A Heuristic Approach to the Problem of Min-Time Coverage in Constricted Environments

Young-In Kim and Spyros Reveliotis

***Abstract*—The problem of min-time coverage in constricted environments concerns the employment of networked robotic fleets for the support of routine inspection and service operations taking place in well-structured but constricted environments. In a series of previous works, we have provided a detailed definition of this problem, a Mixed Integer Programming (MIP) formulation for it, a formal analysis of its worst-case computational complexity, and additional structural results for its optimal solutions that also enable the solution of the problem through a partial relaxation of the original MIP formulation. The current work employs those past developments towards the development of a heuristic algorithm able to address larger problem instances that are not amenable to the previous methods. An accompanying set of numerical experiments demonstrates and assesses the computational advantages of this new method. Furthermore, the presented developments can function as building blocks for additional heuristic approaches to the considered problem; this potential is highlighted in the concluding part of the paper.**

***Index Terms*—Networked mobile robotic systems; multi-robot coordination; coverage problems; combinatorial scheduling; heuristic methods**

I. Introduction

In recent years, *multiple mobile robot systems (MMRS)* have been promoted as a very promising solution for the execution of various tasks that are deemed to be too dangerous or physically challenging for the human element [1]. Among the most celebrated examples of such MMRS applications are some search-end-rescue operations where teams of robots have performed successfully various reconnaissance tasks in environments that are too hazardous for the human respondents [2], [3]. But currently, there is also extensive interest in the employment of MMRS for the support of more routine functions evolving in operational environments that are quite stable and well-managed. Some characteristic examples of such applications are: (i) the employment of fleets of mobile robots as the primary material handling devices in many industrial and warehousing facilities [4], [5]; (ii) the delivery by drones or mobile robots of groceries and take-out orders in some residential or rural areas [6], [7]; (iii) the surveillance of public spaces, like public squares and commercial malls, with strategically (re-)positioned cameras that are mounted on mobile robots [8]; and (iv) various other patrolling and data-gathering functions where the robots must re-visit and monitor persistently a specified set of critical locations [9], [10], [11].

In this work, we focus on a class of MMRS applications that concern the employment of mobile robotic teams for the support of routine monitoring and inspection operations in subterranean or other physically constricted structures that are not easily or safely accessible by the human element. Some examples of such environments are (a) the water supply, sewage and other underground utility networks in modern urban areas, (b) mines and other (e.g., archeological) excavation sites, and also (c) the pipeline networks that are used for the transport of oil, gas and other similar commodities over long distances.

During the past decade, these MMRS applications have received extensive attention by the robotics community [12], [13], [14], [15], [16], [17]. Furthermore, these applications have been the focus of a major DARPA challenge known as the "Subterranean – or SubT – Challenge" [18]. This research activity has provided primarily the technological capability that enables (i) stable, flexible and safe motion of the deployed robots in the spatially constricted, and potentially adversarial, corridors (i.e., tunnels, pipes, etc.) that support the robotic traffic, and (ii) reliable communication mechanisms among the robots and the command-&-control (C-&-C) center that manages the overall operation.

But the corresponding literature has also recognized that the effective use of this emerging technological capability in the targeted MMRS applications requires the development of a methodological base that will model, analyze and control the execution of the involved tasks for *logical correctness* and *operational expediency*. A first set of results in this direction has been provided in our recent work of [19]. More specifically, in [19], we have provided: (i) a systematic introduction of the operational requirements of the considered MMRS; (ii) a formal characterization of the traffic management problem that is induced by these requirements, and the detailed positioning of this problem in the existing MMRS literature; (iii) a complete analytical representation of this problem in the form of some mixed integer programming (MIP) formulations [20]; and (iv) a worst-case complexity analysis of the decision problems that underlie these formulations.

It turns out that the considered optimization problems are strongly NP-Hard [21]. Hence, in an effort to cope with this issue, in [22], [23] we have pursued some further structural analysis aiming at alleviating the computational effort for addressing these problems through their MIP formulations. In particular, the work of [23] has leveraged the structural properties identified for these problems in order to develop a strong combinatorial relaxation of their original MIP formulations that employs a much smaller number of integer variables and provides an optimal solution for the original MIPs.

However, the aforementioned relaxations remain MIP formulations themselves, a fact that limits their applicability to larger and/or harder problems instances. Hence, there is a remaining need for pertinent heuristic methods able to provide

Y.-I. Kim and S. Reveliotis are with the School of Industrial & Systems Engineering, Georgia Institute of Technology. Emails: ykim902@gatech.edu, spyros@isye.gatech.edu. This work was partially supported by NSF grant ECCS-1707695.

good solutions for these harder cases. The work presented in this paper leverages the structural results of [23] in order to provide such a heuristic approach. The presented algorithm is of the *"construction"*-type; i.e., it builds the sought solution for the input problem instance incrementally, through a sequence of decisions that are of a myopic, greedy character. In this sense, our work aligns with similar past developments that have been pursued for other hard combinatorial optimization problems, like the Traveling Salesman Problem (TSP) [24] and various machine scheduling problems [25]. A set of experimental results reported in the last part of the paper (i) demonstrate the ability of the presented algorithm to enhance our computational capability for the considered problem, (ii) enable a direct assessment of the practical applicability and scalability of the algorithm, and (iii)) highlight some factors that define the empirical complexity of the algorithm on various problem instances. Finally, the insights and the computational techniques developed in this work also enable the adaptation to the considered problem of other generic heuristic methods for hard combinatorial optimization problems, like those based on local search [26]; we highlight this possibility in the concluding part of the paper.

The rest of the paper is organized as follows: Section II provides a detailed positioning of the considered problem, and overviews the results of [19], [22], and [23] that are necessary for the algorithm developed in this work. The algorithm itself is presented in Section III. This section also provides some analysis of the computational complexity and the worst-case performance of the presented algorithm. Section IV provides the experimental results that demonstrate and evaluate the developments of Section III, and Section V concludes the paper and suggests some directions for future work.

II. The considered coverage problems and their solutions

In this section we overview (i) the formal definition of the coverage problems considered in this work, (ii) a representation of their solutions as dynamic integral flows generated by the robotic fleet, and (iii) a set of properties of these flows and their optimal subset that enable a more efficient solution of the considered problems by relaxing the integrality requirement. The relaxed representation of the problem solution space and its properties play a central role in the main developments of this work that are presented in the next section. On the other hand, due to space considerations, the coverage of the material that is presented in this section, is the minimal necessary for a thorough treatment of the new developments of Section III; a more expansive and more thorough coverage of this material can be found in the references that are provided within the subsequent discussion.

A. The considered coverage problems

A fleet of mobile robots must be used to inspect a set of locations of an underground guidepath network that constitutes a *tree*. The robots are initially located at a C&C-center of the entire facility, which defines the root of the tree, and the targeted locations are its leaves. Furthermore, the tunnels are narrow and the robots have limited sensing and maneuvering capability.[1] Therefore, for safety reasons, the robots must be separated through the imposition of a zoning scheme that splits the underlying guidepath network into zones of unit buffering capacity, and grants access to these zones to the contesting robots through a traffic coordinator. Similar zoning schemes have been used extensively in automated unit-load industrial material handling systems [4], and more recently they have provided a safety control mechanism in other mobile robotic applications, as well [27], [1].

The robots possess wireless communication capability, but their communication range is severely limited by their operational environment. Since these wireless communication links are the only way for each robot to communicate with its operational environment, the robot motion must be coordinated in a way that, at any time point, the active links among the robots define a multi-hop communication network connecting the robots to each other and to the C&C-center. A convenient way to ensure this connectivity is by defining the imposed zones so that neighboring zones guarantee robust connectivity between the robots that occupy these zones, and further stipulating that, at any time point, a zone cannot be occupied unless its parent zone in the underlying tree is also occupied.

Finally, following standard practice in the formal study of the traffic dynamics that are generated by zoning schemes similar to those considered in this work, we further assume that zones are defined in a way that they have uniform traversal time. Then, picking this traversal time as the time unit, we can study the resulting traffic dynamics in discrete time.[2]

In view of the above description, the considered MMRS can be formally represented by a tuple $\mathcal{M} = \langle \mathcal{R}, \mathcal{T} \rangle$, where $\mathcal{R}$ is the set of the robots and $\mathcal{T}$ is a rooted tree representing the tunnel system. The node set V of $\mathcal{T}$ represents the zones of the tunnel system, and the edge set E represents the neighboring relation among the zones.

The root node of $\mathcal{T}$ – i.e., the initial location of all robots and the point of command and control for the entire system – is denoted by o. The set of the leaf nodes of $\mathcal{T}$ is denoted by L. As already stated, each zone $v \in L$ must be visited by some robot for inspection purposes, and the inspection of a leaf zone can be carried out by the visiting robot in the time interval corresponding to a discrete period.

The set of neighbors of a zone $v \in V$ is denoted by $\mathcal{N}(v)$, and for any zone $v \neq o$, $p(v)$ denotes the parent of v in $\mathcal{T}$. Let $z(r,t)$ denote the zone $v \in V$ occupied by robot r at period t. Then, $z(r,t+1) \in \{z(r,t)\} \cup \mathcal{N}(z(r,t))$; i.e., robot r can either remain in the same zone at period $t+1$, or advance to a neighboring zone $v' \in \mathcal{N}(v)$. Furthermore, at any period t, a zone $v \neq o$ cannot contain more than one robot.

On the other hand, at any period t, a group of robots can coordinate their advancement over a path of neighboring zones; i.e., for a group of robots $r_1, r_2, \ldots, r_n$ with $z(r_i,t) \in \mathcal{N}(z(r_{i-1},t))$, for $i = 2, \ldots, n$, we allow $z(r_i, t+1) = z(r_{i+1}, t)$, $i = 1, \ldots, n-1$, provided that robot r_n

[1] As discussed in the introductory section, some examples of such a guidepath network might be a water supply network, a sewage network, or a network of tunnels in an underground mine [12], [13].

