traffic systems has been studied by many different communities, under different modeling assumptions and (performance) objectives that are motivated and defined by the particular needs of those communities. Furthermore, while the resulting models are conceptually similar, they can vary substantially in terms of the reachability properties of the generated traffic, and also in terms of the theoretical and empirical complexity of the feasibility analysis of the posed operational requirements. On the other hand, when it comes to the optimization of the traffic that is generated by these models, most of the resulting formulations end up being NP-hard. Finally, in view of all these subtleties and computational challenges, the existing algorithms that have been developed

[6]A few works within the DES community that have tried to develop a more comprehensive approach to the considered scheduling problem by seeking optimized traffic schedules while considering explicitly the issues of traffic liveness and liveness-enforcing supervision, are those presented in [47], [48], [49], [50]. More specifically, the works of [49], [50] essentially pursue some of the decoupled and the coupled methods that were discussed in the earlier parts of this section, while employing a Petri net (PN) [51] based representation of the traffic dynamics that enables a more explicit representation of the notions of "conflict" that arise in the underlying traffic systems. Hence, the algorithms developed in those works maintain a more proactive and systematic attitude regarding the issue of conflict management in the generated traffic, but they also suffer from the completeness and/or scalability limitations that were discussed in the previous parts of this section. On the other hand, the works of [47], [48] have tried to extend to the traffic scheduling problems considered in this work some methodology that was previously developed for the "job-shop" scheduling problem, and was based on the notion of "(augmented) Lagrangian relaxation" [52] for the corresponding MIP formulations.

for managing the traffic in various subclasses of guidepath-based traffic systems are generally limited in terms of their completeness and/or their scalability.

The work that is presented in the rest of this document extends the current state-of-art, by developing a new class of algorithms for the real-time traffic management in the particular class of guidepath-based transport systems that are considered in it. These algorithms are motivated by, and adapt to the considered problem, some of the broader theory on the design of heuristic algorithms for hard combinatorial optimization problems [14]. A particularly important characteristic of these algorithms is that, (i) in the context of the traffic systems that are considered in this work, they are complete, and, at the same time, (ii) they are able to provide efficient solutions for some very hard problem instances while retaining their computational tractability.[7] Furthermore, the last part of the paper discusses some additional potential that is defined by the presented algorithms for other classes of guidepath-based traffic systems, and the corresponding traffic management problems, that will not satisfy all the operational assumptions that are considered in this work.

III. A FORMAL DESCRIPTION OF THE CONSIDERED TRAFFIC SYSTEM AND THE CORRESPONDING TRAFFIC SCHEDULING PROBLEM

The traffic system that is considered in this work can be formally abstracted as follows: The system consists of a guidepath graph $G = (V, E \cup \{h\})$ that is traversed by a set of agents, $\mathcal{A}$. G is assumed connected and undirected. The edges $e \in E$ of G model the "zones" of the underlying quidepath network. These edges can be traversed by a traveling agent $a \in \mathcal{A}$ in either direction, and, in general, they can hold no more than one agent at any time. An exception to this last rule is the case of edge h, which models a "storage" (or "home") location that can hold an arbitrary number of agents that either have not initiated or have completed their intended trips and, thus, they are essentially retired from the underlying traffic system. For notational convenience, we also define $E \cup \{h\} \equiv \hat{E}$.

In the more general model of the considered operations that was outlined in the introductory section, a trip for some agent a is defined by a sequence of edges $\Sigma_a = \langle e \in E \rangle$ that must be visited by a in the specified order, before the agent eventually retires in edge h.[8] However, since in the context of the work that is presented in this document we consider only the core (sub-)problem of the "rolling-horizon" scheme that was outlined in Section I, in the following we shall assume that each agent $a \in \mathcal{A}$ is associated with a single destination edge, d_a, and the posed problem is to transfer each agent a from its current edge, s_a, to its destination edge, d_a, while minimizing the required transfer time w; i.e., w denotes the "makespan" of the corresponding traffic schedule.

Furthermore, in order to simplify the exposition of our key results, in the main presentation of our algorithm in Section IV we shall assume that the traversal time for any given edge $e \in E$ by any agent $a \in \mathcal{A}$ is deterministic and uniform across all edge-agent pairs $(e, a) \in E \times \mathcal{A}$.[9] This constant traversal time defines a "time unit" for the resulting traffic model, and it induces a natural discretization of the motion dynamics of the considered traffic system. In the context of these discretized dynamics, it is further stipulated that an agent cannot move into an edge e at time t from a neighboring edge, unless e was empty at time $t-1$. [10] Also, some additional assumptions that define the considered traffic coordination problem are as follows: A subset of agents might share the same destination location, an effect that models a "rendezvous" requirement for these agents. And since, in the "rolling-horizon" framework that defines the broader context for the traffic-scheduling problem addressed in this work, the initial locations of the traveling agents in any given problem instance can be their destinations in some earlier iteration, this "rendezvous" possibility further implies the potential coincidence of the initial locations, s_a, for certain agents, as well.

Finally, different traffic models will impose different assumptions regarding the potential (immediate) reversibility of the agent motion on any given edge $e \in E$. In this work, we shall assume that the traveling agents can reverse freely the direction of their motion on any given edge $e \in E$ of the guidepath network. This assumption is practically justified by the motion dynamics materialized in many contemporary MHS, and also by the motion dynamics that govern the traversal of the ion traps by their resident qubits in the context of quantum computing. In the next section we shall also see that the above assumption regarding the reversibility of the motion of the traveling agents within their allocated edges has a critical role in the synthesis of the presented solution to the resulting traffic coordination problem, and therefore, it defines, in a substantial manner, the class of the zone-controlled, guidepath-based traffic systems that are amenable to the results that are presented in this paper.[11]

Example: An example instance of the traffic scheduling problem that is considered in this work is depicted in Figure 1. This problem instance concerns the transport of three agents,

[7]This claim is substantiated in Section V that presents our computational experiments on a series of instantiations of the considered traffic problem.

[8]As explained in Section I, in the AGV operational setting, the edges $e \in \Sigma_a$ model pairs of pick-up and deposition locations that must be visited by the vehicle during its trip. In the application context of quantum computing, the edges of Σ_a are the locations where the corresponding qubit will have its informational content processed, possibly through (controlled) interaction with some other qubits.

[9]This assumption will be removed in Section VI, where we discuss the necessary modifications for the presented algorithm that enable it to cope with the broader case of non-uniform traversal times for the various edge-agent pairs (e, a).

[10]As remarked in Section I, this assumption ensures the required separation of the different agents in lack of a perfect synchronization of the agent transitioning across their various zones, and it also implies the agent inability to swap their occupying edges. Also, as it will be discussed in later parts of this document, in certain variants of the considered traffic systems, this particular condition on the agent motion, when combined with the arbitrary topology of the guidepath graph G and the bidirectional traversal of its edges by the traveling agents, can be the source of deadlocks that will permanently stall the further advancement of the agents involved, and necessitates the proactive management of the underlying traffic with the additional objective of deadlock avoidance [8], [44].

[11]To help the reader obtain a better appreciation of the significance of the assumption of the agent motion reversibility for the presented results, we notice that, under this assumption, the generated traffic will always be deadlock-free, irrespective of how the agents select their routes and advance in them. Hence, the corresponding traffic retains its liveness without any further need for a supervisory control policy. This fact is formally established in Section IV-B.

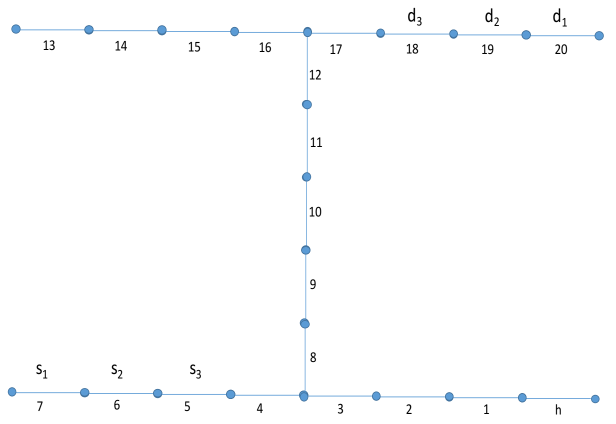

Fig. 1: The problem instance employed in the examples of Sections III and IV.

a_i, $i = 1, 2, 3$, from the corresponding edges that are indicated by s_i in the figure, to the destination edges indicated by d_i. Agents can move by at most one edge at a time, and they can reverse the direction of their motion in their current edge. Furthermore, the "home" edge in the considered configuration is the edge labelled by 'h' in the depicted graph.

It is interesting to notice that the relative positioning of the source and the destination edges for each of the three agents in the underlying guidepath network implies that agent a_1 cannot occupy its destination edge before agents a_2 and a_3 have gone through it, and a similar remark applies to the pair of agents a_2 and a_3. Furthermore, due to the initial placement of the three agents on the depicted guidepath network, it is not possible to route these agents from their current locations to their destinations using the corresponding shortest paths for each agent. In fact, it is easy to see that the synthesis of a feasible solution for this problem instance will require the proactive "sidestepping" of agents a_1 and a_2 to one of the "spears" of the guidepath network that are respectively defined by the edge sets $\{h, e_1, e_2, e_3\}$ and $\{e_{13}, e_{14}, e_{15}, e_{16}\}$, in order to allow the agents behind them to pass ahead.

The above remarks further imply that the considered problem instance cannot be addressed though the sequential logic of the decoupled approaches that were discussed in Section II. On the other hand, as we discuss in the next section, the algorithm that is presented in this work was able to derive an optimal traffic schedule for this problem in a few milliseconds while running on a very simple MacBook Pro.

