Multi-Robot Patrolling with Sensing Idleness and Data Delay Objectives

Jürgen Scherer¹ · Bernhard Rinner¹

Received: 31 July 2019 / Accepted: 9 January 2020 / Published online: 3 March 2020
© The Author(s) 2020

Abstract

Multi-robot patrolling represents a fundamental problem for many monitoring and surveillance applications and has gained significant interest in recent years. In patrolling, mobile robots repeatedly travel through an environment, capture sensor data at certain sensing locations and deliver this data to the base station in a way that maximizes the changes of detection. Robots move on tours, exchange data when they meet with robots on neighboring tours and so eventually deliver data to the base station. In this paper we jointly consider two important optimization criteria of multi-robot patrolling: (i) idleness, i.e. the time between consecutive visits of sensing locations, and (ii) delay, i.e. the time between capturing data at the sensing location and its arrival at the base station. We systematically investigate the effect of the robots' moving directions along their tours and the selection of meeting points for data exchange. We prove that the problem of determining the movement directions and meeting points such that the data delay is minimized is NP-hard. For this purpose, we define a structure called tour graph which models the neighborhood of the tours defined by potential meeting points. We propose two heuristics that are based on a shortest-path-search in the tour graph. We provide a simulation study which shows that the cooperative approach can outperform an uncooperative approach where every robot delivers the captured data individually to the base station. Additionally, the experiments show that the heuristic which is computational more expensive performs slightly better on average than the less expensive heuristic in the considered scenarios.

Keywords Multi-robot systems · Mobile robotics · Patrolling · Coordination · Cooperating robots

1 Introduction

The interest in using mobile robot teams for surveillance and monitoring environments over a longer period of time has emerged with the advances in the fields of robotics, computation and communication. Examples for applications include disaster response [7, 17, 35], wildfire monitoring [12], security tasks [20], environmental monitoring [31], and exploration and mapping [24, 32]. The mobility of the robots extends their sensor coverage and allows areas to be monitored that cannot be covered efficiently with static

sensors that also have to be deployed. The drawback is that not all areas of the environment can be monitored at the same time. In certain scenarios it is not only important that certain locations of interest (referred to as sensing locations) get visited repeatedly, but also that the data captured by the sensors of the robots is transmitted to a base station in due time. This allows human mission operators to quickly assess a situation or that the collected data can be promptly processed for another purpose. There are various applications with such sensing and data delivery constraints, in particular search and rescue scenarios in remote wide areas without communication infrastructure or disaster response, where infrastructure is not available (cf. Fig. 1).

We assume that the mobile robots and the base station are equipped with wireless transceivers to exchange data as well as sufficient memory to store the data. This enables the data to travel to the base station via multiple robots in a store-and-forward fashion. Two optimization criteria are essential to this multi-robot problem: *idleness* and *delay*. The first describes the time between consecutive visits at a sensing location, and the second describes the time between

✉ Bernhard Rinner
bernhard.rinner@aau.at

Jürgen Scherer
juergen.scherer@aau.at

¹ Institute of Networked and Embedded Systems,
University of Klagenfurt, Universitätsstraße 65-67,
9020 Klagenfurt, Austria

Fig. 1 Multi robot patrolling (e.g. for security reasons, search and rescue or mapping in a disaster response missions) in a wide area using unmanned aerial vehicles (a), and inside and outside buildings (b). Rounded paths depict the tours for the robots (solid triangles), dashed lines depict potential locations for communication among the robots, the solid rectangle depicts the base station. The problems considered in this paper are the selection of locations for communication such that data captured by the robots on their tours reach the base station in a store-and-forward fashion and scheduling the robots on their tours such that they can communicate with each other

the capturing of data at a sensing location and its arrival at the base station. To optimize or constrain these criteria, coordinating the movement of the robots is necessary.

Monitoring of an environment over long time periods is related to the patrolling problem, where mobile robots continuously travel and sense the environment. While idleness is a common optimization criterion for the patrolling problem [6, 19, 21, 30], explicitly minimizing or constraining delay has experienced much less attention in literature. In contrast to most of the existing work, we focus on cooperative data transportation which eliminates the need of detours to the base station for every robot to deliver the data. This can improve idleness and allows to operate robots in environments where traveling to the base station is not possible for every robot (e.g. due to obstacles). The problem is also motivated by the results of Chevalyre [6] who stated that a cyclic strategy in which all robots patrol a common closed tour equally spaced in the same direction is not suitable for graphs with long edges, and

a partitioning of the graph in vertex disjoint subgraphs is preferable. He establishes a relationship between the idleness of the optimal solution of the cyclic strategy opt_C and the partitioning opt_P (where each robot follows a closed tour in its subgraph) with $opt_C \leq opt_P + 3 \times \max_{ij}\{c_{ij}\}$, where $\max_{ij}\{c_{ij}\}$ is the longest edge in the graph. Additionally, if the robots cannot communicate with each other or the base station while moving equally spaced along a single tour, the delay of the data at the base station is as high as the time it takes for one robot to completely traverse the tour.

Depending on the representation of the environment, determining the optimal solution to a patrolling problem can be computationally demanding. Determining the optimal tours for minimal idleness on graphs for example is related to the traveling salesperson problem (TSP) [11] and the k-TSP [10] which are both NP-complete. To decouple the complexity of path planning from planning the coordinated data transport to the base station, we assume that closed tours for each robot are given. Although scheduling robots on given tours considering some idleness criterion is a recurring problem in literature, e.g. [5, 16, 27, 28, 36], this paper performs a comprehensive complexity analysis for scheduling multi-robot tours considering idleness and delay for the first time and investigates novel cooperative scheduling heuristics.

We consider the following patrolling scenarios. A set of closed tours which can have different lengths, one for each robot, is given and the robots are only allowed to move along these tours in a certain direction. Two robots can exchange the data captured on their tours when they are at certain positions on their tours (so-called meeting points) with the aim to transport the data to the base station lying on the tour of a particular robot (cp. Fig. 2). The goal is to limit the maximum idleness to the lowest possible value determined by the tours and to minimize the delay over all sensing locations. This problem involves answering the following questions: (i) which robots should meet, (ii) in what direction should the robots move on their tours, and (iii) if there are more than one possible

Fig. 2 a Example of an abstracted multi-robot patrolling scenario with cooperative data transport to the base station at three time instances (from left to right). Positions and directions of the robots are indicated by triangles, the base station is depicted as filled square. There is a dashed line between robots if they exchange data. The robots move

along fixed tours (depicted as circles) and exchange data with robots on neighboring tours. b If robots do not exchange data and rather transport it individually to the base station, the increased tour length (a tour must include a detour towards the base station) result in a an increased idleness

meeting points between tours, at which one should robots meet. Additionally, a schedule has to be determined which describes where the robots should wait for each other (in case one robot meets with more than one other robot on its tour). We show that all three questions are NP-hard and propose a heuristic for solving this problem. The first question is related to selecting a tour tree from a tour graph (explained in Section 4) and is therefore termed *minimum delay tree* (MDT). The other two questions are related to extensions of MDT and are termed *MDT with directions* (MDTD) and *MDTD with meeting points* (MDTDM).

The contributions of this work can be summarized as follows: (i) We formulate the MDT problem and its extensions and show that they are NP-hard. (ii) We describe an algorithm that efficiently constructs a solution that has the best possible idleness with the given tours. (iii) In case only the directions of the tour traversals have to be chosen (everything else is fixed), we describe a procedure that efficiently constructs a solution that also minimizes the delay. (iv) We propose two heuristics for MDTD which select the tour tree and the directions from a tour graph. (v) Finally, we evaluate and compare the heuristics in experimental simulations.

The article is organized as follows: In Section 2 we review the existing literature. In Section 3 we introduce some notation and formulate the idleness and delay criteria. The MDT problem and its extensions are described in Section 4 and the heuristics for MDTD are presented in Section 5. In Section 6 we describe the algorithm for the online execution once a solution for MDTD has been obtained. Section 7 describes the simulation results, and Section 8 concludes the article.

2 Related Work

The multi-robot patrolling problem can be divided into the problem of determining paths in the environment as well as controlling and coordinating the robot movement along these paths. Pasqualetti et al. [28] present algorithms for the calculation of minimum idleness partitions for robots on a given chain, tree or cyclic graph. Pasqualetti et al. [27] calculate a tour in the environment containing sensing locations with different priorities, and develop a control law that coordinates the robots on that tour with the aim to minimize the weighted idleness. Kingston et al. [18] consider coordinated patrolling accounting for leaving and joining robots on a linear perimeter with dynamic length. Similarly, Acevedo et al. [2] consider robots traveling along their partition on a linear perimeter and using local coordination with their neighbors to react to changes in perimeter length, number of robots and travel speed. Smith et al. [36] develop a velocity controller for robots following

individual tours. The goal is to limit uncertainty, which is growing in the environment at different rates. Fargeas et al. [8] consider the problem of finding tours that meet idleness constraints of sensing locations and investigate in periodicity properties of these tours. Nigam et al. [26] describe an approach that converts the long term goal of minimizing the idleness into a short horizon control law that selects the next sensing location that should be visited by a robot. Chang et al. [5] describe an algorithm for tour planning, dispatching data mules on tours and controlling the speed to meet the revisit constraints of points of interest in a wireless sensor network setting.

Maintaining connectivity is a prevalent requirement for multi-robot task planning [9, 13, 14, 25, 29]. Scherer and Rinner [33, 34] consider persistent surveillance with energy constraints and forcing persistent multi-hop connectivity to a base station. Anisi et al. [3] describe the problem of minimizing the coverage time of an area with recurrent connectivity demands. Kantaros and Zavlanos [16] develop coordination algorithms for robots traveling back and forth along predefined paths between rendezvous points. A distributed controller determines the meeting times such that recurrent connectivity is guaranteed. A similar problem as described in this work is considered by Kantaros et al. [15], where robots visit task locations and schedule meetings to exchange data. In contrast to our work, the meeting robots are predefined and no delay minimization is considered.