[2] Nonuniform traversal times for the system zones can be easily introduced into our model. But this feature would overload the employed notation and complicate some details in the pursued analysis, without adding anything substantial to the expository value of this discussion.

moves itself to a free zone or to the root zone o at period $t+1$. We characterize such a string of robot moves as a robot *flow* occurring at time t, and we shall denote it by $f(z(r_1), z(r_n); t)$. The net effect of this flow is the transfer of a robot from zone $z(r_1, t)$ to zone $z(r_n, t+1)$. Also, the traffic dynamics that were described in the previous paragraphs imply that two flows $f(v_o, v_d; t)$ and $f(v'_o, v'_d; t)$ are conflicting if the supporting paths of these two flows have a common internal node $v \neq o$.

Robots can reverse the direction of their motion within their zone. This assumption is reasonable in the context of the considered applications, and furthermore, it is necessary due to the tree structure of the underlying tunnel system.

Finally, as observed in the opening part of this section, the communication connectivity among the robots and the system controller is established by stipulating that, for every zone $v \neq o$ and every period t,

$$\exists\ r \in \mathcal{R} : z(r,t) = v \implies \exists\ r' \in \mathcal{R} : z(r',t) = p(v) \quad (1)$$

The above requirement implies that for every zone v occupied by a robot in period t, all the zones in the path connecting zone v to the root zone o in tree $\mathcal{T}$ are also occupied by a robot in period t. Furthermore, the root zone o is always occupied by at least one robot.

We want to determine a plan that will advance the robots $r \in \mathcal{R}$ in a way that is consistent with the above assumptions regarding the robot capabilities and the zone allocation protocol, and at the end of its execution, each leaf zone $v \in L$ will have been visited by some robot.[3] Let $\mathcal{P}$ denote the set of feasible plans, and for every plan $P \in \mathcal{P}$ and leaf node $v \in L$, let $C(v; P)$ denote the first period that plan P places a robot in zone v. We are especially interested in plans P^* such that

$$P^* = \arg\min_{P \in \mathcal{P}} \max_{v \in L} C(v; P) \quad (2)$$

or

$$P^* = \arg\min_{P \in \mathcal{P}} \sum_{v \in L} C(v; P) \quad (3)$$

Each of Eqs 2 and 3 defines a combinatorial optimization – or traffic-scheduling – problem. The traffic-scheduling problem defined by Eq. 2 is characterized as the *Makespan-minimization* problem, or the *M-problem*, and the traffic-scheduling problem defined by Eq. 3 is characterized as the *Total Visitation Time-minimization* problem, or the *TVT-problem*. Also, in [19] it is shown that these two problems are in a Pareto optimal relationship [28], [29]; i.e., there are MMRS where the sets of optimal plans for these two problems have no common element. Hence, these two problems require separate treatments.

B. Representing the solution space of the considered problems

Preliminaries: Consider an M- or TVT-problem instance $\mathcal{M} = \langle \mathcal{R}, \mathcal{T} \rangle$. The unique path connecting any nodal pair $\{v_1, v_2\}$ of tree $\mathcal{T}$ is denoted by $\pi(v_1, v_2)$, and the length $l(v_1, v_2)$ of path $\pi(v_1, v_2)$ is defined by the number of edges in it. Since tree $\mathcal{T}$ is undirected, $\pi(v_1, v_2) \equiv \pi(v_2, v_1)$ and $l(v_1, v_2) = l(v_2, v_1)$. A single node is considered as a path of zero length. Furthermore, in the next section, we also use the notation $\pi(v_1, v_2)$ to denote the set of nodes of tree $\mathcal{T}$ that belong on this path.

For a node $v \in V$, the length $l(o, v)$ of the path $\pi(o, v)$ defines the depth of node v in tree $\mathcal{T}$. In particular, node o has zero depth. We also define the depth of tree $\mathcal{T}$ by

$$l(\mathcal{T}) \equiv \max_{v \in L} l(o, v) \quad (4)$$

In view of the requirement of Equation 1, the considered problem instance $\mathcal{M}$ is *feasible* if and only if

$$|\mathcal{R}| \ \geq \ l(\mathcal{T}) \quad (5)$$

Next, consider a strict subset $\hat{L}$ of L. The paths $\pi(o, v),\ v \in \hat{L}$, induce a subtree $\widehat{\mathcal{T}}$ of tree $\mathcal{T}$. Also, let $\hat{V} \subset V$ denote the nodes of $\widehat{\mathcal{T}}$. Then, for a leaf node $\tilde{v} \in L \setminus \hat{L}$, we define

$$b(\tilde{v}, \widehat{\mathcal{T}}) \equiv \arg \max_{v' \in \pi(o,\tilde{v}) \cap \hat{V}} l(o, v') \quad (6)$$

and

$$d(\tilde{v}, \widehat{\mathcal{T}}) \equiv l\big(b(\tilde{v}, \widehat{\mathcal{T}}), \tilde{v}\big) = l(o, \tilde{v}) - l\big(o, b(\tilde{v}, \widehat{\mathcal{T}})\big) \quad (7)$$

Equations 6 and 7 define the *distance* of the leaf node $\tilde{v}$ from subtree $\widehat{\mathcal{T}}$, and the node $b(\tilde{v}, \widehat{\mathcal{T}})$ can be perceived as the *projection* of the leaf node $\tilde{v}$ on the subtree $\widehat{\mathcal{T}}$.

An exact representation of the solution space of the M- and TVT-problems: A plan P for the M- or the TVT-problem can be considered as a sequence of distributions, $\langle \mathcal{D}_t,\ t = 0, 1 \ldots, \bar{T} \rangle$, of the system robots to the various zones of the underlying tunnel system. In the initial distribution $\mathcal{D}_0$, all the robots are located in the root zone o of tree $\mathcal{T}$. For the periods $t \geq 1$, the distribution $\mathcal{D}_t$ is obtained from the distribution $\mathcal{D}_{t-1}$ by relocating a number of robots from their zones at period $t-1$ to some neighboring zone, while abiding to the assumptions about the maneuvering capabilities of the robots and the zone allocation protocol that were introduced in Subsection II-A. Furthermore, all leaf nodes of $\mathcal{T}$ must have been visited by some robot by period $\bar{T}$.

For a feasible M- or TVT-problem instance, there is always a plan P where $\bar{T}$ can be restricted to $|V|$ (i.e., the number of zones of tree $\mathcal{T}$); the plan that visits the zones of $\mathcal{T}$ in a depth-first sense, one new zone at a time, is such a plan.

Also, for any given $\bar{T}$, the set of feasible plans with duration no longer than $\bar{T}$, can be represented by the following set of variables and constraints:

- *State variables* $x_{v,t},\ v \in V,\ t \in \{0, 1, \ldots, \bar{T}\}$, with each variable $x_{v,t}$ being a nonnegative integer variable indicating the number of robots in zone v at period t.
- *Control variables* $u_{v,v',t},\ v \in V,\ v' \in \mathcal{N}(v),\ t \in \{1, \ldots, \bar{T}\}$, with each variable $u_{v,v',t}$ being a nonnegative integer variable representing the number of robots moving from zone v to neighboring zone v' at period t.
- *Constraint set*

$$x_{o,0} = |\mathcal{R}| \quad (8)$$

$$\forall v \in V \setminus \{o\}, \quad x_{v,0} = 0 \quad (9)$$

$$\forall v \in V,\ \forall t \in \{1, \ldots, \bar{T}\},$$
$$x_{v,t} = x_{v,t-1} + \sum_{v' \in \mathcal{N}(v)} \big(u_{v',v,t} - u_{v,v',t}\big) \quad (10)$$

[3] A technical definition of the "plan" concept, that must coordinate the robot advancements through the system zones, is provided in the next subsection.

$$\forall v \in V,\ \forall t \in \{1,\ldots,\bar{T}\},\quad \sum_{v' \in \mathcal{N}(v)} u_{v,v',t} \leq x_{v,t-1} \tag{11}$$

$$\forall v \in V \setminus \{o\},\ \forall t \in \{1,\ldots,\bar{T}\},\quad x_{v,t} \leq 1 \tag{12}$$

$$\forall v \in V \setminus \{o\},\ \forall t \in \{1,\ldots,\bar{T}\},\quad x_{v,t} \leq x_{p(v),t} \tag{13}$$

$$\forall v \in L,\quad \sum_{t=0}^{\bar{T}} x_{v,t} \geq 1 \tag{14}$$

Constraints 8 and 9 define the initial distribution of the robots by means of the state variables $x_{v,0},\ v \in V$. Constraint 10 expresses the evolution of the robot distribution to the system zones at period t, based on the decisions that are expressed by the control variables $u_{v,v',t}$. Constraint 11 stipulates that the control decisions at period t must be feasible with respect to the robot distribution over the system zones at period $t-1$. Constraint 12 enforces the unit buffering capacity of the zones $v \neq o$. Constraint 13 enforces the condition of Eq. 1. Finally, Constraint 14 expresses the requirement that every leaf node has been visited by period $\bar{T}$.

The unit capacity presumed for the zones of $\mathcal{T}$ implies that all the state variables $x_{v,t}$ and the control variables $u_{v,v',t}$ are actually binary variables. Furthermore, it can be easily checked that the explicit enforcement of the integrality of the control variables $u_{v,v',t}$ for any solution of the constraint set 8–14, will also result in integral values for the state variables $x_{v,t}$.

Finally, for any feasible plan P defined by the above set of variables and constraints, the corresponding objective values for the M- and the TVT-problem are, respectively, $\max_{v \in L} C(v;P)$ and $\sum_{v \in L} C(v;P)$, where, for every leaf node $v \in L$, $C(v;P)$ is defined as in Subsection II-A.