IV. The proposed algorithm

In this section we present a canonical version of our heuristic algorithm for the traffic coordination problem that was defined in Section III. We start by introducing a formal representation for the solution space of the considered problem, and subsequently, the presented algorithm is motivated and discussed as a "local-search" scheme [14] on this solution space. The presented developments include a worst-case complexity analysis for the basic version of the algorithm, and they also establish formally the algorithm completeness and the correctness of the derived solutions. An extensive numerical evaluation of the presented algorithm is provided in Section V. In addition, Section VI outlines some variations of this algorithm that can enhance the quality of the derived solutions and provide better control of the trade-off between the quality of these solutions and the computational effort that is required for their derivation.

A. A formal representation for the sought traffic schedules

We start the presentation of our algorithm by discussing the key (data) structures that are employed by it for the representation of the sought traffic schedules and some other data sets that are crucial for the performed computations. Hence, let T denote an upper bound to the optimal makespan, w^*; in the subsequent developments of this section we shall show that the traffic-scheduling problem that is defined in Section III is always feasible, and therefore, both T and w^* will be finite positive integers.[12] For any given T, a complete schedule for the considered problem must specify all the edges, e^a_t, that are held by each agent $a \in \mathcal{A}$ at each period $t \in \{0, 1, \ldots, T\}$. Hence, a complete traffic schedule is a set $\mathcal{S}$ of $|\mathcal{A}|$ finite sequences σ^a, each consisting of edges $e \in \hat{E}$ and having length $T + 1$.

In addition, any tentative schedule $\mathcal{S} = \{\sigma^a : a \in \mathcal{A}\}$ will be considered *feasible* if and only if (*iff*) it satisfies the following conditions:

1) $\forall a \in \mathcal{A}, \;\; e^a_0 = s_a \;\wedge\; e^a_T = d_a$.
2) $\forall a \in \mathcal{A}, \forall t \in \{0, 1, \ldots, T{-}1\}, \; e^a_{t+1} \in \{e^a_t\} \cup NH(e^a_t)$, where $NH(e^a_t)$ denotes the set of the neighboring edges of edge e^a_t in graph G.
3) $\forall a, a' \in \mathcal{A}, \; \forall t \in \{1, \ldots, T\}, \; e^a_t = e \;\wedge\; e^{a'}_t = e' \Longrightarrow (e \neq e') \vee (e = e' = h) \vee (e = e' = e^a_q = e^{a'}_q, \; \forall q \in \{0, \ldots, t\}) \vee (e = d_a \wedge e' = d_{a'})$.
4) $\forall a, a' \in \mathcal{A}, \; \forall t \in \{1, \ldots, T\}, \; e^a_t = e^{a'}_{t-1} = e \neq e^a_{t-1} \Longrightarrow (e = h) \vee (d_a = d_{a'} = e)$.

Condition 1 in the above list expresses the fact that in the considered class of feasible schedules, every agent must start from its current location and be at its destination edge by the end of the provided time horizon T. Condition 2 stipulates that any feasible route must observe the connectivity of the underlying guidepath network. Condition 3 enforces the requirement that two agents cannot cohabit on an edge at any time unless (i) it is the "home" edge h, (ii) a common initial location, or (iii) a common destination. In particular, parts (ii) and (iii) of this condition intend to capture the additional traffic dynamics that can be generated by potential "rendezvous" requirements among the traveling agents; please, c.f. Sections I and III for some further discussion on these requirements. Finally, Condition 4 enforces the requirement that an agent a can move into an edge e at a period t only if this edge was empty in period $t - 1$, unless e is the "home" edge or the common destination for a and some other agent a'.

Example: The above definitions regarding the representation of a feasible traffic schedule in this work are highlighted through Table I that provides a feasible (in fact, optimal) traffic schedule for the example problem instance of Figure 1. This

[12]The integrality of these numbers results from the time discretization that was introduced in Section III.

TABLE I: An optimal traffic schedule for the example problem instance of Figure 1.

t	a_1	a_2	a_3	t	a_1	a_2	a_3	t	a_1	a_2	a_3
0	7	6	5	6	8	3	2	12	18	12	10
1	7	6	4	7	9	4	2	13	19	17	11
2	7	5	8	8	10	8	3	14	20	18	12
3	6	4	3	9	11	9	4	15	20	19	17
4	5	8	2	10	12	10	8	16	20	19	18
5	4	3	2	11	17	11	9				

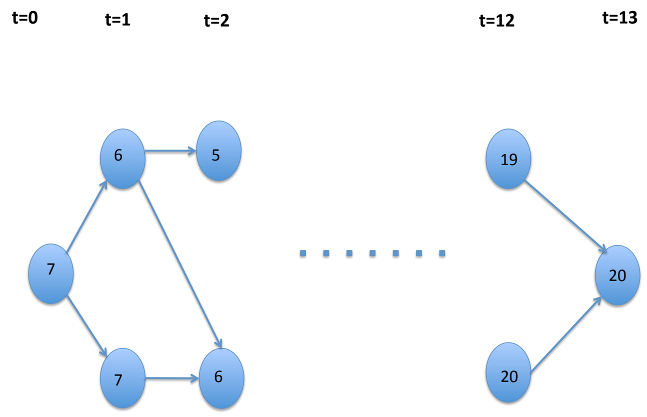

Fig. 2: A fragment of DAG $D(a_1, e_7, 0, 13)$ for the example problem instance of Figure 1. The labels k in the various nodes of the depicted digraph correspond to the various edges e_k in the guidepath graph G of Figure 1.

table provides the edge $e_t^{a_i}$ that is occupied by each agent a_i, $i = 1, 2, 3$, at each period $t \in \{0, \ldots, 16\}$. In the provided schedule, it is interesting to notice (i) the agents' ability to reverse their motion within a particular edge (as exemplified, for instance, by the moves of agent a_1 in periods 3–5), and (ii) the observation of Condition 4 by this schedule (e.g., agent a_2 can make its first move, to edge 5, only at period 2, after agent a_3 has moved out from this edge in period 1). Also, the detailed tracing of this schedule will reveal the agent coordination in their effort to reach their destination, which has agents a_2 and a_3 "side-step" into the spear that is defined by edges e_1, e_2, e_3 and h. $\square$

For the computational needs of the presented algorithm, it is also important to have an efficient representation of all the possible paths that can take any given agent $a \in \mathcal{A}$ from its edge $e_t^a = e \in \hat{E}$ at period t, for any $t \in \{0, 1, \ldots, T-1\}$, to its destination d_a, over the remaining time interval $\{t+1, \ldots, T\}$. Such an efficient representation is provided by a directed acyclic graph (DAG) that will be denoted by $\mathcal{D}(a, e, t, T)$. The nodes of $\mathcal{D}(a, e, t, T)$ are pairs (e', t'), for certain $e' \in \hat{E}$ and $t' \in \{t, \ldots, T\}$, and represent the positioning of agent a at edge e' at time t'. From this description, it is easy to see that the nodes of $\mathcal{D}(a, e, t, T)$ are layered w.r.t. the parameter t' of their labels. On the other hand, a directed edge from node (e', t') to node $(e'', t'+1)$ implies the existence of a feasible path that contains the corresponding move.

Example: Figure 2 demonstrates the $\mathcal{D}(a, e, t, T)$ object by depicting a fragment of the DAG $D(a_1, e_7, 0, 13)$ for the example problem instance of Figure 1. This DAG encodes all the possible paths that take agent a_1 from its initial edge, e_7, to its destination edge, e_{20}, in no more than 13 periods. It is interesting to notice that the layer of nodes corresponding to period 2 does not contain edge e_7, even though it is a viable location for agent a_1 during this period, because, placing agent a_1 at edge e_7 in period 2 will render edge e_{20}, which is the destination edge of agent a_1, inaccessible by this agent in the remaining 11 periods of the considered time horizon. $\square$

1) Start by introducing to the constructed DAG $\mathcal{D}(a, e, t, T)$ the "root" node (e, t), indicating the positioning of agent a at edge e at time t.
2) For each new node introduced in the constructed DAG $\mathcal{D}(a, e, t, T)$, append a next layer of nodes $(e', t+1)$, for all those edges e' that (i) belong to the set $\{e\} \cup NH(e)$, and (ii) there exists a feasible route that can take agent a from edge e' to its destination edge, d_a, in no more than $T-t-1$ time periods; link each node $(e', t+1)$ to node (e, t) with a directed edge leading from the latter node to the former; furthermore, in this expansion, avoid the re-introduction of nodes that have been already introduced in the constructed DAG $\mathcal{D}(a, e, t, T)$.
3) Terminate when all nodes of DAG $\mathcal{D}(a, e, t, T)$ have been processed according to the logic of Step 2.

Fig. 3: A "forward-search" algorithm for constructing the DAG $\mathcal{D}(a, e, t, T)$.

DAG $\mathcal{D}(a, e, t, T)$ can be computed efficiently by the "forward-search" algorithm that is depicted in Figure 3. Condition (ii) in the second step of the algorithm of Figure 3 can be checked efficiently by using the Floyd-Warshall algorithm [54] to precompute all the pairwise shortest distances among the various pairs of edges of the guidepath graph G. Clearly, the number of nodes of the resulting DAG $\mathcal{D}(a, e, t, T)$ cannot be more than $|\hat{E}| \cdot T$, and in most practical instantiations of the considered problem, the actual number of these nodes will be significantly smaller than the above upper bound.

B. *Constructing an initial feasible solution*

Conditions 3 and 4 of the previous section, that must be observed by any feasible schedule for the considered traffic coordination problems, define a notion of "routing conflict" among the traveling agents that, when combined with the arbitrary structure of the underlying guidepath network G, further imply that the task of assessing the feasibility of any given instance of the considered traffic scheduling problems can be pretty challenging, in general [15].