Banfi et al. [4] formulate a MILP (mixed integer linear program) and develop heuristics for the problem of finding a patrolling path for each robot with the goal to minimize the delay. Each robot follows a path containing sensing locations and intermediate detours to communication sites where the data can be transmitted to the base station. Manyam et al. [23] present a MILP formulation and a heuristic for a similar problem with task revisit constraints. Acevedo et al. [1] consider the propagation of information among the robots in a patrolling scenario. A decentralized algorithm maintains a grid shaped partition of the area where each robot is traveling a circular path within its subarea. Robots exchange data on the border of its subarea with each robot of the neighboring subareas, which minimizes the propagation time of information in this grid shaped partition. Table 1 compares the most relevant related work.

3 Problem Formulation

We assume that the tours are closed and can have different lengths, all points on a tour are sensing locations, and the tours contain predefined meeting points for the exchange of data with robots on neighboring tours. Robots can exchange collected data if they are at the meeting point that connects

Table 1 Comparison of related multi-robot patrolling approaches

Reference	Infinite horizon	Data delay	Connectivity
[23]	✓	objective	single-hop
[4]		objective	single-hop
[9]		objective	store-and-forward
[13]		constraint	store-and-forward
[1, 2, 15, 16, 18, 28]	✓		store-and-forward
[33, 34]	✓		full
[3, 14]			recurrent
[25]			full
This work	✓	objective	store-and-forward

A check mark ‘✓’ in the column *Infinite Horizon* indicates that the approach generates a solution for an infinite time horizon (i.e. persistent surveillance/patrolling). The term ‘objective’ in *Data Delay* indicates that the work explicitly considers minimization of the delay, ‘constraint’ that the delay is considered as constraint, and an empty field that the delay is not considered at all. The meaning of the terms in the *Connectivity* column is: ‘store-and-forward’ means delay tolerant data forwarding, ‘single-hop’ that robots deliver data directly to the base station, ‘recurrent’ that all robots meet after a certain interval, and ‘full’ persistent connectivity among all robots and the base station

their tours at the same time. Since tours can have different lengths, a robot might need to wait at a meeting point for its neighbor robot to establish a data exchange. All robots move with the same unit speed in a particular direction (either clockwise or counterclockwise) but every robot can stop at any point on its tour for an arbitrary amount of time.

More precisely, given is a set $V = \{1, \dots, n\}$ of n tours for n robots. There is a one-to-one mapping between tours and robots, and we will use the same variable to identify a robot as well as the tour which it traverses. Every robot $v \in V$ moves along a tour in a particular direction d_v and with unit speed, which is the same for all robots. With each tour a real number $l_v > 0$ is associated, which is the minimum time a robot can traverse the tour completely without intermediate stops. Each point on a tour v is from a set denoted $P(v)$ and can be referred to by a local one-dimensional coordinate system which is determined by an origin on the tour and the direction of the tour. We assume that a subset of the points on a tour are sensing locations, and denote its set with $P_S(v)$ and $P^S := \bigcup_{v \in V} P_S(v)$. The position of a meeting point between tours w and v is specified along the local coordinate system as $p_w^{meet}(v)$ on tour w , and as $p_v^{meet}(w)$ on tour v . The position of a robot v at a certain instant t on its tour is denoted by $p_v(t)$. Vertex v_0 identifies the tour which has a connection to the base station at point $p_{v_0}^{BS}$.

The tours are the vertices in a tour graph $G = (V, E, v_0, l_v, time_v, l_v^S)$, with an edge between two tours in

E if they are connected with a meeting point. Actually, a meeting point identifies two different points, one on each of the two tours it connects. The function $time_v(p, q, d) : P(v) \times P(v) \times \{cw, ccw\} \rightarrow \mathbb{R}_{\geq 0}^1$ gives the minimum time for a robot to travel from meeting point p to q on tour v in clockwise or counterclockwise direction d (i.e. the distance between p and q under the unit speed assumption without intermediate stops). The function $l_v^S(p, d) : P(v) \times \{cw, ccw\} \rightarrow \mathbb{R}_{\geq 0}$ returns the minimum time any sensing location can be reached from p when moving in clockwise or counterclockwise direction. This function will allow the delay calculation after a robot starts its tour from a particular point $p \in P(v)$. In the following, we use the short notation $G = (V, E)$ for the tour graph. The tour graph is connected such that the data collected by the robots can reach a base station, which is connected to a particular tour.

To visit a sensing location $x \in P_S(r)$ at time t , a robot r must be at the position of the sensing location at time t , i.e. $p_r(t) = x$. A robot r_1 visiting a sensing location $x \in P_S(r_1)$ at time t captures and stores the observation data associated with the tuple (x, t) in its local memory. The data is forwarded by robot r_1 to another robot r_2 at time $t' \geq t$ if $p_{r_1}(t') = x_1 \in P(r_1)$, $p_{r_2}(t') = x_2 \in P(r_2)$, and $x_1 = p_{r_1}^{meet}(r_2)$ and $x_2 = p_{r_2}^{meet}(r_1)$. Robot r_2 stores the data associated with the tuple (x, t) and forwards it to any other robot it can communicate with at times $t'' \geq t'$. Finally, all data arrives at the base station $x_0 \in X$. We assume that a sensing location $x \in P^S$ is considered visited when the robot leaves x and not when it enters x . In this way it is possible to decrease the delay by deferring the data generation to the latest possible time. We will see in Section 4 that this assumption does not have an impact on the worst idleness (cp. Definition 1).

A patrolling strategy π is a mapping from instants of time to points in $P_S(r)$ for every robot r and describes when points should be visited by the robots. Two parameters are associated with a point $x \in P^S$, *instantaneous idleness* and *instantaneous delay*. The first describes the duration a point remains unvisited, and the second describes the time between the capturing of observation data and its earliest arrival at the base station. The definition of the idleness criterion is adopted from [19] and is extended by the delay criterion.

Definition 1 (Instantaneous idleness, instantaneous worst idleness, worst idleness criterion [19]) If the robots follow a strategy π , the instantaneous idleness $I_t^\pi(x) \in \mathbb{R}_{\geq 0}$ at time t of point $x \in P^S$ is the elapsed duration since the last visit of x by any robot. By convention, at initial time,

¹We denote the set of non-negative real numbers with $\mathbb{R}_{\geq 0} := \{r \in \mathbb{R} | r \geq 0\}$

$I_0^\pi(x) = 0$, for any strategy π and any $x \in P^S$. The worst idleness criterion WI^π is defined as

$$WI^\pi := \limsup_{t \rightarrow +\infty} WI_t^\pi, \quad (1)$$

where $WI_t^\pi := \max_{x \in P^S} I_t^\pi(x)$ is the instantaneous worst idleness.

Note that the definition of the instantaneous idleness considers the situation when a robot waits at a certain location, its instantaneous idleness stays zero as long as the robot is at that position.

Definition 2 (Instantaneous visit delay, instantaneous delay, instantaneous worst delay, worst delay criterion)

If the robots follow a strategy π , the instantaneous visit delay $D_t^\pi(x, t', t'')$ at time t of point $x \in P^S$ is the elapsed duration since a visit of any robot at point x that happened at time t' before the data arrives at the base station at time t'' :

$$D_t^\pi(x, t', t'') := \begin{cases} t - t', & \text{if } t' \leq t \leq t'' \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

The instantaneous delay of a point $x \in P^S$ is defined as $D_t^\pi(x) := \max_{t' \in T_x^{vis}} D_t^\pi(x, t', \min T_{(x,t')}^{arr})$, where T_x^{vis} is the set of points in time a visit at x happens, and $T_{(x,t')}^{arr}$ is the set of points in time the data associated with the tuple (x, t') arrives at the base station. The worst delay criterion WD^π is defined as

$$WD^\pi := \limsup_{t \rightarrow +\infty} WD_t^\pi, \quad (3)$$

where $WD_t^\pi := \max_{x \in P^S} D_t^\pi(x)$ is the instantaneous worst delay.

With this notation we define the MDT problem (and similar its extensions) as optimization problem

$$\min_{\pi} WD^\pi \quad (4)$$

$$\text{s.t. } WI^\pi \leq \max_{r \in R} \{l_r\} \quad (5)$$

$$\pi \in \Pi_{MDT}, \quad (6)$$

where Π_{MDT} is the set of all solutions for the MDT instance, and l_r is the minimal time for robot r to completely traverse its tour. We will show that there exists a feasible solution for every instance of the MDT problem.

4 Scheduling of Robots on Tours

In this section we discuss the problem of coordinating robots on predefined tours such that the idleness is bounded to the lowest possible value and the delay is minimized. Coordination comprises selecting the data exchange points where robots should meet and defining a travel direction for each robot on its tour. Selecting meeting points and

directions determines the route of the captured data from a sensing location to the base station and has an effect on the delay. We will introduce the basic structure which describes the problem, the *tour graph*. In the subsequent subsections we will consider different variations of this problem: (i) selecting the directions when there is a minimal number of meeting points given, i.e. the tour graph is a tour tree, (ii) selecting a minimal number of meeting points when the directions are given, i.e. selecting a tour tree in a tour graph, (iii) selecting a minimal number of meeting points as well as directions, and (iv) selecting unique meeting points between tours, i.e. selecting a tree in a tour multi-graph. We will show that the latter three problems are NP-hard.

Figure 3a shows an example of a tour graph which is a tree in this particular case (with $|V| = n = 7$). If, like in this example, the tour graph is a tree and the directions are given, the path of data from an origin to the base station can be easily reconstructed. Figure 3b shows the representation of the tour tree as directed graph which indicates the paths the data travels from each robot to the base station. Assume that robot 1 and robot 3 have just met and robot 1 has sent its collected data to robot 3. After that, robot 1 meets robot 2, and robot 2 continues collecting data as it moves along its tour to the meeting point with robot 1 again. Here, after some time it meets robot 1 again (possibly it has to wait for robot 1) and sends the new data to robot 1 which travels along its tour to the meeting point with robot 3. Robot 3 receives the data and moves to the meeting point with robot 5. At the meeting point with robot 5, it sends its own data and data received from robot 1 (and robot 4) to robot 5. Finally, robot 5 sends its own data and all data it received from robot 6 and 3 to the base station.