A relaxed representation of the solution space of the M- and TVT-problems: Next, consider the constraint set that is obtained from Constraints 8–14 by (i) relaxing the state variables $x_{v,t}$ and the control variables $u_{v,v',t}$ to be nonnegative reals taking values in the interval $[0,1]$, and (ii) replacing Constraint 14 with the constraint

$$\forall v \in L,\quad \exists t \in \{1,\ldots,\bar{T}\} :\ x_{v,t} = 1 \tag{15}$$

In this new solution space, the values of the variables $x_{v,t}$ are perceived as the corresponding amount of *fluid* located at node v at period t. The pricing of the entire set of variables $\{x_{v,t} : v \in V\}$, at any period $t \in \{0,1,\ldots,\bar{T}\}$, defines a *fluid distribution* at period t. Also, the pricing of the set of variables $\{u_{v,v',t} : v \in V,\ v' \in \mathcal{N}(v)\}$, at any period $t \in \{0,1,\ldots,\bar{T}\}$, defines a *flow* $F(t)$ at period t, and a sequence of flows $F = \langle F(1), F(2), \ldots, F(\bar{T})\rangle$ defines a *flow plan*. The set of flow plans that constitute feasible solutions in the new solution space, is denoted by $\mathcal{F}$. In analogy to the corresponding definitions for the original M- and TVT-problems, for any feasible flow plan F and any leaf node $v \in L$, we also set

$$C(v;F) \equiv \min\left\{t \in \{1,\ldots,\bar{T}\} : x_{v,t} = 1.0\right\}$$

and we consider the fluid amount of 1.0 as a "target" fluid level that must be attained by any leaf node $v \in L$. Finally, substituting the above $C(v;F)$ values in Equations 2 and 3, we obtain variations of the M- and TVT-problems defined in the new solution space.

Clearly, these new versions of the M- and TVT-problems constitute relaxations of their counterparts that are defined by Constraints 8–14 and the integrality requirement for the variables $x_{v,t}$ and $u_{v,v',t}$. Yet, Theorem 1 in [23] establishes that there exists a flow plan subset $\hat{\mathcal{F}}$ that (i) contains optimal flow plans for the relaxed M- and TVT-problem versions and (ii) each flow plan $F \in \hat{\mathcal{F}}$ can be converted to a plan $P \in \mathcal{P}$ for the original M- or TVT-problem instance with an objective value that is no worse than the objective value of flow plan F. Furthermore, this conversion can be performed in time polynomial with respect to $|V|$.

In more technical terms, flow plan subset $\hat{\mathcal{F}}$ is defined by the following two requirements:

1) Each flow plan $F \in \hat{\mathcal{F}}$ must be *structurally minimal*, i.e., it must minimize the quantity
$$\sum_{v \in V} \sum_{t=1}^{\bar{T}} |x_{v,t} - x_{v,t-1}|$$
among all the flow plans that have the same value with it for the primary objective functions of Equations 2 and 3.[4]
2) Each flow plan $F \in \hat{\mathcal{F}}$ must also satisfy the following condition:
$$\forall v \in V \setminus L,\quad \sum_{t=1}^{\bar{T}} \sum_{v' \in V : p(v') = v} u_{v,v',t} \ \in\ \mathbb{Z}^+ \tag{16}$$
where $\mathbb{Z}^+$ denotes the set of strictly positive integers; i.e., the total flow conveyed from any internal node v of tree $\mathcal{T}$ towards its children must be integral.

In addition, Proposition 3 of [23] establishes that any structurally minimal flow plan F violating the integrality condition of Eq. 16 can be converted to a structurally minimal flow plan $\tilde{F}$ that satisfies this integrality condition and has the same objective value with F, by formulating and solving a linear program (LP).

In the next section, we leverage the above results of [23] in order to develop the proposed heuristic algorithm for the M- and TVT-problems.

III. The new heuristic algorithm for the M- and TVT-problems

A. The defining logic of the presented algorithm

As remarked in the introductory section, the presented algorithm is a *"construction" heuristic* for the considered problems; i.e., it synthesizes a solution for the input M- or TVT-problem instance in an incremental manner, through a sequence of decisions of a myopic, greedy character.

More specifically, the algorithm perceives a feasible solution for any given M- or TVT-problem instance as a *permutation* of the leaf nodes $v \in L$ of the underlying tree $\mathcal{T}$ that is specified by the visitation times of these nodes in the corresponding plan. In general, an efficient solution will tend to minimize the elapsing time between the visitation of two consecutive leaf nodes in the corresponding permutation; and this remark is true for, both, the M- and the TVT-problem. Hence, the

[4] In more conceptual terms, a structurally minimal flow plan prevents the unnecessary fluctuation of the nodal fluid levels from period to period.

presented algorithm seeks to construct a permutation of the leaf nodes $v \in L$, one leaf node at a time, by minimizing the visitation time of the newly added node while observing the predetermined visitation times for the previously entered leaf nodes in the sequence. Similarly, the first node in the constructed permutation is a leaf node that can be reached from the root node o – which is the initial location of all robots – as fast as possible. Finally, in the current implementation of the algorithm, any arising ties are solved arbitrarily.

The problem of determining the minimal possible visitation time of any candidate leaf node to enter the partially constructed plan at each iteration, while observing the visitation times of the previously entered leaf nodes, is a complex combinatorial optimization problem. Fortunately, the complexity of these (sub-)problems can be drastically reduced by formulating and solving them in the relaxed representational space that was introduced in Section II-B, while leveraging the corresponding results of [23]. In addition, it is possible to thin out the set of the candidate nodes at each major iteration of the algorithm, without compromising its defining logic and the efficiency of the derived solutions. We detail all these aspects of the algorithm in the next subsection.

B. The algorithm

Some major data structures and operations of the presented algorithm: Two central data structures that are defined and maintained by the presented algorithm, are the lists $\hat{S}$ and $\hat{H}$. At any time point during the execution of the algorithm, list $\hat{S}$ contains the sequence of the leaf nodes $v \in L$ that have already entered the partially constructed plan P, according to the logic that was outlined in the previous subsection. Let $\hat{S} = \langle v^1, v^2, \ldots, v^i \rangle$, for some $i \in \{1, \ldots, |L|\}$. Then, $\hat{H} = \langle h_1, h_2, \ldots, h_i \rangle$, with $h_j,\ j = 1, \ldots, i$, reporting the visitation time of node v^j in list $\hat{S}$ by the specified (partial) plan P. From the discussion of Subsection III-A, it is clear that $h_j \leq h_{j+1}$, for all $j = 1, \ldots, i-1$, and h_i defines the makespan of the corresponding partial plan. Both lists $\hat{S}$ and $\hat{H}$ are initially empty, and they are augmented one element at a time, as discussed in the later parts of this subsection.

Furthermore, the assessment of the various candidate leaf nodes to enter the partially constructed plan at each iteration, requires the feasibility assessment of a series of partial plans represented by pairs $(\hat{S}, \hat{H})$, for some given lists $\hat{S}, \hat{H}$. From the discussion on the relaxed representation of the M- and TVT-problems that was provided in Section II-B, the feasibility of these partial plans can be assessed by (i) defining the subtree $\widehat{\mathcal{T}}$ that is induced by the paths $\pi(o, v^j),\ v^j \in \hat{S}$, and (ii) testing on this subtree the feasibility of the constraint set that is defined by Constraints 8–13 and the additional constraints

$$\forall v \in V,\ \forall v' \in \mathcal{N}(v),\ \forall t \in \{1, \ldots, \bar{T}\},\quad 0 \leq u_{v,v',t} \leq 1 \tag{17}$$

$$\forall v^j \in \hat{S},\quad x_{v^j,h_j} = 1 \tag{18}$$

This test can be performed very efficiently by solving an LP that is defined by the aforementioned constraints and the trivial objective function "$\min 0$". In the following, we shall represent such a feasibility test by $\mathcal{LP}(\langle \mathcal{R}, \widehat{\mathcal{T}} \rangle, \hat{S}, \hat{H})$.

Algorithm 1 The main part of the presented algorithm

Input: An M- or TVT-problem instance $\langle \mathcal{R}, \mathcal{T} \rangle$
Output: A feasible plan P

```
1:  h_1 := |V| + 1
2:  for v ∈ L do
3:     if l(o, v) < h_1 then
4:        v^1 := v;
5:        h_1 := l(o, v);
6:     end if
7:  end for
8:  Ŝ := {v^1};
9:  Ĥ := {h_1};

10: for i ∈ {2, ..., |L|} do
11:    T̂ := the subtree of T induced by
                 the paths π(o, v^j), v^j ∈ Ŝ;
12:    C := ∅;
13:    for v ∈ L \ {Ŝ} do
14:       C := C ∪ {v};
15:       for c ∈ C \ {v} do
16:          if b(v, T̂) = b(c, T̂) then
17:             if l(o, v) < l(o, c) then
18:                C := C \ {c};
19:                break;
20:             else
21:                C := C \ {v};
22:                break
23:             end if
24:          end if
25:       end for
26:    end for
27:    (Ŝ, Ĥ) := APPEND(⟨R, T⟩, Ŝ, Ĥ, C);
28: end for

29: F := LP(⟨R, T⟩, Ŝ, Ĥ);
30: if F violates the integrality condition of Eq. 16 then
31:    F := F̃, where F̃ is obtained from F via Proposition
       3 of [23];
32: end if
33: Convert flow plan F to a plan P using the results of
    Theorem 1 of [23];
34: return P.
```

The main algorithm: The pseudocode for the main part of the presented algorithm is provided in Algorithm 1. The algorithm receives as input an instance of the M- or the TVT-problem and returns the generated plan P.

Lines 1–9 in Algorithm 1 select the first node, v^1, to enter the generated list $\hat{S}$, and they also determine the corresponding visitation time, h_1, that enters the list $\hat{H}$. The selected node v^1 is the first encountered leaf node of $\mathcal{T}$ having minimal depth.