However, in this section we shall show that the particular problem version defined in Section III will always be feasible. We establish this result by construction, i.e., by providing a systematic procedure that will always generate a feasible solution. The feasibility of this construction stems from (i) the availability of the "home" edge h, that can accommodate simultaneously all agents $a \in \mathcal{A}$, and (ii) the presumed reversibility of the agent motion within its running edge. More specifically, the proposed procedure for the construction of a feasible routing schedule can be perceived as a "two-stage" computation where, in the first stage, all agents are collected to the "home" edge h, and in the second stage they are routed to their destination edges d_a. The presumed reversibility of the agent motion within any given edge further implies that in each of these two stages, the traveling agents can be routed to their respective destinations in a way that avoids any potential conflicts among them. The technical details of this construction are established by the following two results.

Lemma 1: For the traffic scheduling problem defined in Section III, it is always possible to reach from the initial traffic state $\mathbf{s}_0$ that is defined by the edge set $\{s_a : a \in \mathcal{A}\}$, to the traffic state $\mathbf{s}_h$ where every agent is located at the "home" edge h.

Proof: Let us assume, without loss of generality, that there exist some agents $a \in \mathcal{A}$ with $s_a \neq h$. Pick any agent, a_1, from this set, that is located closest to the "home" edge h, in terms of the smallest number of edges that must be traversed in order to reach h from s_{a_1}. Clearly, this selection of agent a_1 implies that all edges on any shortest path leading from its current location s_{a_1} to the "home" edge h are free. Hence, agent a_1 can reach edge h while keeping all other agents still. But then, Lemma 1 is proved by an inductive invocation of the above argument for the remaining set of agents a with $s_a \neq h$. $\square$

Proposition 1: For the traffic scheduling problem defined in Section III, there is always a routing schedule that takes all agents $a \in \mathcal{A}$ from their initial locations s_a to their destinations d_a, and abides to the Conditions 1–4 of Section IV-A.

Proof: Lemma 1 has established that it is possible to reach from the initial state $\mathbf{s}_0$ the state $\mathbf{s}_h$ where all agents are collected on the "home" edge h. An argument similar to that used in the proof of Lemma 1 can be used to establish that, from state $\mathbf{s}_h$, all agents $a \in \mathcal{A}$ can be routed to their destinations d_a one at a time, starting with the agents for which their destination edge d_a is furthest from the "home" edge h; the relevant details are pretty straightforward and they are left to the reader. $\square$

Besides establishing the feasibility of all the instantiations of the considered traffic scheduling problem, the proof of Proposition 1 further implies that a feasible traffic schedule can be computed in polynomial time w.r.t. the size of the guidepath network G and the number of the traveling agents, $|\mathcal{A}|$. It is also useful to notice that the construction of an initial feasible traffic schedule for the considered problem instances that was outlined in Lemma 1 and Proposition 1, is meant as a formal argument regarding the problem feasibility. A more practical implementation of this procedure first will prioritize the agent trips at each of the two stages according to the corresponding logic that was established in the proofs of Lemma 1 and Proposition 1, but subsequently it will allow all agents to advance simultaneously to their (stage-dependent) destinations, as long as such an advancement does not impede any agents with higher traveling priority to reach their own destinations; such potential blockages can be assessed and resolved very efficiently through some testing procedures similar to those that were presented in [8] for an efficient implementation of Dijkstra's Banker's algorithm [55] in guidepath-based traffic systems.

Example: In the example problem instance of Figure 1, an initial traffic schedule can be obtained by first routing agents a_3, a_2 and a_1 (in this order) to the "home" edge h through the corresponding shortest paths, and subsequently routing them from edge h to their respective destinations, in the sequence $\langle a_1, a_2, a_3\rangle$. The reader can also check that the first part of this traffic schedule can executed with a minimal makespan of 10 periods, while the more naive implementation of this part that routes the three agents to edge h one at a time will lead to a makespan for this part of 18 periods. Similarly, an efficient implementation of the second part of the proposed routing plan has a minimum required makespan of 14 periods, while the naive implementation of this part that routes one agent at a time, will have a makespan of 33 periods. Hence, the total makespans for the two initial traffic schedules that are constructed by the naive and the efficient implementation of the corresponding procedure are, respectively, 51 and 24 periods. On the other hand, in both of these plans, the last agent to reach its destination in this schedule is agent a_3, and the length of the corresponding sequence σ^{a_3} defines the makespan of the entire schedule.

C. Searching locally for an improved solution

In this section we detail a basic version for the "improving step" of the proposed algorithm. This version will be further enhanced in the last subsection of Section IV and through the discussion that is provided in Section VI.

Hence, suppose that we have already computed a feasible schedule $\mathcal{S} = \{\sigma^a : a \in \mathcal{A}\}$ with makespan w. By the definition of the "makespan" concept, there exists a set of agents $\dot{\mathcal{A}} \subseteq \mathcal{A}$ that reach their corresponding destinations d_a exactly at period w. The proposed improving step seeks to find an agent $\hat{a} \in \dot{\mathcal{A}}$ and a new route $\hat{\sigma}^{\hat{a}}$ for this agent that (i) presents no conflicts with the routes σ^a that are specified by the original schedule $\mathcal{S}$ for all the other agents $a \in \mathcal{A} \setminus \{\hat{a}\}$, and (ii) places agent $\hat{a}$ to its destination location $d_{\hat{a}}$ at a period earlier than w. In such a case, the original schedule $\mathcal{S}$ is replaced by the schedule $\hat{\mathcal{S}} \equiv \{\sigma^a : a \in \mathcal{A} \setminus \{\hat{a}\}\} \cup \{\hat{\sigma}^{\hat{a}}\}$, which is considered as an "improving" schedule w.r.t. $\mathcal{S}$.

To provide a complete description of the improving step that was outlined in the previous paragraph, we also need to describe the mechanism that computes a feasible route $\hat{\sigma}^{\hat{a}}$, provided that such a route is available for the selected agent $\hat{a}$. This can be done by formulating and solving a simple "shortest-path" problem [54] on the DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$, that was introduced in Section IV-A. According to the relevant definitions that were provided in that section, DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$ encodes all the possible routes that take agent $\hat{a}$ from its initial location $s_{\hat{a}}$ to its destination location $d_{\hat{a}}$ no later than period $w-1$, and each node of this DAG carries a label (e, t) indicating that agent $\hat{a}$ is located at edge e at period t. To formulate the aforementioned "shortest-path" problem, we also associate a "cost" with each node (e, t) of DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$ that expresses the number of conflicts that are generated w.r.t. the remaining routes $\sigma^a,\ a \in \mathcal{A}\setminus\{\hat{a}\}$, by placing agent $\hat{a}$ at edge e at period t. Then, it is easy to see that any feasible route $\hat{\sigma}^{\hat{a}}$ w.r.t. the specifications that are defined by Conditions 1–4 in Section IV-A, is represented by a path of zero total cost leading from the "root" node $(s_{\hat{a}}, 0)$ of DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$ to its "terminal" node $(d_{\hat{a}}, w-1)$. The considered method will identify all the zero-cost paths by formulating and solving the corresponding shortest-path problem, and eventually it will select as the new route $\hat{\sigma}^{\hat{a}}$, for agent $\hat{a}$, any of these zero-cost paths that takes agent $\hat{a}$ to its destination as soon as possible.

Example: As remarked in the example of Section IV-B, the naive and the efficient implementations of the construction procedure that was developed in that section will provide initial schedules for the example problem instance of Figure 1 with respective makespans of 51 and 24 periods. Furthermore, in each of these two schedules, the makespan is defined by the routing plan of agent a_3. Hence, assuming that we have started with the schedule that is generated by the naive implementation of the corresponding procedure, the application of the improving step of our algorithm on that schedule will seek to identify a new routing plan for agent a_3 that will bring this agent to its destination edge, e_{18}, in no more than 50 periods. It can be easily checked in Figure 1 that, in view of the routing plans that are provided by the considered schedule for agents a_1 and a_2, such an improved plan for agent a_3 can be obtained either (i) by advancing agent a_3 from edge h towards its destination edge e_{18} before agent a_2 reaches its own destination, or (ii) by initially routing this agent to edge e_{16} (instead of edge h), and keeping it there until agents a_1 and a_2 have cleared through edge e_{17}.

On the other hand, it is also interesting to notice that neither of the above two options will work in the case where the initial traffic schedule is the efficient one with the makespan of 24 periods. In fact, in this case, the improving step that was described in the previous paragraphs will fail to identify an improved routing plan for agent a_3, and the basic implementation of our algorithm will exit with the initial schedule as the proposed solution. $\square$

We close the discussion of this subsection with the following two remarks.

Remark 1: The "shortest-path" problem outlined in the earlier part of this subsection can also be perceived as a "reachability" problem on the subgraph of DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$ that is defined by its zero-cost edges. Therefore, this problem is also solvable through the simpler enumerative techniques that are available for such "reachability" problems [38]. But we have pronounced the "shortest-path" perspective in the above discussion because it provides naturally some additional information that is exploited in the algorithmic enhancement discussed in Section IV-F.

Remark 2: We also notice, for completeness, that in certain variations of the traffic scheduling problem considered in this work, it might be necessary to enforce the additional requirement that, in the derived schedules S, an agent a reaching its destination d_a at some period t will remain at this edge until the end of the planning horizon T. This requirement can be expressed by adding the condition:

5) $\forall a \in \mathcal{A},\ \forall t \in \{1, \ldots, T-1\},\ e^a_t = d_a \Longrightarrow e^a_{t+1} = d_a$

to the four conditions in Section IV-A that define the schedule feasibility in the considered problem context. This new condition will be naturally satisfied during the construction of the initial feasible solution that is presented Section IV-B, thanks to the agent ordering that is adopted in the second part of the corresponding procedure. On the other hand, in the context of the schedule-improving procedure that is presented in this subsection, Condition 5 can be easily enforced by pruning accordingly the DAGs $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w-1)$.

```
1) Use the procedure outlined in Section IV-B in order
   to construct an initial feasible schedule $\mathcal{S}^{(0)}$ for the
   considered problem instance.
2) $\mathcal{S} := \mathcal{S}^{(0)}$.
3) $Q := \dot{\mathcal{A}}$.
4) While ($Q \neq \emptyset$) do
    a) Pick an element $a \in Q$.
    b) Test whether agent $a$ can be used for generating
       an improving schedule $\hat{\mathcal{S}}$.
    c) If the above test is positive, do
        i) $\mathcal{S} := \hat{\mathcal{S}}$.
        ii) Goto Step 3.
    d) else $Q := Q \setminus \{a\}$.
5) Return $\mathcal{S}$.
```

Fig. 4: A basic version of the heuristic algorithm for the traffic-scheduling problem that is considered in this work.