4.1 Selecting Directions

We consider the situation where the tour graph forms a tree $T = (V, E)$ (the problem of selecting a tour tree in a tour graph is described in subsequent subsections), i.e. there is a minimal number of meetings points such that the tour graph is connected and the data from each sensing location can travel to the base station. We will first define a structure called *schedule*, which contains the information for coordinating the robots on their tours. This information contains the start position of a robot on its tour, the direction of traversal, and the positions where a robot should stop and how long it should wait at a particular position.

Definition 3 (Schedule) Given a tour tree $T = (V, E)$, a schedule $\pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ is a set of tuples with $\pi_v = (p_v^{start}, d_v, wait_v)$. In a schedule a robot starts and stops at p_v^{start} , i.e., $p_v^{start} = p_v(0) = p_v(\tau_v)$, with $\tau_v := l_v + \sum_{p \in P(v)} wait_v(p)$. The function $wait_v : P(v) \rightarrow \mathbb{R}_{\geq 0}$ defines the waiting times for points on tour v (the waiting

Fig. 3 **a** Example of a tour graph which is a tree with $|V| = n = 7$ and $P(v) = P_S(v)$ for all $v \in V$. The tours are depicted as circles, labeled by numbers, and furnished with arrows that show the movement direction of the robots. The solid small circles are the meeting points and the small solid rectangle is the base station. The straight lines indicate the edges of the tour graph. **b** Structure of the tour tree depicted as directed graph with an arc from a child to its parent and with vertex

5 as root. **c** A repeated schedule constructed from the graph in (a). A horizontal line denotes movement of a robot and the spacing between two horizontal lines indicates that a robot does not move. A vertical, curved arrow indicates that two robots meet and exchange data. The directions of the arrows indicate the paths the captured data travels towards the base station. The two solid rectangles show the position of the base station on tour 5

positions are the positions where the function returns values > 0). $d_v \in \{cw, ccw\}$ are the traversal directions (clockwise and counterclockwise). Additionally, $p_v(t) = p_v^{start} \forall t < 0$ and $t > \tau_v$. $wait_v(p_v^{start})$ is the initial waiting time (possibly 0) starting at time 0. Every robot meets its neighbor, i.e., $\forall [v, w] \in E : \exists t \leq \tau : p_v(t) = p_w^{meet}(w)$ and $p_w(t) = p_w^{meet}(v)$, with $\tau := \max_{i \in V} \{\tau_i\}$.

The requirement that every robot meets its neighbors defined by the edges of the tree T ensures that all collected data reach the base station (at least if the schedule is repeated). We define a *repeated schedule* as an infinite horizon patrolling strategy π^+ that can be constructed from a schedule:

Definition 4 (Repeated schedule) A repeated schedule π^+ is a repetition of a schedule defined by $\pi^+ = (\pi, \bar{v}, \gamma)$, where π is a schedule, $\bar{v} \in V$, $\gamma \in \mathbb{R}_{\geq 0}$, such that

$$\forall w \in V : \tau_w \leq \tau_{\bar{v}} + \Delta_{w\bar{v}} + \gamma, \quad (7)$$

with $\Delta_{w\bar{v}} := wait_w(p_w^{start}) - wait_{\bar{v}}(p_{\bar{v}}^{start})$.

The “spacing” between two repetitions of a schedule is defined by a robot $\bar{v}$ and γ , the time between the schedules of robot $\bar{v}$. Basically, inequality (7) states that each robot has to finish its tour before it can start again in the following repetition of the schedule. Obviously, the worst idleness $WI^{\pi^+} \geq L := \max_{v \in V} \{l_v\}$, the length of the largest tour traversed by a robot.

Figure 3c shows two repetitions of a schedule of the tour tree in Fig. 3a. This repeated schedule can be defined by $(\pi, 1, 0)$ for example. Robot 1 and 3 start at their meeting point (described by the undirected edge $[1, 3]$ in the tour graph) at the same time and robot 1 moves without

intermediate stops, whereas robot 3 has to wait for robot 1 when it finished its tour. As robot 1 moves on its tour, it meets robot 2, which is waiting for robot 1 at the meeting point $[1, 2]$. Robot 2 starts to move, finishes its tour, and waits for robot 1 to meet again at the meeting point. Note that the minimum worst idleness schedule in Fig. 3c does not necessarily minimize the worst delay. For example, when robot 2 finished its tour, it has to wait for robot 1 to transmit the captured data to it, which imposes a delay on the data of robot 2 (robot 2 could have postponed its start such that no new data is captured for a certain amount of time after it transmitted its data to robot 1).

Proposition 1 A repeated schedule $\pi^+ = (\pi, \bar{v}, 0)$, with $\bar{v} := \arg \max_{v \in V} \{l_v\}$, can be constructed from any schedule π with no intermediate waiting times, i.e. $wait_v(p) = 0, \forall p \in P(v) \setminus \{p_v^{start}\}, v \in V$. The worst idleness $WI^{\pi^+} = L$.

Proof Since every robot follows its tour without intermediate stops and meets all neighbors, after a finite time $\max_{v \in V} \{l_v + wait_v(p_v^{start})\}$, all robots have returned to the starting position. A repeated schedule must fulfill the inequality $\tau_w \leq \tau_{\bar{v}} + \Delta_{w\bar{v}} + \gamma, \forall w \in V$, which can be rewritten as $wait_w(p_w^{start}) + l_w \leq wait_{\bar{v}}(p_{\bar{v}}^{start}) + L + wait_w(p_w^{start}) - wait_{\bar{v}}(p_{\bar{v}}^{start})$ (which results in $l_w \leq L$).

Since the difference between the start times at the starting positions between $\bar{v}$ and any $w \in V$ in each repetition of the schedule π is $\Delta_{w\bar{v}}$, and the difference between consecutive start times of $\bar{v}$ is L , also $WI^{\pi^+} = L$. $\square$

Restricting the waiting times to the meeting point of v with its parent to point p_v^{start} has no negative impact on the delay since waiting at any other position on the tour cannot

decrease the delay when the tour tree is given. Moreover, the data generation is deferred if $p_v^{start} \in P_S(v)$ due to the assumption described in Section 3, i.e. a sensing location is considered visited if a robot leaves this sensing location.

With a given tour tree, selecting the directions has an impact on the worst delay WD . Compared to the schedule with counterclockwise directions in the lower part of Fig. 4, the schedule with clockwise directions in the upper part results in a lower delay.

Algorithm 1 determines the schedule with directions of a given tour tree $T = (V, A)$. To identify the direction of an edge, the arc set A is used where each edge from E is directed towards the root node v_0 , which contains the base station. In Line 1 the recursive procedure $REC(v, u)$ is called. This function returns the maximum delay for a branch of a tree originating at tour v , including the path of the data on tour v to its parent u (the parent of v is the unique node u in an edge (v, u)). If v is a leaf, the direction is chosen that leads to a smaller delay when robot v starts at p_v^{start} . For tour v the procedure tests which direction results in a smaller delay on tour v given the maximum delays of the branches (Line 20 and 21). The function $time_v(p, q, d)$ returns the time it takes to travel from point p to point q on a tour v given the traversal direction d . Additionally, the procedure calculates the differences in the starting times and stores them in the variables Δ_{vw} (Line 26). These values are used to determine the starting times of the robots (Line 10).

The starting point p_w^{start} of a robot w is set to the meeting point $p_w^{meet}(v)$ with its successor v in the traversal order (Line 9). The only point where w has to wait is the meeting point with v , and the waiting time is the sum of the waiting time of the successor v and the previously calculated value Δ_{vw} (Line 10). This produces a schedule where w is waiting

for its successor v and starts moving as soon it has met v , and follows the whole tour without intermediate stops. Finally, the wait times are shifted to be positive.

Algorithm 1 Minimum delay schedule.

Input:

Tour tree $T = (V, A)$, base station vertex v_0 , position of the base station (on tour v_0) $p_{v_0}^{BS}$, meeting positions $p_v^{meet}(\cdot)$, time of tours traversal $l_v \forall v \in V$

Output:

$wait_v(\cdot)$, start positions p_v^{start} , directions $d_v \forall v \in V$

```

1:  $REC(v_0, null)$ 
2:  $p_{v_0}^{start} \leftarrow p_{v_0}^{BS}$ 
3:  $wait_{v_0}(p_{v_0}^{start}) \leftarrow 0$ 
4:  $Q \leftarrow \langle \rangle$  //empty queue (LIFO)
5:  $Q.push(v_0)$ 
6: while  $Q \neq \langle \rangle$  do
7:    $v \leftarrow Q.pop()$ 
8:   for  $w \in V$  with  $(w, v) \in A$  do
9:      $p_w^{start} \leftarrow p_v^{meet}(v)$ 
10:     $wait_w(p_w^{start}) \leftarrow wait_v(p_v^{start}) + \Delta_{vw}$ 
11:     $Q.push(w)$ 
12:  $mw \leftarrow \min_w \{wait_w(p_w^{start})\}$ 
13:  $wait_v(p_v^{start}) \leftarrow wait_v(p_v^{start}) - \min\{0, mw\} \forall v \in V$ 

14: procedure  $REC(v, u)$ 
15:   if  $v$  is leaf then
16:      $d_v \leftarrow \arg \min_{d \in \{cw, ccw\}} \{l_v - l_v^S(p_v^{start}, d)\}$ 
17:     return  $\min_{d \in \{cw, ccw\}} \{l_v - l_v^S(p_v^{start}, d)\}$ 
18:   for  $(w, v) \in A$  do
19:      $M_w \leftarrow REC(w, v)$ 
20:      $M^{cw} \leftarrow \max_{(w,v) \in A} \{M_w + time_v(p_v^{meet}(w), p_v^{meet}(u), cw)\}$ 
21:      $M^{ccw} \leftarrow \max_{(w,v) \in A} \{M_w + time_v(p_v^{meet}(w), p_v^{meet}(u), ccw)\}$ 
22:   if  $M^{cw} \leq M^{ccw}$  then
23:      $d_v \leftarrow cw$ 
24:   else
25:      $d_v \leftarrow ccw$ 
26:   for  $(w, v) \in A$  do
27:      $\Delta_{vw} \leftarrow time_v(p_v^{meet}(u), p_v^{meet}(w), d_v) - l_w$ 
28:    $M \leftarrow \min\{M^{cw}, M^{ccw}\}$ 
29:   return  $\max\{\min_{d \in \{cw, ccw\}} \{l_v - l_v^S(p_v^{start}, d_v)\}, M\}$ 

```

Fig. 4 Tour tree with $P(v) = P_S(v)$ for all $v \in V$ where all robots move in clockwise (a) or in counterclockwise (c) directions. The worst delay for clockwise directions is smaller (b) than for counterclockwise directions (d)

Proposition 2 Algorithm 1 produces a schedule π , from which a repeated schedule $\pi^+ = (\pi, \bar{v}, 0)$, with $\bar{v} := \arg \max_{v \in V} \{l_v\}$, can be constructed. The worst idleness $WI^{\pi^+} = L$. Furthermore, π^+ minimizes the worst delay.