Lines 10–28 construct the remaining parts of lists $\hat{S}$ and $\hat{H}$, selecting the elements to enter these lists one pair at a time. More specifically, the algorithm goes through $|L| - 1$ primary iterations (c.f. Line 10), and each such iteration consists of the following two parts:

I) In the first part, corresponding to Lines 11–26, the algorithm determines the set $\mathcal{C}$ of the candidate nodes to enter the list $\hat{S}$ at the current iteration. In principle, $\mathcal{C} = L \setminus \{\hat{S}\}$, where $\{\hat{S}\}$ is the current nodal content of $\hat{S}$. However, it is easy to see that in an optimized plan that respects the visitation requirements specified by the current lists $\hat{S}$ and $\hat{H}$, for any two nodes v_1, v_2 in $\mathcal{C}$ with $b(v_1, \widehat{\mathcal{T}}) = b(v_2, \widehat{\mathcal{T}})$ and $d(v_1, \widehat{\mathcal{T}}) < d(v_2, \widehat{\mathcal{T}})$, the required time to reach node v_2 is longer than the required time to reach node v_1. Hence, the algorithm removes from set $\mathcal{C}$ any such noncompetitive nodes during the construction of this set. Furthermore, for any subset of leaf nodes in $L \setminus \{\hat{S}\}$ that have the same distance and the same projection with respect to tree $\widehat{\mathcal{T}}$, the algorithm enters in $\mathcal{C}$ only the first such encountered element of this subset. This "thinning" of the candidate sets $\mathcal{C}$ is very significant for controlling the execution time of the algorithm.[5]

II) The nodal candidates that remain in the thinned set $\mathcal{C}$ are evaluated in Line 27 of Algorithm 1, by calling the procedure APPEND, that is described in a subsequent part of this subsection. This procedure returns a leaf node from the set $\mathcal{C}$ and the corresponding visitation time that are respectively appended to the current lists $\hat{S}$ and $\hat{H}$.

The last part of Algorithm 1, corresponding to Lines 29–34, constructs the plan P for the input M- or TVT-problem from the flow plan F that corresponds to the computed lists $\hat{S}$ and $\hat{H}$. This conversion is based on the developments of [23] that were discussed in the closing part of Section II-B.

The procedure APPEND: The pseudocode for procedure APPEND appears in Algorithm 2. This procedure takes as input (i) an instance $\langle \mathcal{R}, \mathcal{T} \rangle$ of the M- or the TVT-problem, (ii) two lists $\hat{S}$ and $\hat{H}$ defining a feasible partial visitation plan for the leaf node subset contained in list $\hat{S}$, and (iii) a set $\mathcal{C} \subseteq L \setminus \{\hat{S}\}$ collecting all the leaf nodes that are competitive candidates for extending the current partial plan by one node. The procedure returns a new list pair $(\hat{S}, \hat{H})$ that consists of the corresponding input lists augmented, respectively, with the selected candidate and the corresponding visitation time.

As indicated in the presented pseudocode, this procedure consists of three major parts. In the first part, consisting of Lines 1–11, the input candidate set $\mathcal{C}$ is scanned in order to detect a leaf node in it that can be appended to the current partial plan without increasing its makespan h_{i-1}. For each leaf node $c \in \mathcal{C}$, the aforementioned test is performed by formulating and solving the corresponding testing LP along the lines that were discussed in the first part of this subsection (c.f. Lines 3–6). As soon as such a node is detected, the procedure picks this node as the node to be returned to the calling algorithm, and breaks out of the for-loop of Lines 2–11 (c.f. Lines 6–10). Furthermore, in this case, Lines 1 and 12 imply that the for-loop of Lines 13–36 will be skipped, and the procedure proceeds to its last part, defined by Lines 37–39, where it updates the input lists $\hat{S}$ and $\hat{H}$ with its selection and returns the results to the calling algorithm.

The part of procedure APPEND that is defined by Lines 12–36, is executed when the first part discussed above fails to detect a candidate node that can augment the current partial plan without increasing its makespan. This part scans the candidate nodes $c \in \mathcal{C}$ trying to determine, for each such node, the earliest time, $\hat{h}_c$, that it can be visited, in view of the visitation requirements that are posed by the partial plan specified by the current lists $\hat{S}$ and $\hat{H}$.

[5]An alternative implementation may choose randomly among all the candidate leaf nodes with the same distance and the same projection with respect to tree $\widehat{\mathcal{T}}$. Such randomization enables the generation of many different solutions through the repetitive execution of the algorithm.

Algorithm 2 Procedure APPEND

Input: An M- or TVT-problem instance $\langle \mathcal{R}, \mathcal{T} \rangle$;
$\hat{S} := \langle v^1, \ldots, v^{i-1} \rangle$; $\hat{H} := \langle h_1, \ldots, h_{i-1} \rangle$;
$\mathcal{C} := \{c_1, \ldots, c_{|\mathcal{C}|}\}$;
Output: The updated $\hat{S}$ and $\hat{H}$

1: $h_i := |V| + 1$;
2: **for** $c \in \mathcal{C}$ **do**
3: $\hat{S}' := \langle v^1, \ldots, v^{i-1}, c \rangle$;
4: $\hat{H}' := \langle h_1, \ldots, h_{i-1}, h_{i-1} \rangle$;
5: $\widehat{\mathcal{T}} :=$ the subtree of $\mathcal{T}$ induced by the paths $\pi(o, v^j), v^j \in \hat{S}'$;
6: **if** $\mathcal{LP}(\langle \mathcal{R}, \widehat{\mathcal{T}} \rangle, \hat{S}', \hat{H}')$ is feasible **then**
7: $v^i := c$;
8: $h_i := h_{i-1}$;
9: **break**
10: **end if**
11: **end for**

12: **if** $h_i > h_{i-1}$ **then**
13: **for** $c \in \mathcal{C}$ **do**
14: $LB_c := h_{i-1} + 1$;
15: $UB_c := h_{i-1} + \min_{v \in \{ \bigcup_{j:h_j = h_{i-1}} \pi(o, v^j) \}} l(v, c)$;
16: $\hat{h}_c := UB_c$;
17: **while** $LB_c < UB_c$ **do**
18: $q_c := \lfloor \frac{LB_c + UB_c}{2} \rfloor$;
19: $\hat{S}' := \langle v^1, \ldots, v^{i-1}, c \rangle$;
20: $\hat{H}' := \langle h_1, \ldots, h_{i-1}, q_c \rangle$;
21: $\widehat{\mathcal{T}} :=$ the subtree of $\mathcal{T}$ induced by the paths $\pi(o, v^j), v^j \in \hat{S}'$;
22: **if** $\mathcal{LP}(\langle \mathcal{R}, \widehat{\mathcal{T}} \rangle, \hat{S}', \hat{H}')$ is feasible **then**
23: $\hat{h}_c := q_c$;
24: $UB_c := q_c - 1$;
25: **else if** $q_c \geq h_i$ **then**
26: **break**
27: **else**
28: $LB_c := q_c + 1$;
29: **end if**
30: **end while**
31: **if** $\hat{h}_c < h_i$ **then**
32: $v^i := c$;
33: $h_i := \hat{h}_c$;
34: **end if**
35: **end for**
36: **end if**

37: $\hat{S} := \langle v^1, \ldots, v^{i-1}, v^i \rangle$;
38: $\hat{H} := \langle h_1, \ldots, h_{i-1}, h_i \rangle$;
39: **return** $\hat{S}$, $\hat{H}$

Time $\hat{h}_c$ is determined through a binary search where the lower bound, LB_c, is initially set to $h_{i-1}+1$ and the upper bound, UB_c, is initially set to $h_{i-1}+\min_{v\in\{\bigcup_{j:h_j=h_{i-1}}\pi(o,v^j)\}} l(v,c)$. The specification of LB_c is an immediate consequence of the aforementioned negative result of the first part of the procedure. The specification of UB_c recognizes the fact that, at time h_{i-1} (i.e., the visitation time of the last visited node in the current partial plan, v^{i-1}), every node on a path $\pi(o,v^j)$, for a leaf node $v^j\in\hat{S}$ with $h_j=h_{i-1}$, must be occupied by a robot, and therefore, the smallest additional time to reach the considered leaf node c cannot be higher than the projection of node c on the subtree that is induced by this set of paths. The feasibility of a tentative visitation time for node c, in the interval $[LB_c,UB_c]$, is tested by formulating and solving the corresponding testing LP (c.f. Lines 18–22).

The conducted binary search starts with testing the feasibility of $\lfloor\frac{LB_c+UB_c}{2}\rfloor$ as a visitation time for the considered leaf node c, and if this time is feasible, the search restricts itself in the lower subinterval of the current search interval that is induced by the aforementioned value $\lfloor\frac{LB_c+UB_c}{2}\rfloor$, in an effort to find an even lower feasible time (c.f. Lines 22–24); otherwise, it searches for a feasible visitation time for c in the upper subinterval of the current interval that is induced by $\lfloor\frac{LB_c+UB_c}{2}\rfloor$ (c.f. Lines 27–28). Furthermore, the evaluation of the minimum possible visitation time of the candidate node c is terminated prematurely, if it is detected that this time exceeds the minimum visitation time that has been already computed for some previously evaluated node of $\mathcal{C}$ (c.f. Lines 25–26).

Every time that the current node c is found to be more competitive than the previously evaluated candidate nodes, it is recorded as the current best choice in Lines 31–34 of the procedure. Finally, upon completing the execution of its second part, procedure APPEND proceeds to its third part for reporting the results, along the lines that were described above, in the discussion of the first part.

In the next subsection, we demonstrate the execution of the presented algorithm by applying it on a small but elucidating example.