D. A complete canonical version of the presented algorithm

In view of all the above discussion, a complete canonical version for the proposed algorithm can be organized as follows: The algorithm will start by constructing an initial feasible schedule according to the methodology that was presented in Section IV-B. Then, with this initial feasible schedule available, the algorithm will go into an iterative mode that seeks to generate a sequence of improving schedules according to the logic that was described in Section IV-C. More specifically, at the i-th iteration, for $i = 1, 2, \ldots,$, the algorithm will work with the schedule $\mathcal{S}^{(i-1)}$ that was obtained during the previous iterations (or the initial schedule, in the case that $i = 1$), and it will search the corresponding set $\dot{\mathcal{A}}^{(i-1)}$ for an agent $a^{(i)} \in \dot{\mathcal{A}}^{(i-1)}$ that can be used to construct an improving schedule. In the absence of any other pertinent information, this search for the agent $a^{(i)}$ through the elements of the set $\dot{\mathcal{A}}^{(i-1)}$ can be done opportunistically (ensuring, however, that we avoid the repetitive selection of the same agents). On the other hand, once an improving schedule $\hat{\mathcal{S}}$ is constructed, this schedule becomes the incumbent schedule $\mathcal{S}^{(i)}$, and the algorithm advances to iteration $i+1$. The algorithm will terminate at the first iteration, k, where it is not possible to identify an improving schedule for any of the agents $a \in \dot{\mathcal{A}}^{(k-1)}$; at that point, the algorithm will return the schedule $\mathcal{S}^{(k-1)}$ as its final solution, with corresponding makespan $w^{(k-1)}$.

Figure 4 provides a more formal statement of the algorithm that was outlined in the previous paragraph. The completeness of this algorithm, and the correctness of the returned schedule $\mathcal{S}$, result immediately from all the developments that were presented in the earlier parts of this section. Furthermore, since in the new schedule $\hat{\mathcal{S}}$ that is generated at every iteration of the Steps 3–4, one of the traveling agents reaches its destination earlier by one period, the algorithm will terminate in a finite number of iterations, and the returned schedule $\mathcal{S}$ will have a makespan that is no greater than the makespan of the initially constructed schedule $\mathcal{S}^{(0)}$.

Example: The execution of the algorithm of Figure 4 on the example problem instance of Figure 1, with the basic computational scheme that is suggested by the proof of Proposition 1 for the generation of the initial schedule (i.e., the "naive" initial schedule described in the example of Section IV-B), lasted 25 msecs and it resulted in the traffic schedule that is presented in Table I. The reader can verify that this is an optimal traffic schedule for the considered problem instance.

E. Complexity Analysis for the Algorithm of Figure 4

In this subsection, we provide a complexity analysis for the algorithmic version of Figure 4. As discussed in the earlier parts of Section IV, the proposed algorithm will benefit from an initial, "off-line" execution of the Floyd-Warshall algorithm for the computation of the pairwise shortest distances among the edges of the set $\hat{E}$. The same information can also be used for the specification of the shortest paths that are necessary for the construction of the initial schedule $\mathcal{S}^{(0)}$. The computational cost of the Floyd-Warshall algorithm is $O(|\hat{E}|^3)$ [54].

On the other hand, for the iterative part of the algorithm that is presented in Figure 4 we have the following complexity result.

Proposition 2: Let w^* denote the optimal makespan for the considered problem instance, and $\hat{w}$ denote the makespan of the initially constructed traffic schedule $\mathcal{S}^{(0)}$. Then, the (worst-case) computational complexity of the iterative part of the algorithm presented in Figure 4 is $O((\hat{w}^2 - w^{*2})|\mathcal{A}|^2|\hat{E}|^2)$.

Proof: First consider the test that is performed in Step (4b) of the algorithm in Figure 4. As remarked in Section IV-C, this test can be organized efficiently as a "reachability" test that seeks the existence of a zero-cost path in DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w^* + i)$, leading from node $(s_{\hat{a}}, 0)$ to node $(d_{\hat{a}}, w^* + i)$ for some $\hat{a} \in \mathcal{A}$ and $i \in \{0, \ldots, \hat{w} - 1 - w^*\}$. DAG $\mathcal{D}(\hat{a}, s_{\hat{a}}, 0, w^* + i)$ will have $O(|\hat{E}|(w^* + i))$ nodes and each node will have $O(|\hat{E}|)$ emanating edges. Assessing the availability of any of these edges in the underlying reachability problem requires the assessment of potential conflict between the corresponding step and the incumbent schedules of the remaining agents, a task that has a cost of $O(|\mathcal{A}|)$. Hence, the total cost for a single execution of the considered test is $O(|\mathcal{A}||\hat{E}|^2(w^* + i))$. According to the "while"-loop of Step (4), at any given i, this test will be performed $O(|\mathcal{A}|)$ times. Finally, the result of Proposition 2 is obtained by further noticing that $\sum_{i=0}^{\hat{w}-1-w^*}(w^* + i) = O(\hat{w}^2 - w^{*2})$. □

When the initial traffic schedule is obtained through the "naive" implementation of the procedure of Section IV-B, the corresponding makespan, $\hat{w}$, is $O(|\mathcal{A}|\bar{D})$, where $\bar{D}$ denotes the largest distance, among the edges $e \in E$, from the "home" edge h. Hence, the computational complexity of the iterative part of the algorithm of Figure 4 is $O(|\mathcal{A}|^3|\hat{E}|^2\bar{D})$.

On the other hand, the factor $(\hat{w}^2 - w^{*2})$ appearing in the more general result of Proposition 2 reveals quite vividly the computational value of having a good initial traffic schedule. In Section V we also present a set of numerical results that evaluate the computational complexity of our algorithm from a more empirical standpoint that focuses upon the observed execution times. The corresponding results indicate that the algorithm executes very fast. Next, we introduce a variation of the algorithm of Figure 4 that will retain the computational efficiency of the original version, and, at the same time, it will return more competitive solutions for the underlying traffic coordination problem.

F. Enhancing the algorithm performance by digressing into the infeasible region

In this subsection, we discuss a modification of the algorithm presented in the earlier parts of Section IV, that has proved particularly effective in terms of avoiding a premature entrapment into some bad local optima. This is attained by allowing the algorithm to take controlled excursions to infeasible solutions that are close to a reached local optimum, in expectation of the eventual identification of an improved feasible schedule. The computational results reported in Section V will reveal that this new mechanism can enhance very substantially the algorithm performance, enabling the computation of very efficient solutions even for some pretty hard problem instances, with very short execution times.

Under the proposed enhancement, every time that the algorithm of Figure 4 will reach an iteration k where no improving schedule can be identified, it will go into a new phase of the overall computation that works as follows:

First, the algorithm will seek to identify among the paths of the DAGs $\mathcal{D}(a, s_a, 0, (w-1)^{(k-1)})$, $a \in \mathcal{A}^{(k-1)}$, that were computed in iteration k, one of minimal total cost (i.e., minimal conflict w.r.t. the remaining fixed routes of schedule $\mathcal{S}^{(k-1)}$); in the selection of this path, potential ties are resolved arbitrarily. For further reference, let us denote by a_1 the agent that corresponds to the selected path, and by $\sigma_1^{a_1}$ the routing schedule for agent a_1 that is defined by this path. We shall also denote by $\mathcal{S}_1^{(k-1)}$ the routing schedule that is obtained by replacing the route of agent a_1 in schedule $\mathcal{S}^{(k-1)}$ by $\sigma_1^{a_1}$.

Next, the algorithm will try to obtain a new feasible traffic schedule from schedule $\mathcal{S}_1^{(k-1)}$ by eliminating incrementally the various conflicts that are present in this schedule. This is done by starting with the new schedule $\mathcal{S}_1^{(k-1)}$ and trying to identify an agent $a_2 \in A \setminus \{a_1\}$ that possesses a minimal-cost path in the corresponding DAG $\mathcal{D}(a_2, s_{a_2}, 0, w^{(k-1)})$ such that the corresponding total cost is lower than the number of conflicts between agent a_2 and all the remaining agents in schedule $\mathcal{S}_1^{(k-1)}$. Replacing the route of agent a_2 in schedule $\mathcal{S}_1^{(k-1)}$ with the route $\sigma_2^{a_2}$ that is defined by the aforementioned minimal-cost path, will lead to schedule $\mathcal{S}_2^{(k-1)}$, and it can be easily checked that, by its construction, schedule $\mathcal{S}_2^{(k-1)}$ involves a smaller number of conflicts than schedule $\mathcal{S}_1^{(k-1)}$. Hence, iterating the above computation on the new schedule $\mathcal{S}_2^{(k-1)}$ and the further schedules that are obtained in this manner, either we shall reach a schedule $\mathcal{S}_n^{(k-1)}$ involving zero conflicts, or the algorithm will get stuck with an infeasible schedule and no possibility for further reduction of the number of the existing conflicts. In the first case, schedule $\mathcal{S}_n^{(k-1)}$ can be treated as schedule $\mathcal{S}^{(k)}$, i.e., as a new feasible schedule obtained in the k-th iteration of the original algorithm of Figure 4, and the algorithm can move on with the next basic iteration, $k+1$, as specified in Figure 4. In the opposite case,

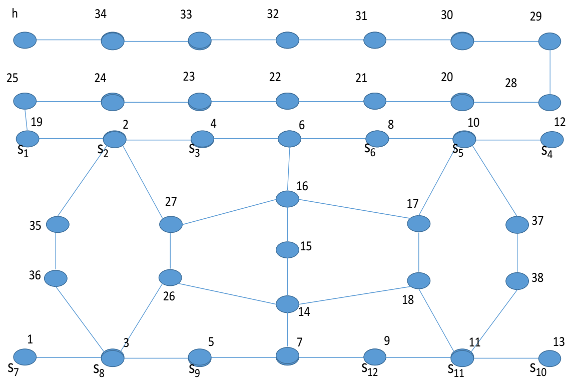

Fig. 5: The problem instance addressed in the first part of Section V.

no further progress is possible, and the algorithm will exit returning as its solution the feasible schedule $\mathcal{S}^{(k-1)}$, that was the best schedule available at the beginning of the considered excursion to the infeasible region. The next section presents a series of computational experiments that demonstrate very vividly the efficacy of the resulting algorithm.