Proof In the loop in Line 6 the start times are chosen such that every robot v meets its neighbors $w, \forall (w, v) \in A$ without intermediate stops. The worst idleness of L follows from Proposition 1.

Now we show that the algorithm produces a schedule with minimum worst delay. Note that because of the chosen starting times, all captured data on a tour of a robot travels to the base station within the same schedule (assuming no repetition). When the minimum worst-case delays M_w to node v towards the base station in a call of $\text{REC}(v, u)$ are known (which is certainly true if w is a leaf), then $\max\{\min_{d \in \{cw, ccw\}}\{l_v - l_v^S(p_v^{\text{start}}, d_v)\}, M\}$ is also the minimum worst-case delay of all data including the data captured by v until meeting position of v with its parent u towards the base station (where M is the delay of the data sent to v when v sends it to its parent u). $\square$

4.2 Selecting a Tree in the Tour Graph

Now we consider a tour graph $G = (V, E)$ instead of a tour tree. To show that the problem of determining a tree with a minimum delay schedule in a tour graph is NP-hard, we will formulate it as a decision problem $d\text{-MDT}$ and reduce the NP-complete problem 3SAT² to it. We will assume that the directions of the tours are given and formulate $d\text{-MDT}$ as follows. Given a tour graph with directions and the distances between the meeting points, and a bound B , the question is: is there a tree in the tour graph that admits a schedule with worst case delay of at most B ? The optimization problem MDT cannot be easier than the decision problem, since a solution of the optimization problem also gives an answer to the decision problem.

The construction of an $d\text{-MDT}$ instance from an arbitrary 3SAT instance is shown by means of the example $\{c_1 = \{x_1, x_2, x_3\}, c_2 = \{\bar{x}_1, \bar{x}_2, x_4\}, c_3 = \{x_2, \bar{x}_3, \bar{x}_4\}$ in Fig. 5. In the reduction a vertex appears for each variable and each clause, and a meeting point connects a variable x_i with a clause c_j if the variable appears in the clause. The position of the meeting point on x_i depends on whether the variable is complemented or not complemented in the clause. The basic idea is that for each clause c_j an edge (c_j, x_i) has to be selected such the data from each c_j can pass some x_i with a low additional delay. This selection has the interpretation that the variable x_i makes the clause evaluate to *True*. Since the result has to be a tree, a low additional delay for all clauses results in a satisfying assignment of the 3SAT instance. The details are described in the proof of the following proposition:

Proposition 3 $d\text{-MDT}$ is NP-hard.

²An instance of 3SAT consists of a set W of Boolean variables, and a set C of clauses where each clause contains exactly three literals. The literals are of the form x_i or $\bar{x}_i$ where $x_i \in W$. The question is, whether there is an assignment of values from $\{\text{True}, \text{False}\}$ to the variables such that in every clause at least one literal evaluates to *True*.

Proof Given an instance of 3SAT with variables $W = \{x_1, \dots, x_a\}$, and clauses $C = \{c_1, \dots, c_b\}$ a tour graph with the following $n = a + b + 3$ vertices is constructed:

- one vertex for every clause c_i
- one vertex for every variable x_j
- two vertices x and $\bar{x}$
- a vertex t

The direction is set arbitrary and can be the same for all the tours, $P(v) = P_S(v)$ for all v , and the following meeting points between the tours are introduced:

- On every c_i there is a meeting point with every variable x_j which appears as literal in c_i . The distances between the meeting points on c_i is $2/3$.
- On every x_i there are two meetings points with x and $\bar{x}$ with distance 1 between them on each side of the tour. The meeting points with the clauses c_j , where the variable x_i appears, are grouped such that the distance to the meeting point with x is 0 and to the meeting point with $\bar{x}$ is 1 if the variable appears as x_i in c_j , and vice versa if the variable appears as $\bar{x}_i$ in c_j .
- On each of x and $\bar{x}$ there is a meeting point with every variable x_i with distance 0 between them except for two meeting points, each with distance 1 to the meeting point with t (such that the distance between them is 2 on the other side of the tour).
- On t there are meeting points with x and $\bar{x}$ with distance 0 between them on one side of the tour and distance 1 between each of them and the meeting point with the base station on the other side of the tour (such that the distance between them is 2 on the other side).

The bound B is set to 4. Given a satisfying assignment for the variables x_i , the parent in the tree for a variable x_i is x if the variable is *True* in the assignment, or $\bar{x}$ if the variable is *False*. The parent of a clause c_j can be any x_i that appears as satisfying literal in the clause. In this way the worst case delay, which is caused by the tours c_j , is 4 (including the length of the tours). Note that the distances do not have to be 0 and 1, but sufficiently small and large, respectively. Based on these distances, the bound B has to be set accordingly.

Next, we have to show that a tree with worst case delay of 4 also determines a satisfying assignment for the 3SAT instance. We will do this by showing that the tree has to have a certain structure. First, both edges from t to x and $\bar{x}$ have to be in the tree. Otherwise, if e.g. (t, x) is not chosen, data from x to $\bar{x}$ has to pass some tour x_j which leads to a delay of $5 = 2$ (length of tour x) + 1 (on x_j) + 1 (on $\bar{x}$) + 1 (on t). Second, exactly one edge from any x_j to either x or $\bar{x}$ has to be in the tree. Choosing both edges results in a cycle containing x_j , x , $\bar{x}$, and t . If none of these edges is in the

tree, the data has to travel along a path from x_j to some c_i to some x_k and then to either x or $\bar{x}$ which causes a delay of at least $4 + 2/3 = 2$ (length of tour x_j) + $2/3$ (on c_i) + 0 (on x_k) + 1 (on x or $\bar{x}$) + 1 (on t). Finally, for every c_i exactly one edge to some x_j has to be in the tree. Because of the arrangement of the meeting points on the tours x_j , choosing the edges for tours c_i and the edges between x_j and x or $\bar{x}$ that admit a worst case delay of 4, is equivalent to finding a satisfying assignment for the 3SAT instance. $\square$

Consider the example in Fig. 5 again with the assignment $x_1 = x_2 = x_4 = \text{True}$, and $x_3 = \text{False}$. The parent of x_1, x_2 , and x_4 is x , and the parent of x_3 is $\bar{x}$. The parent of c_1 can be either x_1 or x_2 .

4.3 Selecting Directions and Meeting Points

The problem minimum delay tree with directions (MDTD) is similar to MDT with the additional problem of finding the directions. By proofing the following proposition we will show that the decision version d-MDTD is also NP-hard and present a heuristic algorithm for this problem in Section 5.

Proposition 4 *d-MDTD is NP-hard.*

Proof The proof is similar to the proof of Proposition 3. In addition to selecting a tree, the directions for traversing the tours have to be determined as well. The difference in the reduction is the arrangement of the meeting points on the tours for the variables x_j . The distance on the tour

between a meeting point c_i and x is 0 if the variable x_j does not appear as complement in clause c_j , and the distance between a meeting point c_j and $\bar{x}$ is 0 if the variable appears as complement. Figure 6 shows the construction of the reduction for the same example as in Section 4.2.

The direction of the tours except for the tours corresponding to variables can be set arbitrary. If in an assignment a variable $x_j = \text{True}$, then the direction of the corresponding tour is counterclockwise, and clockwise otherwise. Therefore, a satisfying assignment admits a tree with worst case delay of 4.