C. Example

In this subsection, we detail the execution of Algorithm 1 on an instance of the M- or the TVT-problem that is defined by the rooted tree $\mathcal{T}$ and a set of seven robots, $\mathcal{R}$, that are depicted in Figure 1(a).[6] In Figure 1(a), the root node of tree $\mathcal{T}$ is labeled by o, and the seven robots are depicted by the small blue colored circles next to o. Also, the leaf nodes of $\mathcal{T}$ are labeled from 1 to 5, and this labeling defines an ordering for the scanning of the leaf node subsets that are processed at the various steps of the algorithm. Finally, the lists $\hat{S}$ and $\hat{H}$ employed by Algorithm 1 are initially empty.

For the selection of the first node to enter list $\hat{S}$, the algorithm first computes the depth of each leaf node of $\mathcal{T}$ as presented in Figure 1(b), and picks leaf node 3 to enter $\hat{S}$, since this is the first encountered node of minimal depth in the running candidate set $\mathcal{C}$. Also, $h_1=l(o,3)=3$ enters list $\hat{H}$, and this value also defines the makespan t_{max} of the partial schedule that is specified by the current list pair $(\hat{S},\hat{H})$. In addition, Figure 1(b) highlights in yellow the subtree $\widehat{\mathcal{T}}$ that is induced by the path $\pi(o,3)$ (i.e., the path leading to the only leaf node currently in $\hat{S}$), and the projections $b(i,\widehat{\mathcal{T}})$ of the remaining leaf nodes of $\mathcal{T}$ on $\widehat{\mathcal{T}}$. Finally, path $\pi(o,3)$ is marked with robots in Figure 1(b), to indicate the fact that at the corresponding period $t_{max}=3$, all nodes of this path must be occupied by a robot.

In order to select the next leaf node to enter list $\hat{S}$ from the remaining leaf nodes $\{1,2,4,5\}$, the algorithm first thins out this candidate set by selecting for every maximal subset of this set with a common projection $b(i,\widehat{\mathcal{T}})$ in Figure 1(b), a single representative of minimal depth; the resulting candidate set is the set $\mathcal{C}=\{1,4\}$ presented in the top-right part of Figure 1(c). Subsequently, the algorithm scans this thinned candidate set and detects that node 4 can be appended to $\hat{S}$ as the next leaf node, v^2, without an increase in the makespan of the new partial schedule; i.e., $t_{\max}=3$ after this addition. Thus, node 4 enters $\hat{S}$ and the corresponding visitation time, $h_2=3$, enters list $\hat{H}$. Again, Figure 1(c) also depicts in yellow the subtree $\widehat{\mathcal{T}}$ that is induced by the paths $\pi(o,3)$ and $\pi(o,4)$, which lead to the leaf nodes that belong in $\hat{S}$, and annotates the projections of the remaining leaf nodes of $\mathcal{T}$ on $\widehat{\mathcal{T}}$. Finally, in this case, both paths $\pi(o,3)$ and $\pi(o,4)$ are indicated as occupied by robots, since the corresponding leaf nodes 3 and 4 are reached simultaneously at period $t_{max}=3$.

Using the nodal projections that are annotated in Figure 1(c), we can see that the thinned candidate set for the next node to enter list $\hat{S}$ is the set $\mathcal{C}=\{1,5\}$, that is presented at the top-right part of Figure 1(d). In this case, there does not exist any candidate node in $\mathcal{C}$ that can be visited concurrently with node 4 (i.e. without incurring an increment of the current makespan t_{max}). Hence, the algorithm executes binary searches to determine the earliest possible periods $\hat{h}_1$ and $\hat{h}_5$ for visiting, respectively, nodes 1 and 5, while observing the specified visitation times $\hat{h}_3$ and $\hat{h}_4$ (c.f. lines 12–36 in Algorithm 2). The lower bound LB_c for these binary searches is set to 4 (i.e., $t_{max}+1$). On the other hand, the upper bound UB_1 is set to the summation of the t_{max} value in Figure 1(c) and the minimum distance of node 1 from the subtree covered by robots in the same figure; hence $UB_1=3+3=6$. Similarly, the upper bound in the binary search for node 5 is $UB_5=3+2=5$. The results of these two binary searches are $\hat{h}_5=5<6=\hat{h}_1$; hence, node 5 and $h_3=5$ enter, respectively, the lists $\hat{S}$ and $\hat{H}$. Also, the makespan of the resulting partial schedule is $t_{max}=5$.

Figure 1(d) also indicates that (i) both nodes 1 and 2, that remain outside list $\hat{S}$, have the same projection on the subtree $\widehat{\mathcal{T}}$ of $\mathcal{T}$ that is induced by the nodal contents of $\hat{S}$, and (ii) among them, node 1 has the minimal depth. Hence, in the top-right part of Figure 1(e) that concerns the selection of the fourth node to enter list $\hat{S}$, we have $\mathcal{C}=\{1\}$. However, node 1 cannot be added to the current partial schedule without increasing the makespan, and therefore, in order to determine the new makespan, we need to perform another binary search with lower bound $LB_1=5+1=6$ and upper bound $UB_1=5+4=9$ (4 is the distance of node 1 from the robot-covered

[6] It is clear from the developments of Sections III-A and III-B that the considered algorithm will execute in the same manner, for, both, the M and the TVT-problem instances that are defined by any tuple $\mathcal{M}=\langle\mathcal{R},\mathcal{T}\rangle$ like that provided in Figure 1(a).

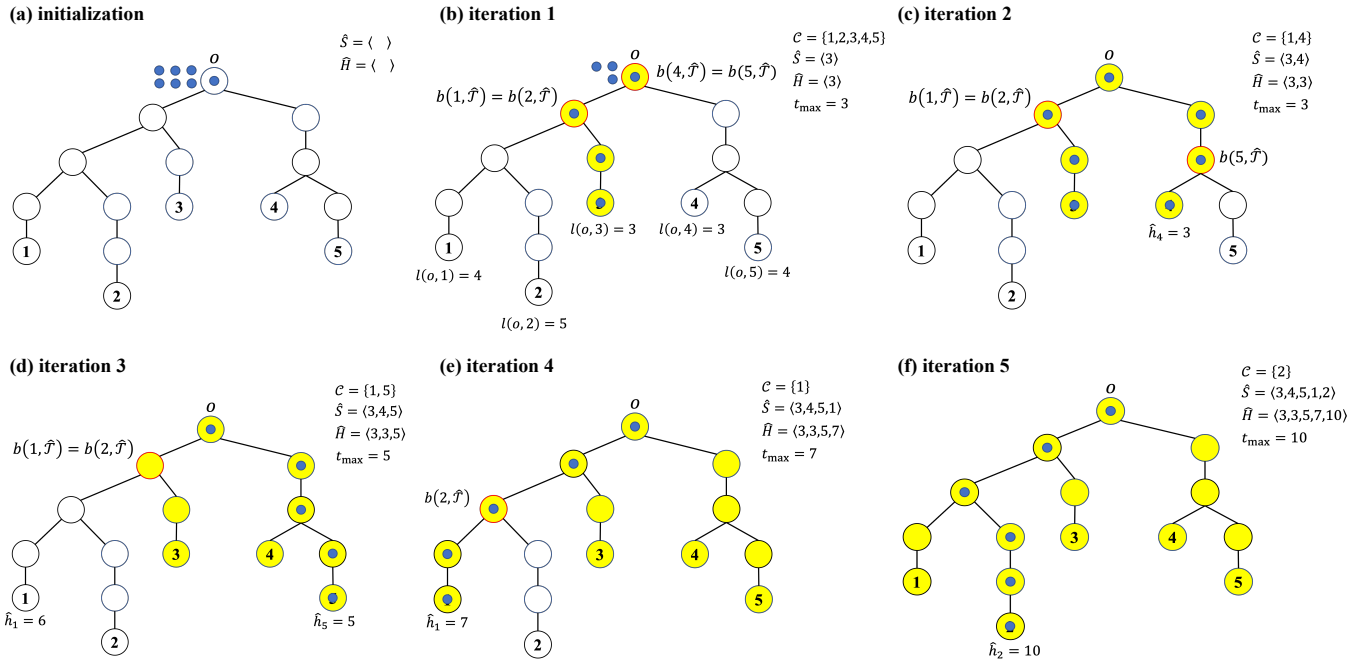

Fig. 1: An example execution of Algorithm 1.

path in Figure 1(d), which constitutes the path leading to the last visited node in the previous partial schedule). The earliest visitation visitation time for node 1 is found to be $\hat{h}_1 = 7$, and the two lists $\hat{S}$ and $\hat{H}$ are updated accordingly.

Similarly, in the last iteration, the candidate set $\mathcal{C}$ contains only leaf node 2, this node cannot be visited simultaneously with node 1 (which is the last visited node in the current partial schedule), and the distance of node 2 from the path $\pi(o,1)$ that is known to be covered by robots as a result of the construction of the previous iteration, is equal to 3. Hence, the algorithm conducts a binary search with $LB_2 = 7+1 = 8$ and $UB_2 = 7+3 = 10$. This binary search returns the earliest visitation time for node 2 as $\hat{h}_2 = 10$, and the algorithm enters node 2 and $h_5 = 10$ in the lists $\hat{S}$ and $\hat{H}$, respectively.

At this point, all leaf nodes $i = 1,\ldots,5$ have entered list $\hat{S}$, and they also have an assigned visitation time $\hat{h}_i$ in list $\hat{H}$. The algorithm now obtains a feasible flow plan F by solving the LP $\mathcal{LP}(\langle \mathcal{R}, \mathcal{T} \rangle, \hat{S}, \hat{H})$ (c.f. line 29 in Algorithm 1). Next, it converts flow plan F to an integral plan P for the original problem instance presented in Figure 1 (c.f. lines 30–34 in Algorithm 1), and it returns this plan P as its output.