V. Computational results

In the first part of this section we report the results from the application of the presented algorithm on another hard instance of the traffic-scheduling problem that is considered in this work. The second part of the section reports the results of a more extensive computational experiment that has been designed to demonstrate and assess the computational efficiency of the presented algorithm and the quality of the derived solutions, as it is applied on increasingly congested traffic systems. An additional role of this experiment is to identify some further important factors that can impact the algorithm performance w.r.t. its computational efficiency and the quality of the derived solutions. Finally, part of this experiment also demonstrates the limitations of the coupled methods discussed in Section II when applied on increasingly harder problem instances.

A. Applying the algorithms of Section IV on the problem instance of Figure 5

In this part of our computational experiments we applied the algorithms presented in Section IV to the problem instance that is depicted in Figure 5. This problem instance was carefully crafted in order to assess the performance of our algorithm in a very congested environment that involves an extensive overlap among the available paths of the traveling agents towards their target destinations. In addition, the "home" edge h was placed at a very remote location, in an effort to assess the potential inefficiencies that may be incurred by the role of this edge in the construction of the initial traffic schedule, especially in such an adversarial situation.

For a proper understanding of the problem instance that is depicted in Figure 5, the reader should also notice that the problem representation employed in this figure differs slightly from the previous models presented in this paper, in that the traveling agents are placed at the nodes instead of the edges of the underlying guidepath network, and the graph connectivity now defines a notion of "neighborhood" among the different nodes; nevertheless, all the rules of Conditions 1–4 in Section IV, that define the dynamics of the underlying traffic, extend naturally to this new setting, and the same applies to the logic of the algorithms that were presented in that section.[13]

TABLE II: The traffic schedule computed for the problem instance of Figure 5 by the variation of the proposed algorithm that allows for excursions to the infeasible region – c.f.Section IV-F.

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
19	2	4	12	10	8	1	3	5	13	11	9
25	2	4	12	10	6	1	26	5	13	18	7
25	19	4	12	8	6	3	27	5	11	17	14
25	19	2	10	8	6	26	27	7	18	16	15
25	19	35	10	8	4	14	27	7	17	16	15
25	19	36	10	6	2	18	27	7	17	16	15
24	19	3	8	4	35	11	27	14	17	16	15
24	25	5	6	2	36	38	27	18	17	16	15
24	25	7	4	19	3	37	27	18	17	16	15
24	25	9	2	19	5	10	27	18	17	6	15
24	25	9	35	19	5	12	27	18	16	8	15
24	25	9	36	2	5	12	26	17	6	8	15
24	19	9	3	27	5	12	14	16	4	8	15
25	2	9	1	26	5	12	18	6	4	10	15
19	27	9	1	3	5	12	17	8	4	37	16
19	26	9	1	3	5	12	10	8	2	38	6
19	14	9	1	3	5	12	10	8	4	11	6
2	7	9	1	3	5	12	10	8	4	18	6
27	7	9	1	3	5	12	10	8	4	14	6
16	7	9	1	3	5	12	10	8	2	26	6
17	7	9	1	3	5	12	10	8	19	27	4
18	7	9	1	3	5	12	10	8	19	2	4
11	14	9	1	3	5	12	10	8	19	2	4
13	18	9	1	3	5	12	10	8	19	2	4
13	11	9	1	3	5	12	10	8	19	2	4

Under the aforementioned interpretation, the considered problem instance involves twelve agents a_i, $i = 1, \ldots, 12$, each initially located at the corresponding node that is marked by s_i. These agents must swap their positions pairwise, with the corresponding pairs being $\{a_1, a_{10}\}, \{a_2, a_{11}\}, \{a_3, a_{12}\}, \{a_4, a_7\}, \{a_5, a_8\}$ and $\{a_6, a_9\}$. The "home" location is the node at the left-top corner of Figure 5, and it is connected to the rest of the graph by the depicted long path between nodes h and 19.

The application to this problem instance of the canonical version of our algorithm that is depicted in Figure 4, on an HP Z230 workstation with an Intel core i7 processor and 8 GB RAM, running Fedora, lasted 739 msecs and gave a feasible schedule with a makespan equal to 72 periods. We also applied the variation of our algorithm that allows for excursions to the infeasible region, in an effort to overcome a premature entrapment in local optima (c.f. Section IV-F). In this case, the algorithm ran in 8.6 secs, and the obtained schedule is presented in Table II; the corresponding makespan was 24 periods. Finally, we also applied the MIP formulation of [11], [10] to this problem instance. The solution of this formulation

[13]These remarks essentially reveal an existing "duality" in the role of the "node" and "edge" concepts in the graphical representation of the traffic dynamics that are considered in this work. At the end, what really matters is the set of the available "locations" for the traveling agents, and the "proximity" relation that is defined among them.

TABLE III: An optimal traffic schedule for the problem instance of Figure 5 computed through the MIP formulation of [10], [11].

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}
19	2	4	12	10	8	1	3	5	13	11	9
19	2	4	12	17	6	1	26	5	13	38	7
19	27	4	10	18	16	3	26	5	11	37	7
2	27	6	10	18	17	36	26	5	9	38	14
4	16	8	37	18	17	35	26	3	7	11	15
6	16	10	38	14	17	2	27	36	5	9	15
8	16	37	11	26	18	4	27	35	3	7	15
10	17	38	9	26	14	6	27	2	36	5	15
37	18	11	7	26	14	8	16	4	35	3	15
38	18	9	5	27	14	10	17	6	2	36	15
11	18	9	3	27	7	10	17	8	19	35	16
13	18	9	1	26	5	12	17	8	19	2	6
13	11	9	1	3	5	12	10	8	19	2	4

through CPLEX lasted almost 5 days, but the solver was eventually able to come up with an optimal solution of 12 periods; this solution is tabulated in Table III.

B. A more extensive numerical experiment

As stated in the opening part of this section, in this part we report a numerical experiment that was designed to (i) assess more systematically the performance of the traffic-scheduling algorithm that is presented in the paper, and (ii) identify critical factors that will impact this performance. The guidepath network used in this experiment has the particular structure of a "grid", that has been used in similar experiments reported in the past literature, and the traffic congestion is determined by controlling the number of agents that circulate on this grid. The employed grid is depicted in Figure 6, and as in the case of Figure 5, this figure provides a "dual" representation of the employed guidepath graph, where the various zones are represented by the nodes of the depicted graph, and the edges define the neighboring structure of these zones.

There are 133 nodes in the grid of Figure 6. The red node at the center of this grid indicates the location of the "home" zone. In fact, in our experiments, the "home" zone was placed at two different locations: (i) the middle of the guidepath network, as indicated in Figure 6, and also (ii) one of the four corners of the depicted graph.[14] The results that are reported in the rest of this section reveal that the aforementioned placement of the "home" zone can have a significant impact on the performance of the presented algorithm.

The problem instances addressed in the considered experiment, for each of the two placements of the "home" zone, involved a number of agents ranging from 3 to 45, with a step-increase of three agents. The starting and the destination locations for each agent were determined randomly, with the provision that the resulting problem instance was consistent with the problem formulation that was introduced in Section III. Furthermore, the construction of these problem instances was such that the problem instance involving n agents subsumed the problem instance that was defined with $n-3$ agents, $n = 6, 9, \ldots, 45$. This structure of the experiment in terms of the number of the traveling agents and their routing specifications intended to assess the performance of the algorithm as the guidepath network became increasingly more congested.[15]

We executed five replications of the aforementioned experiment and the obtained results are reported in Table IV. More specifically, this table reports for each problem instance addressed in the experiment, an estimate of the optimality gap for the obtained schedule, assessed through the solution of the Lagrangian dual problem of the considered problem instance that is studied in [10], [11]; in particular, for each considered problem instance, the value reported in Table IV is computed by the following formula:

$$\frac{\text{obt. sched. makespan} \; - \; \text{opt. value of Lagrangian dual}}{\text{opt. value of Lagrangian dual}} \times 100$$

Also, Figure 7 depicts the evolution of the average of these numbers, as the traffic density increases from 3 traveling agents to 45. The two plots presented in this figure suggest that the performance of the presented algorithm is pretty close to the optimal in environments with a low zone occupancy by the traveling agents, but this performance degrades as the ratio of the number of the traveling agents to the number of zones of the guidepath network increases to some higher levels. Furthermore, the experienced degradation is higher in the case that the underlying "home" zone is located away from the "center" of the guidepath network. This effect is explainable by the fact that, in this case, the initially constructed schedules by the presented algorithm will possess a highly congested phase that results from the collection of the traveling agents to the "home" zone; the recovery from this congesting effect will require a pretty drastic revision of the originally constructed schedules, that can be facilitated only by "neighborhood" structures involving a high representational and computational complexity.[16]

The careful perusal of the numbers that are reported in Table IV also reveals considerable variability across the performed replications, especially in the case of dense traffic. This variability is interpretable by the distribution of the agent starting locations and their destinations across the entire guidepath network. For reasons similar to those discussed in the previous paragraph, some of the more difficult problem instances for the considered algorithm are those where the destination edges for a group of agents are clustered together in a pretty compact area of the underlying guidepath network. Such a situation implies extensive coupling among all the efficient routing plans for the involved agents, and it requires a very careful coordination of the order in which these agents approach and eventually occupy their destinations.