Again, to show that a tree with worst case delay of 4 also determines a satisfying assignment for the 3SAT instance, we will show that the tree has to have a certain structure. First, both edges from t to x and $\bar{x}$ have to be in the tree. Otherwise, if e.g. (t, x) is not chosen, the shortest possible path for data from x to $\bar{x}$ has to pass some tour x_j and some c_i and some x_k which leads to a delay of $4 + 2/3 = 2$ (length of tour x) + 0 (on x_j) + $2/3$ (on c_i) + 0 (on x_k) + 1 (on $\bar{x}$) + 1 (on t). Second, exactly one edge from any x_j to either x or $\bar{x}$ has to be in the tree. Choosing both edges results in a cycle containing $x_j, x, \bar{x}$, and t . If none of these edges are in the tree, the data has to travel along a path from x_j to some c_i to some x_k and then to either x or $\bar{x}$ which causes a delay of at least $4 + 2/3 = 2$ (length of tour x_j) + $2/3$ (on c_i) + 0 (on x_k) + 1 (on x or $\bar{x}$) + 1 (on t). Finally, for every c_i exactly one edge to some x_j has to be in the tree. To limit the delay for data from tours c_i to 4, the direction for the tours x_j have to be chosen accordingly. This is only possible if the 3SAT instance has a satisfying assignment. $\square$

Fig. 5 Example of a reduction from the 3SAT instance $\{c_1 = \{x_1, x_2, x_3\}, c_2 = \{\bar{x}_1, \bar{x}_2, x_4\}, c_3 = \{x_2, \bar{x}_3, \bar{x}_4\}\}$ to d-MDT. The circles represent the tours (which do not show for better readability), the connection between the tours are depicted with the named meeting points, and the direction is ccw for all tours

Fig. 6 Example of a reduction from the 3SAT instance to d-MDT (same example as in Fig. 5)

4.4 Selecting Unique Meeting Points

In case there are more than one potential meetings points between two tours, a unique set of meeting points has to be selected to obtain a tour graph (without multiple edges between two vertices). The decision problem d-MDTDM (minimum delay tree with directions and meeting points) is also NP-hard:

Proposition 5 *d-MDTDM is NP-hard.*

Proof The proof is based on a similar idea as the proofs of Proposition 3 and Proposition 4. Figure 7 shows the reduction of the 3SAT instance of the example in Fig. 5. An assignment of 3SAT selects the meeting point between x'_i and x_i : if $x_i = \text{True}$, the upper meeting point is selected (the directions of x'_i and x_i are counterclockwise), if $x_i = \text{False}$, the lower meeting point is selected (the directions of x'_i and x_i are clockwise). A satisfying assignment of 3SAT results in a delay of $17/2$. Selecting a meeting point between tours x'_i and x_i and their directions such that the delay is $17/2$, also determines a satisfying assignment for the 3SAT instance. $\square$

4.5 Approximation

We have shown that the MDTD is already NP-hard when $P(v) = P_S(v)$ for all $v \in V$ and all tours have the

Fig. 7 Example of a reduction from the 3SAT instance (same example as in Fig. 5) to the problem of selecting unique meeting points between tours d-MDTDM

same length. The formal definition allows tours containing (arbitrarily large) segments without sensing locations which can be used to derive the result that the problem can not be approximated with a constant factor unless $P = NP$. In Fig. 8 a direction gadget is shown that is inserted between x_i and c_j if there is an edge in the tour graph (see Fig. 6). This gadget allows the data to pass in one direction within a delay of 2 but causes a delay of at least $\Gamma + 1$ in the other direction and should prevent that the data from c_j travels along a path on tours c_j, x_i, c_k . Additionally, the segments on x_i which are 1 in Fig. 6 get Γ and do not contain sensing locations. Then, as before a WI of 6 gives also a solution of the 3SAT instance. Now for every α , Γ is chosen large enough, e.g. $\Gamma = 7\alpha$, and an α -approximation also results in a solution for the 3SAT instance.

A straightforward approximation for the case $P(v) = P_S(v)$ is a breadth first traversal of the tour graph to determine a tour tree which is the union of the shortest paths from each vertex to the base station vertex (cf. Algorithm 2). Since $L = \max_{v \in V} \{l_v\}$ is a lower bound for the optimal worst delay WD_{OPT} , the worst delay WD_{SP} of a breadth first traversal starting from the base station tour cannot be worse than $depth_{SP}(G) \cdot WD_{OPT}$, where $depth_{SP}(G) := \max_{v \in V} \{dist_G(v, v_0)\}$ is the maximum length of all shortest paths in the (unweighted) tour graph from the tours to the base station tour. For example, $depth_{SP}(G) = 3$ which is the length of the path from tour 2 to tour 5 in the tour graph in Fig. 9a.

5 Heuristics for MDTD

We present two heuristics for MDTD that select a tree in a tour graph and the directions for the tours. The first algorithm (MDTD-SP) determines a tree from the union of the shortest paths from all vertices to the base station vertex. The rationale behind this idea is to minimize the longest path in the tour graph in terms of the number of tours the generated data passes. This heuristic is shown in Algorithm 2.

Fig. 8 Direction gadget for d-MDTD

Fig. 9 Example of a tour graph $G = (V, E)$ (a) and the converted graph $G' = (V', E', W)$ (b). The meeting points of the original graph are the vertices V' in the new graph and the lengths w of the edges in E' between the vertices in V' have the minimum travel times of the segments of the original tours. If there are two edges between two meeting points, the longer one is discarded

The second algorithm (MDTD-CG) is shown in Algorithm 3 and requires a converted graph $G' = (V', E', W)$ which is constructed from a tour graph $G = (V, E)$. The vertices V' of the converted graph contain the meeting points, i.e., if there is an edge $[k, l] \in E$, then there is a vertex $v_{kl} \in V'$. The lengths W of the edges E' between vertices in V' are the lengths of the segments of the tours in V . Figure 9 depicts an example of a tour graph and its converted graph. The idea behind this algorithm is to minimize the longest path that data actually travels on a path to the base station.

The algorithm determines the shortest path from every vertex in v_{kl} (representing a meeting point between tours k and l) to the base station v_{0x} . The function $dist_G(s, d)$ returns the length of the shortest path from vertex s to vertex d in a weighted graph G . This path represents a path for the data in the original tour graph for both tours k and l (in $path_k$ and $path_l$) and is stored together with the length of the path in G' (in len_k and len_l) only if it is shorter than the shortest paths that have already been found for tours k and l (see the loop starting at Line 3). When this loop has

Algorithm 2 Heuristic for MDTD (MDTD-SP).

Input:

Tour graph $G = (V, E, v_0, l_v, time_v, l_v^S)$

Output:

Tour tree $T = (V, A)$, directions $d_v \forall v \in V$

- 1: $p \leftarrow shortest_path_G(v_0)$ /* Dijkstra: all shortest paths p_v from each $v \in V$ to v_0 */
- 2: $A \leftarrow \bigcup_{v \in V} p_v$
- 3: /* Call Algorithm 1 (returns directions $d_v \forall v \in V$): */
- 4: $(d_1, \dots, d_n) \leftarrow MINIMUM_DELAY_SCHEDULE(G, A)$

terminated, the shortest paths in G' for every tour $v \in V$ have been found. Note that $\max_{v \in V} (len_v + l_v)$ is a lower bound on the worst delay WD .

Next, the branches of the tree T are added to A (loop in Line 12) in descending order of $len_i + del_i$. This is shown in Fig. 10 by means of the example of Fig. 9. Only tours that are not part of the tree yet are added in each iteration. When all tours are part of the tree T , the algorithm stops.

Basically, in Line 17 the algorithm checks if a tour has been left and adds the appropriate arc to the arc set U . If a tour is already in the tree T , the path loop exits (Line 16), and the algorithm continues with the next tour in V . The direction of the tour m which has been left depends on the order of meeting points v_{rs} and its successor on $path_i$ on the tour m (Line 23).

Algorithm 3 Heuristic for MDTD (MDTD-CG).

Input:

Tour graph $G = (V, E)$, converted graph $G' = (V', E', W)$ with edge lengths $W(e), \forall e \in E'$, base station v_{0x}

Output:

Tour tree $T = (V, A)$, directions $d_v \forall v \in V$

- 1: $len_i \leftarrow \infty, \forall i \in V$
- 2: $p \leftarrow shortest_path_{G'}(v_{0x})$
- 3: **for** $v_{kl} \in V'$ **do**
- 4: **if** $dist_{G'}(v_{kl}, v_{0x}) < len_k$ **then**
- 5: $len_k \leftarrow dist_{G'}(v_{kl}, v_{0x})$
- 6: $path_k \leftarrow p_{v_{kl}}$
- 7: **if** $dist_{G'}(v_{kl}, v_{0x}) < len_l$ **then**
- 8: $len_l \leftarrow dist_{G'}(v_{kl}, v_{0x})$
- 9: $path_l \leftarrow p_{v_{kl}}$
- 10: $A \leftarrow \emptyset, r \leftarrow null, s \leftarrow null$
- 11: $del_i \leftarrow \min_{d \in \{cw, ccw\}} \{l_i - l_i^S(path_i(1), d)\}$
- 12: **for** $i \in V$ in descending order of $(len_i + del_i)$ **do**
- 13: $U \leftarrow \emptyset, m \leftarrow i$
- 14: **for** each v_{kl} on path $path_i$ to v_{0x} **do**
- 15: **if** m is already in T **then**
- 16: break
- 17: **if** $(m \neq k)$ and $(m \neq l)$ **then**
- 18: **if** $m = r$ **then**
- 19: $U \leftarrow U \cup \{(m, s)\}, m \leftarrow s$
- 20: **else**
- 21: **if** $m = s$ **then**
- 22: $U \leftarrow U \cup \{(m, r)\}, m \leftarrow r$
- 23: $d_m \leftarrow$ determine direction based on order of v_{rs} and its successor on $path_i$ on tour m
- 24: $v_{rs} \leftarrow v_{kl}$
- 25: $A \leftarrow A \cup U$

Proposition 6 Let WD_{SP} and WD_{CG} be the worst delay of a tree determined by MDTD-SP and MDTD-CG, respectively. Then, for every $\alpha > 0$ there are instances of tour graphs such that $WD_{SP}/WD_{CG} > \alpha$.

Fig. 10 Two steps of the tree generation of Algorithm 3. First, the path in G' starting from 27 is considered (bold edges in (a)), which results in the path (2, 7, 3, 6, 5) in G and the directions $d_7 = d_6 = cw$, $d_3 = d_5 = ccw$ (solid lines in (c)). Next, the path starting at 14 is considered (b), which results in the path (1, 4, 3, 5) in G . Since 3 is already in the tree, only the branch (1, 4, 3) with $d_4 = ccw$ is added (d), the dashed line indicates the discarded part of the path). After this step the tree contains all tours

Proof Consider a tour graph with a chain $[v_0, v_1], [v_1, v_2], \dots, [v_{k-1}, v_k]$ of large tours of length Γ where each tour is connected with an arm of small tours to v_0 (see Fig. 11). Each tour on an arm has length ϵ and each arm has at least k tours. The meeting points on the chain of the large tours are on the opposite sides of the tours, i.e. $time_{v_i}(p_{v_i}^{meet}(v_{i-1}), p_{v_i}^{meet}(v_{i+1})) = \Gamma/2$. Then MDTD-SP will result in a tree where all large tours are in a chain. If Γ/ϵ is large enough, MDTD-CG will create a tree where each large tour is connected with its arm to v_0 . If $k > 2\alpha$, then $WD_{SP}/WD_{CG} > \alpha$. $\square$

Fig. 11 Example of a tour graph (size of the circle indicates the length l_v)

6 Online Execution

Once the tree (Section 5), the directions and the schedule (Section 4.1) have been determined, the robots have to execute this schedule. If the robots need to be deployed in the environment, they have to move to their tours from a deploy site. Therefore, the schedule determined by Algorithm 1 has to be reached by the robots from an initial state. We assume that each robot navigates along some path in the environment to the meeting point with its parent at the beginning of the mission and reaches this position after some time. The algorithm for the online execution is described in the following subsection.