The objective value of plan P for the M-problem instance that is defined by the tuple $\mathcal{M} = \langle \mathcal{R}, \mathcal{T} \rangle$ depicted in Figure 1(a), is $h_5 = 10$, since node 2 was the last node to enter list $\hat{S}$. On the other hand, the objective value of plan P for the TVT-problem instance that is defined by the tuple $\mathcal{M} = \langle \mathcal{R}, \mathcal{T} \rangle$ depicted in Figure 1(a), is $\sum_{i=1}^{5} h_i = 3+3+5+7+10 = 28$. Finally, it can be verified through the MIP formulations of [23], that, in the considered case, the performance of the obtained plan P is optimal for, both, the M- and the TVT-problem instances.

D. Complexity considerations

In this subsection, we briefly discuss the worst-case computational complexity of the presented algorithm. The empirical complexity of the algorithm, and some factors that impact this complexity, are considered in the next section, that reports a series of computational experiments with the algorithm.

We start by considering the complexity of procedure APPEND. The first part of this procedure involves the formulation and solution of $O(|\mathcal{C}|)$ LPs, with each LP involving $O(|V|^3)$ variables and constraints. Similarly, the second part of APPEND involves $O(|\mathcal{C}|)$ iterations, while each iteration involves a binary search of $O\big(\log_2(l(\mathcal{T}))\big)$ steps. Each step of the binary search involves the solution of an LP with $O(|V|^3)$ variables and constraints. Furthermore, $|\mathcal{C}| < |L|$ for every call of APPEND. Hence, every call of APPEND executes in polynomial time with respect to $|V|$.

The depths $l(o,v)$ of the nodes $v \in V$ of tree $\mathcal{T}$ that are computed in the first part of Algorithm 1, can be obtained in time $O(|V|)$ by a forward-reaching process that starts from the root node o. Subsequently, the selection of the first node, v^1, to enter the list $\hat{S}$, and the determination of the corresponding visitation time, can be performed in time $O(|L|)$ (c.f. Lines 1–9 of Algorithm 1).

The second part of Algorithm 1, corresponding to Lines 10–28, involves $O(|L|)$ iterations, with each iteration consisting of (i) the construction of the corresponding candidate set $\mathcal{C}$, and (ii) a call of procedure APPEND. The computation required for the construction of the set $\mathcal{C}$ is $O\big(|L| \cdot \min\big\{l(\mathcal{T}), |L|\big\}\big) = O(|V|^2)$. Hence, considering also the polynomial worst-case complexity of APPEND with respect to $|V|$, it follows that the second part of Algorithm 1 is also of polynomial worst-case

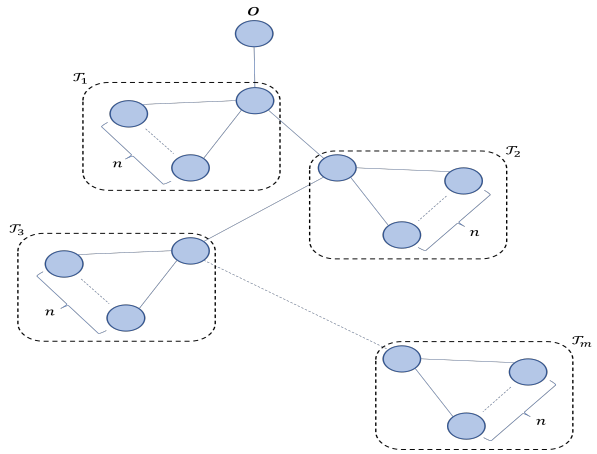

Fig. 2: The tree $\mathcal{T}$ employed in the worst-case performance analysis of Algorithm 1.

complexity with respect to $|V|$.

Finally, according to the developments of [23], the last part of Algorithm 1 is also of polynomial worst-case complexity with respect to $|V|$. Hence, the entire algorithm has polynomial worst-case complexity with respect to $|V|$.

E. Some performance considerations

In this subsection, we analyze the worst-case performance of Algorithm 1. Hence, let $\hat{S}_p := \langle v_p^1, \ldots, v_p^{|L|} \rangle$ and $\hat{H}_p := \langle h_1^p, \ldots, h_{|L|}^p \rangle$ denote the lists of the visitation sequence of the leaf nodes and the corresponding visitation times of the solution attained by Algorithm 1. We can easily notice the following relationships:

$$h_1^p \;\leq\; l(\mathcal{T}) \;\leq\; O_{opt}^M \tag{19}$$

and

$$\forall i \in \{2, \ldots, |L|\}, \quad (h_i^p - h_{i-1}^p) \;\leq\; l(\mathcal{T}) \;\leq\; O_{opt}^M \tag{20}$$

where O_{opt}^M denotes the optimal objective value for the M-problem. The makespan of the solution attained by Algorithm 1, O_p^M, is upper-bounded by $|L| \times O_{opt}^M$ as follows:

$$O_p^M \;=\; h_{|L|}^p \;=\; h_1^p \;+\; \sum_{i=2}^{|L|} (h_i^p - h_{i-1}^p) \;\leq\; |L| \times O_{opt}^M \tag{21}$$

Hence, the objective value attained by Algorithm 1 is at most $|L|$ times the optimal objective value of the M-problem instance. Similarly, for the TVT-problem, we have

$$h_1^p \;\leq\; l(\mathcal{T}) \;\leq\; O_{opt}^{TVT} \tag{22}$$

and

$$\forall i \in \{2, \ldots, |L|\}, \quad (h_i^p - h_{i-1}^p) \;\leq\; l(\mathcal{T}) \;\leq\; O_{opt}^{TVT} \tag{23}$$

where O_{opt}^{TVT} denotes the optimal objective value for the TVT-problem. Then, we can easily get an upper-bound for the solution for the TVT-problem by Algorithm 1, O_p^{TVT}, as follows:

$$O_p^{TVT} \;=\; h_1^p + \ldots + h_{|L|}^p \;\leq\; \frac{|L| \times (|L|+1)}{2} \times O_{opt}^{TVT} \tag{24}$$

Equations 21 and 24 bound the "inflation ratios" of the objective values attained by Algorithm 1, for any given M- and TVT-problem instance, in terms of the number of leaf nodes, $|L|$, of these problem instances. The dependence of these bounds on $|L|$ pronounces the important role of the leaf nodes in the specification of the pursued plans and of the logic of the presented algorithm.

In the rest of this section we show, by means of some specific instances of the M- and TVT-problems, that the inflation ratios of Equations 21 and 24 can get arbitrarily large. Figure 2 depicts a tree $\mathcal{T}$ that consists of m subtrees, $\mathcal{T}_1, \ldots, \mathcal{T}_m$. Each subtree $\mathcal{T}_i,\ i \in \{1, \ldots, m\}$, consists of one root node and n leaf nodes. We assume that the number of robots, $|\mathcal{R}|$, is equal to $2m+1$. The optimal solution of this problem instance, for both M- and TVT-problems, visits the sets of leaf nodes of every subtree in the decreasing order of the distances of the subtree's root from node o in Figure 2. In this way, while the set of leaf nodes of subtree $\mathcal{T}_i,\ i \in \{2, \ldots, m\}$, is visited, the set of leaf nodes of subtree $\mathcal{T}_j,\ j \in \{1, \ldots, i-1\}$, can be visited simultaneously. As a result, the j-th visited leaf node among the leaf nodes of subtree $\mathcal{T}_i$ is visited at period $m + (m-i) + j$. Therefore, the optimal objective value for the M-problem is

$$O_{opt}^M = m + (m-1) + n = n + 2m - 1$$

and for the TVT-problem is

$$O_{opt}^{TVT} = \sum_{i=1}^{m} \sum_{j=1}^{n} (m + (m-i) + j) = \frac{mn^2}{2} + \frac{3m^2 n}{2}.$$

On the other hand, the myopic nature of Algorithm 1 implies that the set of leaf nodes of each subtree in Figure 2 is visited in the increasing order of the distance of its root from node o in Figure 2. Furthermore, the processing of subtree $\mathcal{T}_i,\ i = 2, \ldots, m$, cannot start until subtree $\mathcal{T}_{i-1}$ has been fully processed (i.e., all of its leaf nodes have been visited). Hence, in this case, the visitation time of the j-th visited leaf node among the leaf nodes of the subtree $\mathcal{T}_i$ is $(i-1)n + i + j$, and therefore, the objective values for the M- and the TVT-problems are, respectively,

$$O_p^M = m(n+1)$$

and

$$O_p^{TVT} = \sum_{i=1}^{m} \sum_{j=1}^{n} ((i-1)n + i + j) = \frac{m^2 n^2}{2} + \frac{(m^2 + 2m)n}{2}$$

As the number of the subtree leaf nodes, n, goes to infinity,

$$\frac{O_p^M}{O_{opt}^M} \;=\; \frac{m(n+1)}{n + 2m - 1} \;\underset{(n \to \infty)}{\longrightarrow}\; m$$

and

$$\frac{O_p^{TVT}}{O_{opt}^{TVT}} \;=\; \frac{\frac{m^2 n^2}{2} + \frac{(m^2+2m)n}{2}}{\frac{mn^2}{2} + \frac{3m^2 n}{2}} \;\underset{(n \to \infty)}{\longrightarrow}\; m.$$

Hence, for the tree structure depicted in Figure 2, as $n \to \infty$, the performance degradation of the solution attained by Algorithm 1 grows with the depth of the specified tree $\mathcal{T}$.