Next we discuss the results of this experiment from the standpoint of the computational efficacy of the presented algorithm. We start by noticing that the reported experiment was executed on an HP Z230 workstation with an Intel core i7 processor and 8 GB RAM, running Fedora Linux. Furthermore, Table V reports the computational times, in seconds, for

[14]Due to the symmetries of the graph of Figure 6, all these corners are topologically equivalent.

[15]Problem instances with 45 agents imply a pretty congested guidepath network, since, at each time period, the traveling agents occupy about 1/3 of the available zones.

[16]In fact, this effect was also manifested in the example of Section IV-C where the improving procedure was unable to identify an improved routing plan for agent a_3 in the case of the "efficient" initial traffic schedule.

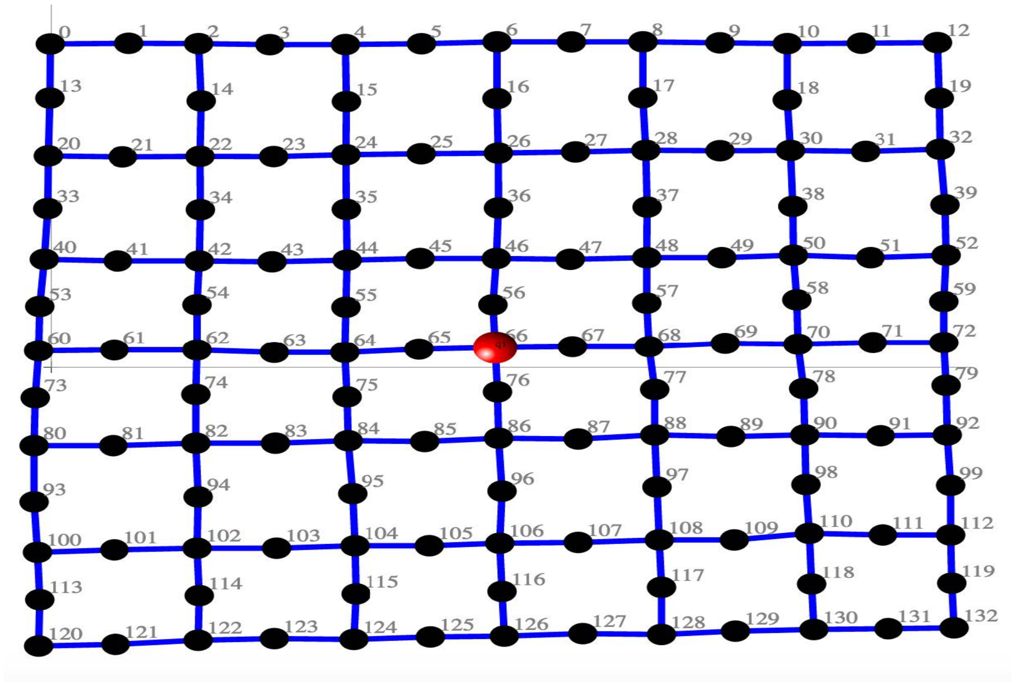

Fig. 6: The guidepath graph used in the numerical experiment of Section V-B.

TABLE IV: Estimates of the optimality gap for the numerical experiment that is reported in Section V-B.

# of	Corner Depot						Center Depot					
agents	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5	Avg.	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5	Avg.
3	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0
15	11.76	7.69	0	0	0	3.89	5.88	7.69	0	0	17.65	6.24
18	0	15.38	0	0	0	3.08	5.88	15.38	0	0	5.88	5.43
21	5.88	0	0	0	0	1.18	5.88	0	0	0	11.76	3.53
24	11.76	5.88	0	0	11.76	5.88	5.88	31.25	5	0	11.76	10.78
27	0	29.41	5	9.52	11.76	11.14	0	25	10	0	23.53	11.70
30	18.18	76.47	10	104.76	29.41	47.76	36.36	25	5	23.81	23.53	22.74
33	50	64.70	400	61.90	52.63	125.85	45.45	43.75	105	176.19	36.84	81.45
36	50	82.35	45	160.87	31.58	73.96	59.09	35.29	45	104.35	36.84	56.11
39	109.09	141.18	435	112.5	47.37	169.03	59.09	58.82	195	25	47.37	77.06
42	186.36	276.47	520	87.5	65	227.07	68.18	100	175	129.17	75	109.47
45	145.45	605.88	625	537.5	100	402.77	86.36	105.88	270	37.5	55	110.95

the execution of the presented algorithm on the aforementioned problem instances, while Figure 8 plots the corresponding averages as a function of the traffic density. As it can be seen in the provided data, our algorithm executes very fast, even for problem instances that correspond to highly congested environments. Also, the plots of Figure 8 are consistent with our earlier remarks regarding the identification of (i) the traffic density and (ii) the centrality of the location of the "home" zone as important factors that determine the difficulty of the considered problem instance.

Finally, Table VI highlights the performance of the MIP formulation of [10], [11] when applied to some of the problem instances that were considered in this experiment. More specifically, we employed this formulation for generating feasible and/or optimal schedules for the various problem instances considered in this experiment, and Table VI reports the results obtained for the first five problem instances of the second problem set corresponding to the "Center Depot" case that was employed in the generation of Tables IV and V. For each of these instances, Table VI provides the performance of the progression of the improving traffic schedules that were generated through the solution of the corresponding MIP formulation by CPLEX, and the computation times involved. In each case, the solver either ran to completion, or it was stopped after one hour of computation. The juxtaposition of the reported results in Table VI with the corresponding results reported in Tables IV and V reveals very clearly the limitations of the MIP-based approach, and its inability to cope effectively with the increasing problem complexity.[17]

[17]We should also mention that the experience reported in Table VI is representative of the performance that we obtained when the MIP-based approach was applied on any other problem set of the considered experiment. Also, we chose to focus this reporting on the "Center Depot" case, since, for the reasons that were discussed in the earlier part of this section, this case is easier than the "Corner Depot" one.

TABLE V: The computational times, in seconds, for the numerical experiment that is reported in Section V-B.

# of agents	Corner Depot Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5	Avg.	Center Depot Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5	Avg.
3	0.578	0.997	1.072	0.207	0.566	0.684	0.064	0.202	0.010	0.117	0.078	0.094
6	1.646	2.108	2.029	2.161	1.648	1.918	0.696	0.503	0.353	0.356	0.438	0.469
9	3.547	3.263	4.435	3.401	3.642	3.658	0.934	1.047	0.583	1.367	1.043	0.995
12	6.426	5.324	7.375	4.754	6.279	6.032	2.883	1.744	1.195	2.535	2.074	2.086
15	10.832	8.912	10.003	6.941	10.234	9.385	5.061	2.887	2.544	6.190	5.150	4.366
18	16.415	14.314	20.085	11.022	15.202	15.408	6.498	4.790	8.507	6.006	7.623	6.685
21	23.325	22.275	30.002	17.016	19.962	22.516	8.452	6.865	11.311	8.286	13.287	9.640
24	41.147	29.571	59.755	26.108	31.421	37.600	10.647	16.931	15.554	10.422	12.425	13.196
27	52.565	40.889	42.776	47.026	42.274	45.106	19.762	18.791	21.880	16.794	20.228	19.491
30	72.063	53.518	111.397	110.126	63.599	82.141	27.615	25.240	29.823	30.649	26.039	27.873
33	97.556	94.606	133.889	105.607	100.407	106.413	33.467	30.906	78.517	40.778	35.837	43.901
36	143.186	103.203	299.298	107.343	82.586	147.123	42.159	52.968	87.547	59.112	47.455	57.848
39	174.941	104.938	183.402	317.641	211.703	198.525	47.996	82.556	87.653	84.874	38.086	68.233
42	209.937	260.069	229.955	234.994	259.284	238.848	74.069	66.680	155.846	88.233	69.370	90.839
45	296.739	251.565	287.195	194.762	430.644	292.181	74.918	105.301	494.313	218.629	76.077	193.848

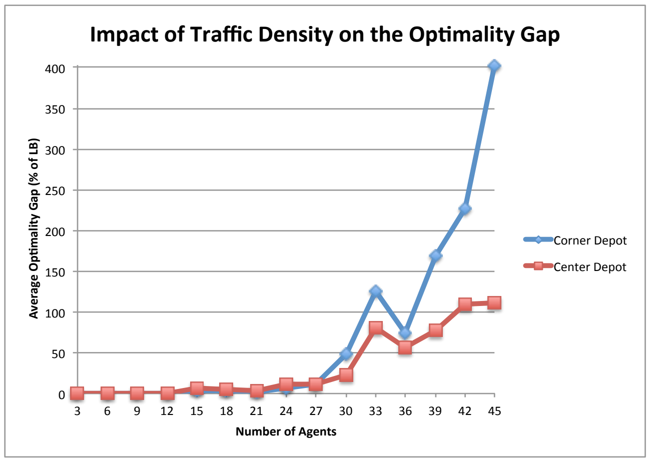

Fig. 7: Plotting the optimality gap for the numerical experiment of Section V-B.