6.1 State Machine

Algorithm 4 is running on every robot v and resembles a state machine where the variable *state* can take one of the states $\{INIT, AT_WAIT, MOVING\}$. The robot is in initial state *INIT* as long as it is moving from the initial position to the meeting point with its parent on its tour (p_v^{start}). In state *AT_WAIT* it is waiting for its parent on p_v^{start} , and in state *MOVING* it is moving along its tour. The input to the algorithm is the schedule (in particular Δ_{uv} determined by Algorithm 1) and the output are commands *Move* and *Stop* for the motion actuators. We will show that Algorithm 4 will execute an infinite sequence of state transitions $INIT, AT_WAIT, MOVING, AT_WAIT, \dots$ and that the schedule will converge to the optimal schedule after a finite time. A state transition A, B means that variable *state* changes from $state = A$ to $state = B$. Because of the assumption that every robot v will reach p_v^{start} , a transition from $state = INIT$ to $state = AT_WAIT$ will eventually occur for every robot.

Proposition 7 A robot never has to wait for an infinite time. This implies an infinite sequence of state transitions $AT_WAIT, MOVING, \dots$ for each robot.

Proof The situations when a robot has to wait infinitely long is in state *AT_WAIT* when condition $p_u(t) = p_u^{meet}(v)$ (waiting for the parent) never holds or in state *MOVING* when condition $p_w(t) = p_w^{meet}(v)$ (waiting for a child) never holds. Since the meeting points define a tour tree $T = (V, A)$, it is sufficient to show that no robot has to wait for its parent infinitely long. After a robot has met its parent, it is waiting for a finite time (line 19) and traverses its tour. The proof is by induction on the number of robots in the tree. In the base case only the robot v which has the base station on its tour is in the tree and the condition $p_0(t) = p_0^{meet}(v)$ always holds (0 is the base station and the parent of v). In the inductive step a tour v is added to the

tree. Since its parent u does not have to wait for its parent and starts its tour after a finite waiting time, also v meets u at p_v^{start} (when $p_u(t) = p_u^{meet}(v)$). $\square$

Proposition 8 After a finite number of state transitions for each robot from *MOVING* to *AT_WAIT* the schedule has converged to the schedule determined by Algorithm 1, i.e. the deviation from the schedule Δt stays 0.

Proof Consider a tour v with largest distance from the base station tour in the tour tree which has only leaves as children. After v met its parent it starts traversing the tour and possibly has to wait for children to reach the meeting point. After the first traversal (state transition from *MOVING* to *AT_WAIT*) of the tour all children started their tour and had enough time to finish their tour and to reach the meeting position with v on the second traversal of v . Therefore, Δt will be 0 for v after the second traversal. The same holds for the parent u of v after an additional state transition from *MOVING* to *AT_WAIT* of u . This argument can be repeated until the base station tour is reached. $\square$

Figure 12 shows the startup phase and execution of the state machine for the given example tour tree. After the schedule has emerged, robot 5 arrives late at p_v^{start} (short horizontal line at robot 5 indicates no progress on its tour) and the schedule emerges again.

Open issues not considered in this work are robot failures and limited energy. A straight forward approach to these problems could be that a robot on a tour that is closer to the base station in the tour tree moves up to the neighboring tour with a larger distance to the base station if it does not encounter the other robot at the meeting point after some waiting period. Similarly, a spare robot starts at the base

station if the robot at the root of the tour tree does not appear at the meeting point with the base station. In this way robots move up the tree along the unique paths towards the leaves and the system does not stall completely due to a failure or energy depletion of robots.

Algorithm 4 State machine of robot v .

Input:

$p_v^{start}, p_v^{meet}(w) \forall w$ with $(w, v) \in A, \Delta_{uv}$ for $(v, u) \in A$,
initially $state = INIT$

Output:

actuator commands $\{Move, Stop\}$

```

1: switch state do
2: case INIT
3:   if  $p_v(t) = p_v^{start}$  then
4:      $\Delta t \leftarrow 0$ 
5:     state  $\leftarrow$  AT_WAIT
6:   Stop
7: else
8:   Move to  $p_v^{start}$ 
9: case AT_WAIT
10:  if  $p_u(t) = p_u^{meet}(v)$  then
11:    Wait for  $\max\{\Delta_{uv} - \Delta t, 0\}$ 
12:     $\Delta t \leftarrow 0$ 
13:    state  $\leftarrow$  MOVING
14:    Move on tour
15: case MOVING
16:  if  $p_v(t) = p_v^{meet}(w)$  then
17:    Stop
18:    time  $\leftarrow$  current_time()
19:    Wait until  $p_w(t) = p_w^{meet}(v)$ 
20:     $\Delta t \leftarrow \Delta t + (\text{current\_time}() - \text{time})$ 
21:    Move on tour
22: else
23:   if  $p_v(t) = p_v^{start}$  then
24:     state  $\leftarrow$  AT_WAIT
25:   Stop

```

Fig. 12 Example of startup phase and execution of the state machine. **a** Tour tree with tours of equal length and cw direction for all robots. **b** Position $p_v(t)$ over t for robots 1 to 5 where each bottom line indicates p_v^{start} . The small numbers on the vertical axis indicate the position

of the meeting points on tour v . A small dot indicates when robot v reaches p_v^{start} , i.e., the state transition from *INIT* to *AT_WAIT*, and the small triangle indicates the time when the schedule has converged

Fig. 13 Worst idleness WI (left column) and worst delay WD (right column) for MDTD-CG/SP/opt and single-hop approach with varying number of robots n for scenarios with 20×40 (top row), 20×60 (center row), and 20×80 (bottom row) sensing locations. The plots show the average values and the standard deviations (the latter values are very small for some plots)

7 Simulation Results

In this section we describe the results from simulation experiments with the aim to assess the performance of the heuristics (MDTD-CG/SP) in terms of worst idleness WI and worst delay WD in different settings (e.g. size of the area, number of robots). To assess the effect of robots cooperating for the data transportation, we compare MDTD-CG/SP with an approach where the data is not transported via other robots to the base station but directly by the robot which captures the data (this approach is denoted as single-hop approach).

The environment is modeled as rectangular grid of cells of unit size, and time is discretized into time steps. A robot can move from one cell of the grid to one of the 8 neighboring cells or stay at the same cell within one time step. The communication range R^{com} (measured in number of cells) determines which cells are within communication range. The base station is in the cell at the lower left corner.

A genetic algorithm implementation³ is used to determine a tour through all sensing locations and the base station. To obtain the individual tours for the robots, the tour is split with k-SPLITOUR [10].

For MDTD, meeting points from a set of potential meeting points between each pair of tours have to be selected. In Section 4.2 we have considered selecting the edges (corresponding to meeting points) in the tour graph such that the resulting graph is a tree, whereas here we are concerned with the selection of one from possible multiple edges between two vertices in the tour graph (cf. Section 4.4). For a certain R^{com} , a potential meeting point between two tours is a pair of cells on the two tours within the communication range. For the selection of meetings points, tours are traversed in a breadth-first order starting at the tour which is connected to the base station. The

³Matlab function `tsp_ga` from Joseph Kirk at <https://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm>

tours are added in the traversal order to a converted graph (cf. Section 5), where the vertices are the meeting points selected so far. For every potential meeting point of v with a neighboring tour v' , the shortest path to the base station on the converted graph is calculated, and the meeting point with the shortest path is selected as meeting point between v and v' . This heuristic tries to shift meeting points as close as possible to the base station in the converted graph.

First, we compare the performance of MDTD-CG/SP with a single-hop algorithm similar to the one in [4] where robots make detours to communication sites (places in communication range to the base station) to individually transmit the data to the base station. An increasing number of detours is inserted in a pre-calculated tour for each UAV until the total travel distance exceeds a certain travel budget. The heuristic of [4] tries to minimize the average delay and is not well suited for minimizing the worst delay. To minimize the worst delay, we adopt this approach such that an increasing number of detours to the base station, which are evenly spread along a robot's tour (using k-SPLITOUR), are inserted until a certain total tour length of a robot is exceeded. This bound for the tour length is set to the maximum of the worst idleness resulting from MDTD-CG/SP and the maximum tour length (including the base station) for each robot, such that every robot is able to transmit the data from its tour to the base station at least once. The results for WI and WD for different number of robots n on different grids (only neighboring cells are within communication range, i.e. $R^{com} = 1$) with an area of 20×40 , 20×60 and 20×80 sensing locations are shown in Fig. 13. We apply the genetic algorithm 10 times on each environment, which results in a set of 10 different tours through all sensing locations due to the stochastic nature of the genetic algorithm. For each n , the algorithms are applied to all 10 tours, and the mean and standard deviation of the 10 resulting values are shown in the graphs.

Figure 13 also shows the worst delay WD of the optimal solutions for the MDT instances (only for the 20×40 and 20×60 grids and up to 18 robots due to the large computation times) generated with the state-of-the-art IP solver Gurobi⁴ (an MDT instance is defined by the tours and the meeting points, see Appendix A for the MILP formulation). Note that the WI is the same for MDTD-CG/SP and MDTD (opt).