IV. An experimental evaluation of the presented algorithm

In this section, we present an experiment that assesses the computational time and the quality of the solutions for the M- and TVT-problems generated by Algorithm 1, against the corresponding performance attained by the algorithm for these two problems that was developed in [23]. The considered problem instances and the obtained results are tabulated in Table I. In Table I, column $|V|$ reports the number of nodes of the corresponding tree $\mathcal{T}$. For the selected values of $|V|$ up to 150, we randomly generated five instances, and for each instance, we considered three levels for the number of available robots, $|\mathcal{R}|$: (i) low-level of $|\mathcal{R}|$, where $|\mathcal{R}|$ is set equal to the depth of the corresponding tree $\mathcal{T}$ plus 10% of $|V|$, (ii) high-level of $|\mathcal{R}|$, where $|\mathcal{R}|$ is set equal to the depth of $\mathcal{T}$ plus 40% of $|V|$, and (iii) the case where $|\mathcal{R}|$=$|V|$, and therefore, there is an abundance of robots. For the considered values of $|V|$ higher than 150, only one replication with a high-level of $|\mathcal{R}|$ was considered due to the very high cost of the involved computations.

For each problem instance, we first obtained a plan P by executing Algorithm 1, and subsequently, we ran the solution algorithms that were presented in [23] for the corresponding M- and TVT-problem instances. These algorithms first formulate and solve a MIP corresponding to the relaxed problem version that was introduced in Section II-B, and subsequently they convert the obtained solution to a feasible plan for the original problem instance, respectively denoted by $\tilde{P}_M$ or $\tilde{P}_{TVT}$. As explained in Section II-B, if the formulated MIP is solved to completion, then the obtained plan is optimal. However, the time required for the complete solution of these MIPs grows to prohibitively large values for larger problem instances, and therefore, their solution is terminated when a provided time budget is exhausted; in this case, the returned plan $\tilde{P}_M$ or $\tilde{P}_{TVT}$ is the plan obtained from the best generated solution for the MIP during the performed computation. In this experiment, we set the time budget for the aforementioned MIPs equal to the execution time of Algorithm 1 when applied on the corresponding problem instances.

In order to evaluate the performance of the plan P obtained through Algorithm 1 against the performance of the plans $\tilde{P}_M$ or $\tilde{P}_{TVT}$ obtained through the solution method of [23], we define "r^M" and "r^{TVT}" as follows:

$$r^M = \frac{\max_{v \in L} C(v;P)}{\max_{v \in L} C(v;\tilde{P}_M)} \tag{25}$$

$$r^{TVT} = \frac{\sum_{v \in L} C(v;P)}{\sum_{v \in L} C(v;\tilde{P}_{TVT})} \tag{26}$$

Hence, a value for r^M or r^{TVT} less (resp., greater) than 1.0 implies that the plan P returned by Algorithm 1 has a better (resp., worse) performance than the plan returned by the algorithm of [23].

For each pair of $|V|$ up to 150 and the selected levels of $|\mathcal{R}|$, we report in columns "Time limit", "r^M", and "r^{TVT}" of Table I the respective averages of the algorithm execution times and the values for r^M and r^{TVT} that were obtained in the five performed replications. In addition, Figure 3 presents more detailed scatter plots of the performance ratios r^M and r^{TVT} for each considered $|V|$ value up to 150. Finally, Table I also reports the obtained results for the instances with the higher values of $|V|$.

TABLE I: Comparing the performance of the plan obtained through the heuristic algorithm of Section III to the performance of the plans obtained through the corresponding MIP formulation of [23] under a computational time budget equal to the execution time of the heuristic algorithm.

$\|V\|$	$\|\mathcal{R}\|$-level	Exec. time (sec)	r^M	r^{TVT}
10	low-level of $\|\mathcal{R}\|$	0.2	1.0905	1.0577
	high-level of $\|\mathcal{R}\|$	0.0	1.0250	1.0000
	$\|\mathcal{R}\|$ = $\|V\|$	0.0	1.0250	1.0000
20	low-level of $\|\mathcal{R}\|$	4.6	1.0951	1.0411
	high-level of $\|\mathcal{R}\|$	4.2	1.0343	1.0250
	$\|\mathcal{R}\|$ = $\|V\|$	3.8	1.0343	1.0250
30	low-level of $\|\mathcal{R}\|$	17.8	0.8533	1.0671
	high-level of $\|\mathcal{R}\|$	16.8	1.0786	1.0454
	$\|\mathcal{R}\|$ = $\|V\|$	14.4	1.0786	1.0454
40	low-level of $\|\mathcal{R}\|$	93.6	0.9433	1.0652
	high-level of $\|\mathcal{R}\|$	65.8	1.0563	1.0485
	$\|\mathcal{R}\|$ = $\|V\|$	53.8	1.1365	1.0728
50	low-level of $\|\mathcal{R}\|$	195.4	0.8242	0.9492
	high-level of $\|\mathcal{R}\|$	169.2	0.9669	1.0092
	$\|\mathcal{R}\|$ = $\|V\|$	135.4	1.1014	1.0642
75	low-level of $\|\mathcal{R}\|$	628.0	0.7472	0.8452
	high-level of $\|\mathcal{R}\|$	619.2	1.0195	1.0149
	$\|\mathcal{R}\|$ = $\|V\|$	497.0	1.1471	1.1115
100	low-level of $\|\mathcal{R}\|$	3229.4	0.6363	0.6849
	high-level of $\|\mathcal{R}\|$	2545.6	0.8239	0.9929
	$\|\mathcal{R}\|$ = $\|V\|$	1824.8	1.0604	1.0447
125	low-level of $\|\mathcal{R}\|$	5822.2	0.5236	0.6458
	high-level of $\|\mathcal{R}\|$	5172.6	0.7156	0.8862
	$\|\mathcal{R}\|$ = $\|V\|$	4572.6	1.0695	1.0560
150	low-level of $\|\mathcal{R}\|$	11642.0	0.3080	0.5244
	high-level of $\|\mathcal{R}\|$	8915.0	0.6521	0.9951
	$\|\mathcal{R}\|$ = $\|V\|$	7721.0	0.9107	1.0043
175	high-level of $\|\mathcal{R}\|$	28924.0	0.5755	0.9092
200	high-level of $\|\mathcal{R}\|$	118845.0	0.4179	0.2260
225	high-level of $\|\mathcal{R}\|$	48808.0	0.1637	0.8497
250	high-level of $\|\mathcal{R}\|$	261819.0	0.2560	0.1495
275	high-level of $\|\mathcal{R}\|$	419513.0	0.2436	0.1563
300	high-level of $\|\mathcal{R}\|$	351630.0	0.1600	0.0904

The perusal of the values reported in Table I reveals that the algorithm of [23] tends to outperform Algorithm 1 on problem instances with smaller values of $|V|$. This is expectable since, for those problem instances, it is possible to solve the involved MIPs to completion or to near-optimal solutions within the provided time budget. On the other hand, as $|V|$ is increased to higher values, the reported ratios r^M and r^{TVT} drop below 1.0, and for the highest values of $|V|$ they get some very low values.

Furthermore, the scatter plots of Figure 3 reveal that the performance degradation brought about by the heuristic nature of Algorithm 1 when applied on smaller problem instances, is quite small (less than 30%). But the relative gains attained by this heuristic algorithm on larger problem instances are very substantial.[7] In addition, Figure 3 suggests that these gains are significantly amplified by the relative scarcity of the robots with respect to the size of the underlying tree. Since the algorithm of [23] defines the current state of art for our solution capability for the M- and TVT-problems, it can be

[7]This statement can be seen more clearly when considering the inverses of the min values for the r^M and r^{TVT} ratios reported in Table I.

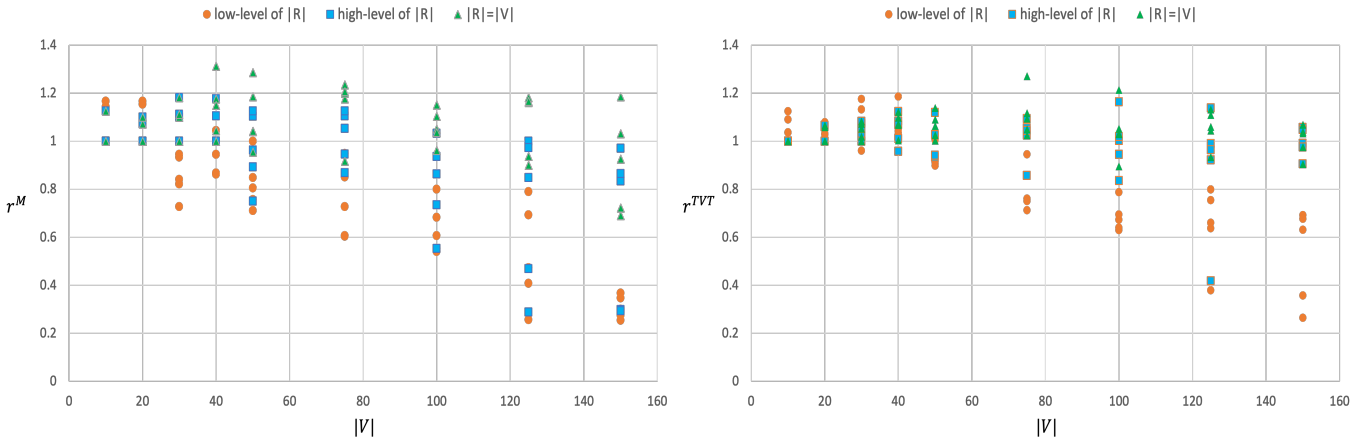

Fig. 3: Scatter plots for the performance ratios r^M and r^{TVT} for each $|V|$ value up to 150 considered in the experiment reported in Section IV; c.f. Equations 25 and 26 for the definition of these ratios.

safely concluded that Algorithm 1 constitutes an important contribution in the corresponding literature.

In the remaining part of this section, we provide some remarks regarding the empirical computational complexity of Algorithm 1. Column "Time Limit" of Table I reveals that the computational time required by Algorithm 1 can be very substantial. This time is primarily expended on the formulation and the solution of the testing LPs that are employed in procedure APPEND. More specifically, while each of these LPs requires no more than a few seconds for its solution even for the largest problem instances, the number of such LPs solved by the algorithm grows polynomially but super-linearly with respect to $|L|$. Hence, the magnitude of $|L|$ is a very significant factor determining the required execution time of Algorithm 1 on any given problem instance.