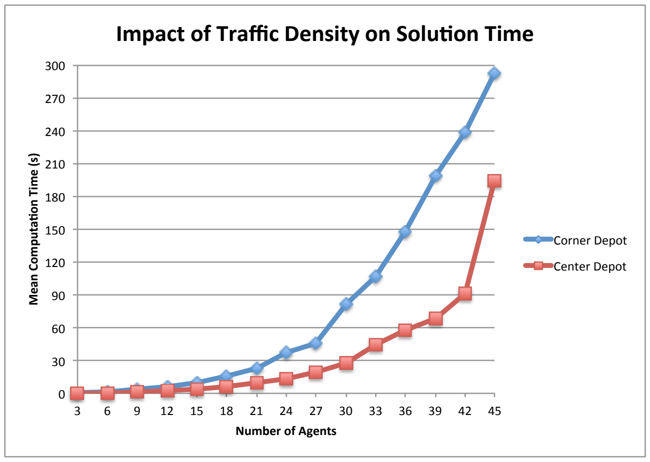

Fig. 8: Plotting the computational times for the numerical experiment of Section V-B.

VI. Discussion

In this section we discuss some further possibilities that can enhance the quality of the traffic schedules that are returned by the considered algorithm, and enable a more

TABLE VI: Characterizing the performance of the MIP-based approach for the considered problem instances.

# of agents	optimality gap	comp. time (secs)	# of agents	optimality gap	comp. time (secs)
3	1.31	0.9	9	1.31	902.56
	0.08	1.06		0.08	916.84
	0.00	1.38		0.00	941.62
	term.	1.42		term.	941.78
6	1.31	4.27	12	1.31	2473.06
	0.08	6.71		0.00	2702.49
	0.00	7.21		term.	2702.55
	term.	7.28	15	term.	3600

explicit management of the trade-off between the quality of the derived solutions and the corresponding computational cost. Furthermore, we address briefly the extension of the presented results towards the real-time management of the traffic taking place in guidepath-based transport systems that might not fit exactly the modeling assumptions that have enabled the algorithmic developments that are presented in this work, and the integration of these results in the "rolling-horizon" framework that was outlined in the introductory section of this paper.

a) Some further enhancements of the presented algorithm: These enhancement possibilities for our algorithm are defined by (i) the revision of the representations for the underlying solution space and the "neighborhood" structure that are employed in the conducted local search, and (ii) the design of additional mechanisms for resolving certain choices that arise during the execution of the algorithm.

Regarding the first of the above two items, one can consider the simultaneous perturbation of the routes of more than one agent during the search for an improved schedule. In order to effect such a simultaneous perturbation of n agent routes, for some $n = 2, 3, \ldots,$ we shall need the redefinition of the DAG structure that is employed in the improving step, so that it is capable to represent, through the employed nodes and their connectivity, all the feasible routes that will enable the considered n agents to reach their corresponding destinations under the imposed timing constraints without any conflicts among themselves. The specification of such a DAG and its systematic construction can be performed in an incremental manner, through a "composing" process of the corresponding single-agent DAGs. An algorithmic procedure for this construction is presented in [56].

[56] also reports some experimental results with this augmented version of the algorithm. As expected, the resulting algorithm can be more powerful than the algorithmic versions that were presented in Section IV, since it employs a more coupling approach in its search for an improved schedule. But, similar to the case of the various coupled methods discussed in Section II, as the parameter n increases to even some moderate values, the size of the generated DAGs explodes pretty fast. An additional interesting finding of the experiments that are reported in [56] is that the for values of n that maintain practical tractability for the resulting algorithm, the improvements incurred in the quality of the obtained solutions are not very significant compared to the solutions that are generated by the algorithm implementation for $n = 1$. More generally, the selection of a pertinent value for the parameter n in any given implementation of the considered algorithm should be based on (i) the available time budget, and (ii) the significance of the incremental improvements that are attained for the returned solutions.

As for the potential development of mechanisms that will resolve the various choices that arise during the execution of the algorithm in a more pertinent manner, one can consider developing a number of heuristics that will employ any available information regarding the topology of the underlying guidepath network, the proximity of the various traveling agents to their respective destinations, and also any information that is contained in the optimal solution of the corresponding "(Lagrangian) dual" problem that was discussed in Section I. [56] also reports some experimentation along these lines. This experimentation has revealed that the integration in the employed algorithm of available information on (i) the particular structure of the underlying guidepath network and (ii) the relative positioning of the different agents with respect to their destinations and each other does have the potential to enhance its computational efficiency and the quality of the solutions that are returned by it. On the other hand, we have not been able to employ with significant advantage the information that is provided in the optimal solutions of the Lagrangian dual problem.

Another way to take advantage of the arbitration that is discussed in the previous paragraph, especially for decisions that cannot be resolved very clearly by an efficient heuristic rule, is by randomizing the corresponding decisions. Such randomization will introduce an additional exploration mechanism in the computational dynamics of the considered algorithm, taking these dynamics to broader regions of the underlying solution space, and increasing, thus, the probability of encountering some high-quality solutions. It also subsumes the frequently used idea of randomizing the starting point of any local search algorithm; in the computational context of our algorithm, the randomization of this particular element can be incurred by randomizing certain indeterminate aspects of the corresponding construction procedure that was described in Section IV-B, like the priority-assignment to agents that are at equal distance from the "home" edge, or the selection of the particular shortest path to be followed by each agent. Of course, the price to be paid for the resulting enrichment of the solution space is the additional computational cost that will result from the need for multiple runs of the algorithm. The detailed determination of the number of these runs constitutes another control parameter for managing the existing trade-off between the quality of the generated schedules and the corresponding computational cost, and it can be based on the available time budget.

b) Extending the presented results to other classes of guidepath-based transport systems: As discussed in the earlier sections of this work, the detailed development of the algorithm that is presented in Section IV has been facilitated by (i) the presence of the "home" edge h in the underlying guidepath network, and (ii) the ability of the traveling agents to reverse their direction of motion on any edge of this network. Moreover, an additional assumption that underlies all the presented developments is the uniformity of the traveling times that are required for the traversal of any given zone by a traveling agent. While these three conditions will be satisfied straightforwardly by most instantiations of the particular applications that are targeted in this work, it is also natural to consider the extension of the presented method to other classes of guidepath-based transport systems that might not satisfy these conditions. In regards to this question, we want to make the following remarks.

First, the reader should notice that while conditions (i) and (ii) in the previous paragraph are especially useful for constructing the initial feasible schedule that is eventually employed by our algorithm in its improving step, the improving step itself can be easily adapted to any new set of assumptions about the traveling agents by adjusting accordingly (a) the construction logic of the DAGs that are employed by this step, and (b) the specification of the costs that are associated with the various nodes of this graph during the formulation and solution of the corresponding shortest-path problems. Hence, the possibility of employing our "local-search" algorithm for the real-time traffic management in guidepath-based transport systems that will not satisfy any of the aforementioned conditions (i) and (ii) is primarily determined by the possibility of obtaining easily an initial traffic schedule for the corresponding problem.

More specifically, in the case of guidepath-based traffic networks that do not possess a "home" edge,[18] the results of [16] that were discussed in Section II suggest that, as long as the target agent configuration is reachable from the initial configuration, then an initial solution can be constructed through the algorithm that is provided in that work with cubic computational complexity w.r.t. the number of the locations that are provided by the underlying guidepath network. Hence, our algorithm can be easily extended to this new class of guidepath-based transport systems.

On the other hand, removing the ability of the agents to reverse the direction of their motion within their assigned zone, while retaining an arbitrary structure for the underlying guidepath network, can give rise to potential deadlocks, and renders the issue of the feasibility of the corresponding traffic scheduling problem that is addressed in this work a significantly harder problem; and these complications arise even in

[18]In the context of the AGV literature, the corresponding MHS are characterized as "closed" [57], [44].

the presence of a "home" edge [15]. Hence, the application of our algorithm in this particular class of guidepath-based traffic systems seems much more challenging. A particular way to deal with the very high computational complexity that arises in this last case, while retaining the completeness and the computational efficiency of our algorithm, is by restricting the overall operation of the underlying traffic system into an operational subspace where (i) the agent "configurations" – or, alternatively, the "traffic states" – admitted by this subspace are mutually reachable from each other, and (ii) for any given pair $(\mathbf{s}, \mathbf{s}')$ of these traffic states, a traffic schedule taking the underlying traffic system from state $\mathbf{s}$ to state $\mathbf{s}'$ is efficiently computable. In the case of guidepath-based traffic systems that possess a "home" edge, such a pertinent subspace can be defined by an adaptation of the notion of the "ordered" traffic state that is presented in [8]; we shall further discuss the employment of this concept in the last part of this section, that addresses the broader issue of the integration of the results that are presented in this paper to the "rolling-horizon" scheme that was outlined in Section I.

Next we consider the necessary modifications of our algorithm in order to accommodate deterministic but nonuniform zone traversal times for each zone-agent pair $(e, a) \in E \times \mathcal{A}$. Along these lines, first, we notice that the procedure for the construction of the initial feasible traffic schedule extends naturally in this case, since the notion of the "shortest paths" that are involved in this construction were defined in Section IV-B on the basis of the number of edges that must be traversed by each agent towards the corresponding destinations, and not through the required traversal time. Furthermore, the logic that supports the improving step can also be adapted easily to this new case by (i) having the length of the discrete time period t defined as the greatest common divisor (GCD) of the zone traversal times for the various zone-agent pairs, and also (ii) having the arc connectivity of the DAGs that are employed by the improving step, reflect the corresponding delays. Everything else in the algorithm logic remains exactly the same as described in Section IV.

c) Integration of the presented results in a "rolling-horizon" framework: As remarked in the introductory section, the "rolling-horizon" scheme that will decompose the overall traffic management problem to a sequence of subproblems of the type that are studied in this work, must ensure the liveness of the resulting traffic, i.e., the ability of every agent to complete its current mission and eventually retire to the "home" location, h, of the guidepath network. In the more technical context of the developments that are presented in this work, traffic liveness implies the feasibility of the various subproblems that are formulated and solved by the employed "rolling-horizon" scheme. Along these lines, an important implication of Proposition 1 is that, for the particular class of guidepath-based transport systems defined in Section III, any agent configuration will be reachable from any other configuration, and the algorithm(s) presented in Section IV will be able to provide the necessary traffic schedules. Hence, the embedding of our algorithm in any employed "rolling-horizon" scheme is a trivial task in this case.