Figure 13 shows that on one hand MDTD-CG/SP/opt can outperform the single-hop approach in terms of WI . The value for WI is the same for all three algorithms since all use the same tours. In the single-hop approach a robot from a more distant subarea has a long path to the base station, which causes a large WI . On the other hand, the single-hop approach can outperform MDTD-CG/SP/opt

Fig. 14 Sum of the traveled distances (number of steps in the grid) for MDTD-CG/SP and Single-hop approach with varying number of robots n for scenarios with 20×40 (top), 20×60 (center), and 20×80 (bottom) sensing locations. The time windows within the distances have been calculated is WI achieved by MDTD-CG/SP. The plots show the average values. Due to the small values, the standard deviations are omitted

in terms of WD because we have chosen a very short communication range, the tours are close to each other and the data travels along the shortest possible path to the base station. Therefore, the WD of the single-hop strategy can be seen as a lower bound for given tours in this scenario. The benefit of a lower WD for the single-hop strategy vanishes in the presence of longer (minimal) distances between tours such that the transfer of the data over the wireless link takes less time than the transport on a robot (see the discussion of the results below for the scenario in Fig. 15).

Figure 14 shows the sum of the traveled distances over all robots for MDTD-CG/SP and the single-hop approach. Since the sum of the traveled distances grows unbounded with time, the time window for the calculation is limited to the WI achieved by MDTD-CG/SP. The reason is that

Fig. 15 A scenario (75×25 cells) with a regular pattern of sensing locations for comparison of MDTD-CG/SP with a single-hop approach. The rectangles with rounded edges show the tours for optimal WI for 24 robots, each having a total length of 20 cells. The dashed lines show possible meeting points for MDTD with $R^{com} = 5$ cells, which is the distance between the tours. With an increasing R^{com} also non-adjacent tours can be connected with a meeting point

⁴<http://www.gurobi.com/>

Fig. 16 Worst idleness WI and worst delay WD of MDTD-CG/SP (the lower values for MDTD-CG and MDTD-SP have been recorded) and the single-hop approach for varying communication range R^{com} for the scenario in Fig. 15

for MDTD-CG/SP every sensing location gets visited once within this window. Since every robot is constantly moving in the single-hop approach, the sum of the traveled distances is higher than for MDTD-CG/SP where robots might have to wait at meeting positions.

There are situations for which the single-hop approach performs arbitrarily bad in terms of WI , e.g., the delay is unbounded if it is not possible for each robot to travel to the base station due to obstacles or when the maximum distance between a sensing location and the base station is arbitrary large. Figure 15 shows a scenario for 24 robots where the optimal WI can be achieved by dividing the area into a regular pattern of tours, such that each tour has the same length. A comparison between MDTD-CG/SP (the smaller value of both algorithms is recorded) and the single-hop approach for different communication ranges R^{com} is shown in Fig. 16. The WI and WD for the single-hop approach is determined by the path between the upper right sensing location and the closest location within the communication range of the base station, which is the path a robot is moving back and forth in the single-hop approach. In this scenario MDTD-CG/SP results in lower WI and WD . On one hand, we have seen that the single-hop approach can outperform the MDTD heuristics in terms of WD (Fig. 13), because it is able to achieve the lower bound for WD in the considered situations (very short communication range). On the other hand, the single-hop approach cannot perform better than the MDTD heuristics in terms of WI , because the MDTD heuristics are able to decrease the traveled distances by omitting detours to the base station for communication. In

the worst case, depending on the scenario, WI is the same for the cooperative and uncooperative approaches.

The average computation times for different number of robots is shown in Table 2. For MDTD-CG/SP a single core and for MDTD (opt) all 8 logical cores of a machine with an Intel Core-i7 6700K and 32GB of RAM were used. The instances for MDTD (opt) are the same as for Fig. 13 (20×60 sensing locations). The instances (tour graphs) for MDTD-CG/SP have been randomly generated (10 tour graphs for each n), with edge probability of 0.25 between tours and randomly sampled meeting point distances on a tour.

8 Conclusion

Multi-robot patrolling is an important application of multi-robot systems. In certain patrolling situations it is not only important that sensing locations get visited repeatedly but also that the data reaches a base station on time for further processing or for an assessment by mission operators. This is typically required in disaster response scenarios where the mission operators need an up-to-date view of the situation. We presented a multi-robot patrolling problem with cooperative data transport to the base station where robots move on predefined tours which eliminates the need for every robot to return to the base station for data delivery. MDT represents the problem of minimizing the data delay which is decoupled from path planning. In spite of its simple definition we proved that MDT is a NP-hard problem. Explicitly minimizing delay for patrolling with cooperative multi-robot data transport has not been investigated so far to the best of our knowledge. We presented heuristics and an algorithm for online execution and evaluated the performance in simulation experiments. The comparison of MDT with an uncooperative single-hop approach (every robot individually transports the data to the base station) on predefined tours shows that the cooperative approach can outperform the uncooperative approach in terms of WI and the traveled distances. The reason is, that the robots that

Table 2 Average computation times (sec) for different number of robots n for MDTD-CG/SP/opt

n	50	100	150	200	250	300
MDTD-SP	0	0	0	0	0	0
MDTD-CG	0	0	1	3	8	18
n	8	10	12	14	16	18
MDTD (opt)	8	28	111	565	1625	9704

handover the data to other robots (in this way the data finally reaches the base station) can continue patrolling their tours, while in the uncooperative approach, robots are forced to leave their tour to move to the base station.

The problem relies on TSP tours (and subtours derived from the TSP tours) through all sensing locations. In our work they are generated with traditional algorithms that try to minimize the length of the tour (and minimize the maximum length of the subtours). An open issue is the generation of such tours that support the joint minimization of idleness and delay. Other open questions are whether there are approximation algorithms with guaranteed bounds and whether some instance classes (e.g. planar graphs) can be solved optimally in polynomial time. Further extensions are the consideration of strategies for robot failures and limited operation time of robots due to energy constraints.

Funding Information Open access funding provided by University of Klagenfurt.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

Appendix A: MILP formulation of MDTD

The mixed integer linear programming (MILP) model of MDTD is based on a multi-commodity flow formulation for trees on a graph $G = (V, A)$ with $n+1$ vertices V (including a vertex 0 for a virtual base station tour) and arc set A [22]. The base station is the source of a commodity flow f_e^c for each vertex (constraint (8)). A flow of commodity c represents the path of the data from robot c towards the base station (though the flow originates at the base station in this formulation). For each vertex the sum of incoming flows is equal to the sum of outgoing flows for each commodity not dedicated to that vertex (constraint (9)), and each vertex c consumes the commodity of type c (constraint (10)). There can be only a flow on an edge if this edge is selected in the tree (constraint (11)) and the sum of the edges must be n (constraint (12)).

$$\sum_{(v,0) \in A} f_{v0}^c - \sum_{(0,v) \in A} f_{0v}^c = -1 \quad \forall c \in V \setminus \{0\} \quad (8)$$

$$\sum_{(w,v) \in A} f_{wv}^c - \sum_{(v,w) \in A} f_{vw}^c = 0 \quad \forall v \in V \setminus \{0, c\}, \forall c \in V \quad (9)$$

$$\sum_{(w,c) \in A} f_{wc}^c - \sum_{(c,w) \in A} f_{cw}^c = 1 \quad \forall c \in V \setminus \{0\} \quad (10)$$

$$f_e^c \leq x_e \quad \forall e \in A, \forall c \in V \setminus \{0\} \quad (11)$$

$$\sum_{e \in A} x_e = n \quad (12)$$

$$x_e \in \{0, 1\} \quad (13)$$

$$f_e^c \geq 0 \quad \forall e \in A, \forall c \in V \setminus \{0\} \quad (14)$$

The data which robot j gets at the meeting point between i and j and is forwarded at meeting point between j and k has to travel the distance $l_{ik}^{j,ccw}$ or $l_{ik}^{j,cw}$ on tour j , depending on the direction robot j traverses its tour. Therefore, two flow variables f_{ij}^c and f_{jk}^c are involved in the cost calculation in constraint (15) for data originating from c and traversing the tour j . The separation of the flows in this formulation allows the definition of a min-max objective. For each commodity c , z_c models the delay of data originating at robot c and the objective is to minimize z . The decision variables u_j^{ccw} and u_j^{cw} determine the direction robot j traverses its tour.

$$\begin{aligned} z_c = & u_j^{ccw} \sum_{(j,c) \in A} f_{jc}^c (l_c - l_c^S(p_c^{meet}(j), ccw)) + \\ & u_j^{cw} \sum_{(j,c) \in A} f_{jc}^c (l_c - l_c^S(p_c^{meet}(j), cw)) + \\ & \sum_{(i,j),(j,k) \in A} f_{ij}^c f_{jk}^c u_j^{ccw} l_{ik}^{j,ccw} + f_{ij}^c f_{jk}^c u_j^{cw} l_{ik}^{j,cw} \quad \forall c \in V \setminus \{0\} \end{aligned} \quad (15)$$

$$z_c \leq z \quad \forall c \in V \setminus \{0\} \quad (16)$$

$$u_j^{ccw} + u_j^{cw} = 1 \quad \forall j \in V \setminus \{0\} \quad (17)$$

$$u_j^{ccw}, u_j^{cw} \in \{0, 1\} \quad \forall j \in V \setminus \{0\} \quad (18)$$

The products, e.g. $f_{ij}^c f_{jk}^c u_j^{ccw}$, can be linearized (likewise $f_{ij}^c f_{jk}^c u_j^{cw}$) with an additional variable $f_{ijk}^{c,ccw}$ and the constraints:

$$f_{ijk}^{c,ccw} \leq f_{ij}^c \quad (19)$$

$$f_{ijk}^{c,ccw} \leq f_{jk}^c \quad (20)$$

$$f_{ijk}^{c,ccw} \leq u_j^{ccw} \quad (21)$$

$$f_{ijk}^{c,ccw} \geq f_{ij}^c + f_{jk}^c + u_j^{ccw} - 2 \quad (22)$$

Appendix B: List of symbols

Symbol	Meaning
X	set of points of environment
P^S	sensing locations
Y	communication relation
$R = \{1, \dots, n\}$	set of n robots/tours
$\pi \in \Pi$	patrolling strategy/schedule (from the set of all strategies Π)
π^+	repeated schedule (repetition of schedule π)
$\mathbb{R}_{\geq 0}$	set of real numbers larger or equal 0