However, besides $|L|$ itself, the actual number of the testing LPs that are formulated and solved by Algorithm 1 on any given problem instance, is significantly affected by other structural attributes that are possessed by the input process instance. For example, for problem instances that allow the concurrent visitation of a large part of their leaf nodes, the execution of the calls of the procedure APPEND will be contained within the first part (Lines 1–11) of its pseudocode, and the more expensive binary search in the second part of the procedure will be avoided. On the other hand, this concurrency can be limited by (a) tree structures like those presented in Figure 4, which tend to serialize the visitation of the corresponding leaf nodes, and (b) a robot scarcity. Similarly, the distances of the leaf nodes $v \in L$ of the input tree $\mathcal{T}$ from the subtrees of $\mathcal{T}$ that are induced by the remaining leaf nodes, can be another important factor determining the execution time of Algorithm 1, since these distances affect the complexity of the binary searches that are performed by APPEND (c.f. Line 15 in Algorithm 2). All these remarks manifest the analytical richness and the intricacies that are exhibited by the considered class of the M- and TVT-problems.

Finally, we notice, for completeness, that the presented experiment was programmed in Python, executed on a laptop with i7-8850H 2.6GHz CPU and 16 GM RAM, and the formulated MIPs and LPs were solved through CPLEX.

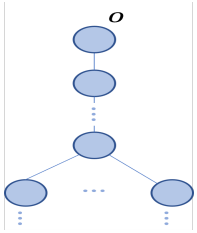

Fig. 4: A "bottleneck" structure encountered in the generated problem instances that impacts considerably the execution time of Algorithm 1.

V. Conclusions

This work has presented a new heuristic algorithm for the M- and TVT-problems that were initially introduced in [19] and further studied in [22] and [23]. The presented algorithm represents compactly a plan for the input problem instance as (i) a permutation of the leaf nodes of the underlying tree $\mathcal{T}$ together with (ii) a sequence of visitation times for these nodes that abides to the permutation. The algorithm is further facilitated by the fact that the feasibility of any tentative plan structured along the aforementioned lines can be assessed through the formulation and solution of an LP with a number of variables and constraints that are low-degree polynomials with respect to the size of $\mathcal{T}$. The numerical experimentation with the algorithm that was reported in the last part of the paper, reveals that, on larger and harder problem instances, this new algorithm outperforms significantly the corresponding algorithm in [23] and defines the current state of art regarding the solution of the M- and TVT-problems.

On the other hand, the reported experimental results also reveal that, even though the presented algorithm has polynomial worst-case complexity with respect to the size of the underlying guidepath network, the iterative solution of a large number of the LP formulations that are necessary for the evaluation of the candidate nodes considered at each iteration, can be very costly in terms of the required computational time. Hence, an important issue for future research is the substitution of the current logic of the procedure APPEND, that performs

this evaluation, with some more computationally efficient tests. This issue is part of our current investigations.

The representation of the solution space and the LP-based feasibility-assessment method of any tentative (partial) plan that are employed by the presented algorithm, can also be used in other search-based heuristic methods for the generation of good solutions for the considered problems. For instance, it is easy to envision local-search type of algorithms where the current algorithm will provide the initial solution, while the employed neighborhoods are defined by interchanges of a number of leaf nodes in the current permutation. It is also possible to devise algorithms that will employ the aforementioned techniques towards the concurrent evaluation of a number of neighboring permutations. But these algorithms will still have to solve a large number of feasibility-assessing LPs, and therefore, for larger problem instances, they will be very costly. On the other hand, the development of a more efficient version of the procedure APPEND, as suggested in the previous paragraph, can have a very significant impact in the practical implementation of such search-based heuristic algorithms.

From a more application-oriented standpoint, we can extend the applicability of the M- and TVT-problems considered in this work by allowing for (i) a more general topology for the underlying guidepath network, and (ii) the assignment of distinct target nodes to particular robots. These issues are also part of our ongoing investigations.

REFERENCES

[1] L. E. Parker, D. Rus, and G. S. Sukhatme, "Multiple mobile robot systems," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Springer, 2016, pp. 1336–1379.

[2] R. R. Murphy, S. Tadakoro, and A. Kleiner, "Disaster robotics," in *Springer Handbook of Robotics (2nd ed.)*. Springer, 2016, pp. 1577–1604.

[3] J. P. Trevelyan, W. R. Hamel, and S.-C. Kang, "Robotics in hazardous applications," in *Springer Handbook of Robotics (2nd ed.)*. Springer, 2016, pp. 1521–1548.

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

[5] N. Boysen, R. de Koster, and F. Weidinger, "Warehousing in the e-commerce era: a survey," *European Journal of Operations Research*, vol. 277, pp. 396–411, 2019.

[6] S. V. Nath, A. Dunkin, M. Chowdhary, and N. Patel, *Industrial Digital Transformation*. Packt, 2020.

[7] N. Boysen, D. Briskorn, S. Fedtke, and S. Schwerdfeger, "Drone delivery from trucks: Drone scheduling for given truck routes," *Networks*, vol. 72, pp. 506–527, 2018.

[8] A. Muralidharan and Y. Mostofi, "Communication-aware robotics: Exploiting motion for communication," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, pp. 115–139, 2021.

[9] L. Huang, M. Zhou, K. Hao, and E. Hou, "A survey of multi-robot regular and adversarial patrolling," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, pp. 894–903, 2019.

[10] M. Guo and M. M. Zavlanos, "Multirobot data gathering under buffer constraints and intermittent communication," *IEEE Trans. on Robotics*, vol. 34, pp. 1082–1097, 2018.

[11] Z. Liu, B. Wu, J. Dai, and H. Lin, "Distributed communication-aware motion planning for networked mobile robots under formal specifications," *IEEE Trans. on Control of Network Systems*, vol. 7, pp. 1801–1811, 2020.

[12] D. Tardioli, A. R. Mosteo, L. Riazuelo, J. L. Villarroel, and L. Montano, "Enforcing network connectivity in robot team missions," *IJRR*, vol. 29, pp. 460–480, 2010.

[13] S. G. Loizou and C. C. Constantinou, "Multi-robot coverage on dendritic topologies under communication constraints," in *Proceedings of IEEE CDC 2016*. IEEE, 2016, pp. –.

[14] H. Ogai and B. Bhattacharya, *Pipe Inspection Robots for Structural Health and Condition Monitoring*. New Delhi, India: Springer, 2018.

[15] C. Piciarelli, D. Avola, D. Pannone, and G. L. Foresti, "A vision-based system for internal pipeline inspection," *IEEE Trans. on Industrial Informatics*, vol. 15, pp. 3289–3299, 2019.

[16] M. Z. Ab Rashid, M. F. M. Yakub, S. A. Z. bin Shaik Salim, N. Mamat, S. M. S. M. Putra, and S. A. Roslan, "Modeling of the in-pipe inspection robot: A comprehensive review," *Ocean Engineering*, vol. 203, p. 107206, 2020.

[17] Q. Ma, G. Tian, Y. Zeng, R. Li, S. H., . Z. Wang, B. Gao, and K. Zeng, "Pipeline in-line inspection method, instrumentation and data management," *Sensors*, vol. 21, p. 3862, 2021.

[18] E. Ackerman, "Robots conquer the Underground: What DARPA's Subterranean Challenge means for the future of autonomous robots," *IEEE Spectrum*, vol. May, pp. 30–37, 2022.

[19] S. Reveliotis and Y. I. Kim, "Min-time coverage in constricted environments: Problem formulations and complexity analysis," *IEEE Trans. on Control of Network Systems*, vol. 9, pp. 172–183, 2022.

[20] L. A. Wolsey, *Integer Programming*. NY, NY: John Wiley & Sons, 1998.

[21] C. H. Papadimitriou, *Computational Complexity*. Reading, MA: Addison-Wesley, 1995.

[22] Y. I. Kim and S. Reveliotis, "Some structural results for the problem of Min-Time Coverage in Constricted Environments," in *Proc. of the 16th Workshop on Discrete Event Systems*. IFAC, 2022, pp. –.

[23] ——, "A strong combinatorial relaxation for the problem of min-time coverage in constricted environments," ISyE, Georgia Tech, Tech. Rep. (submitted for publication), 2021.

[24] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: a case study," in *Local Search in Combinatorial Optimization*, E. Aarts and K. J. Lenstra, Eds. Princeton University Press, 2003, pp. 215–310.

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

[26] E. Aarts and J. K. Lenstra, *Local Search in Combinatorial Optimization*. Princeton, NJ: Princeton University Press, 2003.

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

[28] R. L. Rardin, *Optimization in Operations Research*. Prentice Hall, 1998.

[29] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, 2013.

Spyros Reveliotis received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, the M.Sc. degree in computer systems engineering from Northeastern University in Boston, and the Ph.D. degree in industrial engineering from the University of Illinois at Urbana-Champaign.

He is a Professor with the School of Industrial and Systems Engineering, Georgia Institute of Technology, in Atlanta, GA. His main research interests are in modern control theory and its applications.

Dr. Reveliotis is an IEEE Fellow and a member of INFORMS. He has served on the editorial boards of many journals and conferences on his areas of interest, including a Senior Editorial position for the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and the position of the Editor-in-Chief of the Editorial Board at the IEEE Conference on Automation Science and Engineering (CASE). He has also been a recipient of a number of awards, including the 2014 Best Paper Award of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING.

Young-In Kim received the B.S. degree in systems management engineering in 2017 and the M.S. degree in industrial engineering in 2019 from Sungkyunkwan University, Suwon, South Korea. He is currently pursuing the Ph.D. degree in industrial and systems engineering from Georgia Institute of Technology, Atlanta, GA, USA. His research interests include modeling, scheduling and control of discrete-event dynamic systems.