On the other hand, when seeking to employ our algorithm for some other classes of guidepath-based transport systems, it is important that the corresponding "rolling-horizon" scheme generates feasible and efficiently solvable sub-problems, through a pertinent specification of the vehicle "mission" trips and the decomposition of these trips into a sequence of intermediate destinations. The investigation of this last problem is beyond the scope of the current work, but as we mentioned in the previous paragraphs, we believe that this problem can be resolved effectively and efficiently using concepts and algorithms similar to those that are employed in [8] for the liveness-enforcement supervision of the class of AGV systems that are addressed in that work; the detailed development of the corresponding methodological framework is part of our future work w.r.t. this paper.

VII. Conclusions

The work presented in this manuscript has developed an efficient heuristic algorithm for the real-time management of the traffic that is generated in a class of guidepath-based transport systems arising in various MHS applications and in quantum computing. This algorithm customizes to the considered problem the broader "local-search" framework that is frequently used in the solution of hard combinatorial optimization problems. Furthermore, the paper reports results from numerical experimentation with this algorithm which indicate that the proposed algorithm can return very efficient schedules in very short computational times, even for some very hard cases of the considered problem. Hence, our algorithm is very well suited for the real-time applications that have motivated the considered problem. Finally, the discussion that was provided in the last part of the paper, in combination with the extensive literature review of Section II, have also defined a significant potential of the presented algorithm for adaptation and applicability to additional guidepath-based transport systems that will not satisfy the complete set of the structural and the operational features characterizing the transport systems primarily addressed in this paper.

Our future work will seek to extend the applicability of our algorithm along the lines that were discussed in the previous paragraph and in the last part of Section VI. It will also seek to develop a broader decision-making framework for the considered traffic systems that will address higher-level decisions concerning the pertinent assignment of the arising transport requests to the system agents. Finally, it is also interesting to extend the presented developments so that they can address more explicitly various notions of randomness / stochasticity that might arise in the operational context of the considered traffic systems.[19]

References

[1] S. M. LaValle. *Planning Algorthms.* Cambridge University Press, Cambridge, UK, 2006.

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

[19]However, in regards to this last issue, it is also important to notice that the basic paradigm of the "rolling-horizon" scheme that has been discussed in this work, and the systematic replanning that takes place in the context of this scheme through the formulation and solution of the corresponding subproblems, provides a very popular and practical tool for addressing various elements of operational randomness / stochasticity that take place in similar dynamically evolving, sequential decision-making contexts.

[3] R. M. Wilson. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, B*, 16:86–96, 1974.

[4] D. Silver. Cooperative pathfinding. In *Proceedings of the 1st Conference on Artificial Intelligence and Interactive Digital Entertainment*, pages 23–28, 2005.

[5] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, UK, 2010.

[6] J. Nestel-Patt. Semiconductor manufacturing: the final automation wave. *Robotics World*, Spring:32–35, 1998.

[7] D. A. Bodner and S. A. Reveliotis. Flexible semiconductor manufacturing. In J. G. Webster, editor, *Encyclopedia of Electrical and Electronics Engineering, Vol. 7*, pages 596–607. John Wiley & Sons, 1999.

[8] S. A. Reveliotis. Conflict resolution in AGV systems. *IIE Trans.*, 32(7):647–659, 2000.

[9] M. Pinedo. *Scheduling*. Prentice Hall, Upper Saddle River, NJ, 2002.

[10] G. Daugherty, S. Reveliotis, and G. Mohler. Some novel traffic coordination problems and their analytical study based on Lagrangian Duality theory. In *Proceedings of the 55th IEEE Conf. on Decision and Control (CDC 2016)*, pages –. IEEE, 2016.

[11] G. Daugherty, S. Reveliotis, and G. Mohler. A customized dual ascent algorithm for a class of traffic coordination problems. Technical report, ISyE, Georgia Institute of Technology (submitted for publication), 2016.

[12] A. M. Geoffrion. Lagrangian relaxation for integer programming. *Math. Programming Studies*, 2:82–114, 1974.

[13] D. P. Bertsekas. *Nonlinear Programming (2nd ed.)*. Athena Scientific, Belmont, MA, 1999.

[14] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover, Mineola, NY, 1998.

[15] S. Reveliotis and E. Roszkowska. On the complexity of maximally permissive deadlock avoidance in multi-vehicle traffic systems. *IEEE Trans. on Automatic Control*, 55:1646–1651, 2010.

[16] D. Kornhauser, G. Miller, and P. Spirakis. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proc. IEEE Symp. Found. Comput. Sci.*, pages 241–250. IEEE, 1984.

[17] V. Auletta, A. Monti, M. Parente, and P. Persiano. A linear-time algorithm for the feasibility of pebble motion on trees. *Algorithmica*, 23:223–245, 1999.

[18] J. Yu and S. M. LaValle. Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X*, pages 157–173. Springer, 2013.

[19] J. Yu and D. Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *Algorithmic Foundations of Robotics XI*, 2015.

[20] J. Yu and S. M. LaValle. Optimal multi-robot path planning on graphs: Structure and computational complexity. *ArXiv: http://arxiv.org/pdf/1507.03289v1.pdf*, 2015.

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

[22] H. Ma, C. Tovey, G. Sharon, S. Kumar, and S. Koenig. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *AAAI 2016*, pages 3166–3173, 2016.

[23] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, 1979.

[24] Q. Sajid, R. Luna, and K. E. Bekris. Multi-agent path finding with simultaneous execution of single-agent primitives. In *5th Symposium on Combinatorial Search*, 2012.

[25] T. Standley and R. Korf. Complete algorithms for cooperative pathfinding problems. In *Proc. 22nd Intl. Joint Conf. Artif. Intell.*, 2011.

[26] K-H. C. Wang and A. Botea. MAPP: A scalable multi-agent path planning algorithm with tractability and completeness guarantees. *Journal of Artificial Intelligence Research*, 42:55–90, 2011.

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

[28] P. Surynek. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Proc. 12th Pacific Rim Int. Conf. Artif. Intell.*, pages 564–576, 2012.

[29] T. Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proc. AAAI 2010*, 2010.

[30] G. Wagner and H. Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

[31] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[32] P. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. on Systems Science and Cybernetics*, 4:100–107, 1968.

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

[34] G. Desaulniers, A. Langevin, D. Riopel, and B. Villeneuve. Dispatching and conflict-free routing of automated guided vehicles: An exact approach. *The Intl. Jrnl of Flexible Manufacturing Systems*, 15:309–331, 2003.

[35] G. Ghiani, G. Laporte, and R. Musmanno. *Introduction to Logistics Systems Planning and Control*. John Wiley & Sons, Ltd., West Sussex, England, 2004.

[36] 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.

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

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

[39] M. Zhou and M. P. Fanti (editors). *Deadlock Resolution in Computer-Integrated Systems*. Marcel Dekker, Inc., Singapore, 2004.

[40] S. Reveliotis. Logical Control of Complex Resource Allocation Systems. *NOW Series on Foundations and Trends in Systems and Control*, 4:1–223, 2017.

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

[42] M. P. Fanti. Event-based controller to avoid deadlock and collisions in zone-controlled AGVS. *Inlt. Jrnl Prod. Res.*, 40:1453–1478, 2002.

[43] A. Giua, M. P. Fanti, and C. Seatzu. Monitor design for colored Petri nets: an application to deadlock prevention in railway networks. *Control Engineering Practice*, 10:1231–1247, 2006.

[44] E. Roszkowska and S. Reveliotis. On the liveness of guidepath-based, zoned-controlled, dynamically routed, closed traffic systems. *IEEE Trans. on Automatic Control*, 53:1689–1695, 2008.

[45] S. Reveliotis and E. Roszkowska. Conflict resolution in free-ranging multi-vehicle systems: A resource allocation paradigm. *IEEE Trans. on Robotics*, 27:283–296, 2011.

[46] E. Roszkowska and S. Reveliotis. A distributed protocol for motion coordination in free-ranging vehicular systems. *Automatica*, 49:1639–1653, 2013.

[47] D. Y. Liao, M. Jeng, and M. Zhou. Petri net modeling and Lagrangian relaxation approach to vehicle scheduling in 300mm semiconductor manufacturing. In *IEEE Intl. Conf. on Robotics & Automation*, 2004.

[48] T. Nishi, M. Ando, and M. Konishi. Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition and coordination technique. *IEEE Trans. on Robotics*, 21:1191–1200, 2005.

[49] N. Wu and M. Zhou. Shortest routing of bidirectional automated vehicles avoiding deadlock and blocking. *IEEE/ASME Trans. on Mechatronics*, 12:63–71, 2007.

[50] T. Nishi and R. Maeno. Petri net decomposition approach to optimization of route planning problem for AGV systems. *IEEE Trans. on Automation Science and Engineering*, 7:523–537, 2010.

[51] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77:541–580, 1989.

[52] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati. A practical approach to job-shop scheduling problems. *IEEE Trans. on Robotics & Automation*, 9:1–13, 1993.

[53] 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.

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

[55] E. W. Dijkstra. Cooperating sequential processes. Technical report, Technological University, Eindhoven, Netherlands, 1965.

[56] G. Daugherty. *Multi-Agent Routing in Shared Guidepath Networks*. PhD thesis, Georgia Tech, Atlanta, GA, 2017.

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