$I_t^\pi(x)$	instantaneous idleness of x at time t (using π)
$D_t^\pi(x, t', t'')$	instantaneous delay of x at time t (using π)
$WI_t^\pi(x)$	instantaneous worst idleness at time t (using π)
$WD_t^\pi(x)$	instantaneous worst delay at time t (using π)
WI, WD	worst idleness, worst delay
$G = (V, E)$	(tour) graph with vertex set V and edge set E
$G = (V, A)$	(tour) graph with vertex set V and arc set A
$T = (V, E)$	(tour) tree with vertex set V and edge set E
$[v, w] \in E$	(undirected) edge between v and w
$(v, w) \in A$	(directed) arc from v to w
$G' = (V', E', W)$	converted graph of tour graph G
v_{kl}	vertex of converted tour graph
v_0 or 0	base station
l_v	minimum traversal time (without stops) of tour v
l_v^S	minimum time any sensing location can be reached from a point on tour v
L	$\max_{v \in V} \{l_v\}$
d_v	direction (cw or ccw) robot v traverses its tour (cw or ccw)
$p_r(t)$	position of robot r at time t
$time_v(p, q, d)$	minimum travel time on tour v from point p to point q with direction d
p_v^{start}	start position of robot v on its tour v
$p_v^{meet}(w)$	meeting point of robot v on tour v with robot w
$wait_v(p)$	waiting time for robot v on meeting point p
Δ_{wv}	time difference between the starts of robots v and w at their start positions, $wait_w(p_w^{start}) - wait_v(p_v^{start})$
$P(v)$	set of all positions on tour v
τ_v	traversal time plus waiting times on tour v
$dist_G(s, d)$	length of shortest path between vertices s and d in (weighted) graph G

References

1. Acevedo, J.J., Arrue, B.C., Diaz-Banez, J.M., Ventura, I., Maza, I., Ollero, A.: Decentralized strategy to ensure information propagation in area monitoring missions with a team of UAVs under limited communications. In: Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), pp. 565–574. IEEE (2013)
2. Acevedo, J.J., Arrue, B.C., Maza, I., Ollero, A.: Cooperative perimeter surveillance with a team of mobile robots under communication constraints. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS), pp. 5067–5072. IEEE (2013)
3. Anisi, D.A., Ögren, P., Hu, X.: Cooperative minimum time surveillance with multiple ground vehicles. IEEE Trans. Autom. Control **55**(12), 2679–2691 (2010)
4. Banfi, J., Basilico, N., Amigoni, F.: Minimizing communication latency in multirobot situation-aware patrolling. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 616–622 (2015)
5. Chang, C.Y., Chen, G., Yu, G.J., Wang, T.L., Wang, T.C.: TCWTP: Time-constrained weighted targets patrolling mechanism in wireless mobile sensor networks. IEEE Trans. Syst. Man Cybern.: Syst. **45**(6), 901–914 (2015)
6. Chevalere, Y.: Theoretical analysis of the multi-agent patrolling problem. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT), pp. 302–308. IEEE (2004)
7. Erdelj, M., Natalizio, E., Chowdhury, K.R., Akyildiz, I.F.: Help from the sky: Leveraging UAVs for disaster management. IEEE Pervasive Comput. **16**(1), 24–32 (2017)
8. Fargeas, J.L., Hyun, B., Kabamba, P., Girard, A.: Persistent visitation under revisit constraints. In: Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS), pp. 952–957. IEEE (2013)
9. Flushing, E.F., Gambardella, L.M., Di Caro, G.A.: Simultaneous task allocation, data routing, and transmission scheduling in mobile multi-robot teams. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1861–1868. IEEE (2017)
10. Frederickson, G.N., Hecht, M.S., Kim, C.E.: Approximation algorithms for some routing problems. In: Proceedings of the Annual Symposium on Foundations of Computer Science, pp. 216–227. IEEE (1976)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability - A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1979)
12. Ghamry, K.A., Zhang, Y.: Cooperative control of multiple UAVs for forest fire monitoring and detection. In: Proceedings of the IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), pp. 1–6. IEEE (2016)
13. Grotli, E.I., Johansen, T.A.: Path- and data transmission planning for cooperating UAVs in delay tolerant network. In: Proceedings of the IEEE Globecom Workshops, pp. 1568–1573. IEEE (2012)
14. Hollinger, G.A., Singh, S.: Multirobot coordination with periodic connectivity: Theory and experiments. IEEE Trans. Robot. **28**(4), 967–973 (2012)
15. Kantaros, Y., Guo, M., Zavlanos, M.M.: Temporal logic task planning and intermittent connectivity control of mobile robot networks. IEEE Trans. Autom. Control **PP**(c), 1–1 (2019)
16. Kantaros, Y., Zavlanos, M.M.: Distributed intermittent connectivity control of mobile robot networks. IEEE Trans. Autom. Control **62**(7), 3109–3121 (2017)
17. Khan, A., Rinner, B., Cavallaro, A.: Cooperative robots to observe moving targets: Review. IEEE Trans. Cybern. **48**(1), 187–198 (2018)
18. Kingston, D., Beard, R., Holt, R.: Decentralized perimeter surveillance using a team of UAVs. IEEE Trans. Robot. **24**(6), 1394–1404 (2008)
19. Lauri, F., Créput, J.C., Koukam, A.: The multi-agent patrolling problem theoretical results about cyclic strategies. In: Lecture

- Notes in Computer Science, vol. 8473, pp. 171–182. Springer (2014)
20. Liu, B., Dousse, O., Nain, P., Towsley, D.: Dynamic coverage of mobile sensor networks. *IEEE Trans. Parallel Distrib. Syst.* **24**(2), 301–311 (2013)
 21. Machado, A., Ramalho, G.: Multi-Agent Patrolling: An Empirical Analysis of Alternative Architectures, Lecture Notes in Computer Science, vol. 2581. Springer, Berlin (2003)
 22. Magnanti, T.L., Wolsey, L.A.: Optimal trees. In: *Handbooks in Operations Research and Management Science*, vol. 7, pp. 503–615. Elsevier B.V (1995)
 23. Manyam, S.G., Rasmussen, S., Casbeer, D.W., Kalyanam, K., Manickam, S.: Multi-UAV routing for persistent intelligence surveillance & reconnaissance missions. In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 573–580. IEEE (2017)
 24. Masehian, E., Jannati, M., Hekmatfar, T.: Cooperative mapping of unknown environments by multiple heterogeneous mobile robots with limited sensing. *Robot. Auton. Syst.* **87**, 188–218 (2017)
 25. Mosteo, A.R., Montano, L.: Concurrent tree traversals for improved mission performance under limited communication range. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2840–2845. IEEE (2009)
 26. Nigam, N., Bieniawski, S., Kroo, I., Vian, J.: Control of multiple UAVs for persistent surveillance: Algorithm and flight test results. *IEEE Trans. Control Syst. Technol.* **20**(5), 1236–1251 (2012)
 27. Pasqualetti, F., Durham, J.W., Bullo, F.: Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms. *IEEE Trans. Robot.* **28**(5), 1181–1188 (2012)
 28. Pasqualetti, F., Franchi, A., Bullo, F.: On cooperative patrolling: Optimal trajectories, complexity analysis, and approximation algorithms. *IEEE Trans. Robot.* **28**(3), 592–606 (2012)
 29. Ponda, S.S., Johnson, L.B., Kopeikin, A.N., Choi, H.L., How, J.P.: Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. *IEEE J. Selected Areas Commun.* **30**(5), 861–869 (2012)
 30. Portugal, D., Pippin, C., Rocha, R.P., Christensen, H.: Finding optimal routes for multi-robot patrolling in generic graphs. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 363–369. IEEE (2014)
 31. Rossi, M., Brunelli, D.: Autonomous gas detection and mapping with unmanned aerial vehicles. *IEEE Trans. Instrum. Meas.* **65**(4), 765–775 (2016)
 32. Santos, J.M., Krajník, T., Duckett, T.: Spatio-temporal exploration strategies for long-term autonomy of mobile robots. *Robot. Auton. Syst.* **88**, 116–126 (2017)
 33. Scherer, J., Rinner, B.: Persistent multi-UAV surveillance with energy and communication constraints. In: *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1225–1230 (2016)
 34. Scherer, J., Rinner, B.: Short and full horizon motion planning for persistent multi-UAV surveillance with energy and communication constraints. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 230–235. IEEE (2017)
 35. Scherer, J., Rinner, B., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H.: An autonomous multi-UAV system for search and rescue. In: *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use - DroNet '15*, pp. 33–38 (2015)
 36. Smith, S.L., Schwager, M., Rus, D.: Persistent robotic tasks: Monitoring and sweeping in changing environments. *IEEE Trans. Robot.* **28**(2), 410–426 (2012)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Jürgen Scherer is a Ph.D. student in the Pervasive Computing Group at the Institute of Networked and Embedded Systems, University of Klagenfurt. He received the M.Sc. in computer science from University of Klagenfurt in 2012. His research interests include multi-robot coordination and path planning.

Bernhard Rinner received the M.Sc. and Ph.D. degrees in telematics from the Graz University of Technology, Graz, Austria, in 1993 and 1996, respectively. He is professor of pervasive computing and deputy head of the Institute of Networked and Embedded Systems at University of Klagenfurt, Austria. He held research positions with the Graz University of Technology from 1993 to 2007 and with the University of Texas at Austin, TX, USA, from 1998 to 1999. His current research interests include sensor networks, multi-robot systems, selforganization and pervasive computing. He has authored and co-authored more than 230 papers for journals, conferences and workshop, has led many research projects and has served as reviewer, program committee member, program chair and editor-in-chief. Dr. Rinner is currently associate editor of the *Ad Hoc Networks Journal* and member of the board of the Austrian Science Fund.