Short and Full Horizon Motion Planning for Persistent multi-UAV Surveillance with Energy and Communication Constraints

Jürgen Scherer and Bernhard Rinner[1]

***Abstract*— The strong resource limitations of unmanned aerial vehicles (UAVs) pose various challenges for UAV applications. In persistent multi-UAV surveillance, several UAVs with limited communication range and flight time have to repeatedly visit sensing locations while maintaining a multi-hop connection to the base station. In order to achieve persistence, the UAVs need to fly back to the base station in time for recharge. However, simple motion planning algorithms can result in mutual movement obstructions of UAVs caused by the constraints. We introduce two planning algorithms with different planning horizons and cooperation and compare their performance in simulation studies. It can be seen that the short horizon uncooperative strategy can outperform other strategies if a sufficient number of UAVs is used. The full horizon strategy can generate a solution visiting all sensing locations if the existence conditions for such a solution are fulfilled.**

I. Introduction

Recent progress in the field of aerial robotics has lead to increasing interest in using unmanned aerial vehicles (UAVs) for civilian use cases including environmental monitoring and disaster management [1]. In the latter case, they can be used to continuously monitor disaster sites and send the sensed data, such as images or videos, via wireless transceivers to a base station to support the mission operators.

In this work we assume communication and energy constraints. The *communication constraint* forces the UAVs to maintain a connection to the base station via a multi-hop link throughout the mission. The *energy constraint* forces a UAV to reach the base station with the remaining energy and to recharge before continuing the mission. This is especially important for disaster management scenarios where the mission operators have to assess the situation over long mission periods. The goal of persistent surveillance is to plan paths for multiple UAVs such that all sensing locations are repeatedly visited and the communication and energy constraints are satisfied.

The persistent surveillance problem has been tackled from different perspectives in literature. One approach is to map it to a Vehicle Routing Problem (VRP) with inter-depot routes to account for the limited energy capacity [2], [3]. Other approaches decouple path generation and controlling of agents along these paths [4], [5]. Control policies have been derived to select the next goal [6] to minimize the maximum cell age or to adjust direction and speed of UAVs [7] to achieve a desired coverage. The Persistent Vehicle Routing Problem (P-VRP) with recharging stations is modeled with temporal logic specifications in [8].

Maintaining connectivity is a prominent task in robotic networks. In [9] the effect of connectivity on the coverage performance is presented. A distributed controller for maintaining network integrity is proposed in [10]. In [11] the planning of a mission to visit certain targets is done offline exploiting a radio propagation path loss simulator. Planning for periodic connectivity in environments with obstacles is done in [12]. A heuristic for the VRP with communication sites is presented in [13]. In [14] a mathematical programming approach for planning search and rescue missions for different connectivity demands is presented.

In contrast to the state of the art, we jointly consider communication and energy constraints which can result in mutual movement obstructions of UAVs in persistent surveillance missions. These problems do not occur when both constraints are considered individually. In this paper we compare algorithms with different planning horizons to overcome the obstructions. *Short horizon* approaches based on [6] only plan for the next sensing location a UAV has to approach, whereas *full horizon* algorithms plan tours through all sensing locations the UAVs have to follow. Additionally, our proposed full horizon strategy can generate a solution visiting all sensing locations if the existence conditions for such a solution are fulfilled.

The outline of the paper is as follows: Section II describes persistent surveillance and the movement obstructions. Section III introduces our planning algorithms, and Section IV presents the simulation results.

II. Problem description

We assume that the mission area is convex and divided into a two-dimensional grid of square cells. A subset $\mathcal{S} = \{1, \ldots, S\}$ of these cells represents the sensing locations and a base station is located at a particular cell and denoted as 0. A set of UAVs $\mathcal{U} = \{1, \ldots, U\}$ visits sensing locations repeatedly to take measurements. Time is divided into discrete time steps, and a UAV can move to one neighboring cell (there are at most 8 neighboring cells) or stay at its current cell at each time step. The energy capacity or maximum flight time E is measured in time steps, and a UAV consumes one energy unit at every time step regardless of whether the UAV is staying at the current cell or moving to a neighboring cell. The energy constraint requires that each UAV has to reach the base station before its energy is depleted. The communication range R^{com} is measured in number of cells. There is a link between two UAVs or a UAV and the base station if

[1]Both authors are with the Institute of Networked and Embedded Systems, Alpen-Adria-Universität Klagenfurt, Austria, {juergen.scherer, bernhard.rinner}@aau.at

the distance between them is less than R^{com}. The distance between two UAVs or a UAV and the base station is the Euclidean distance between the centers of the cells at which they are located. The communication constraint requires that there must be a single- or multi-hop link from every UAV to the base station at every time step. The age $a_{t,s}$ of a sensing location s at a certain instant t denotes the number of time steps that have passed since the most recent visit of any UAV at s. The goal is to find solutions that minimize the maximum age over all sensing locations.

The following conditions for the existence of a solution that visits every sensing location can be specified. First, if the distance between the base station and any sensing location is smaller than R^{com}, the minimum amount of energy that is necessary to guarantee a solution that visits every sensing location is the energy necessary to fly directly from the base station to the sensing location and return to the base station. If the communication range is smaller than the distance to the sensing location, UAVs are necessary that act as relays and leave the base station as necessary. In this case the supporting UAVs need enough energy to fly to the designated relay positions and return to the base station. Second, there must be enough UAVs such that every sensing location can be reached. These are sufficient and necessary conditions for the existence of a solution, but not sufficient ones for every algorithm (as described in the next paragraphs).

The sensing selection policy in [6] assigns a sensing location for each UAV u at each time step t based on a weighted combination of the age $a_{t,s}$ of sensing location s, the distance between the UAV and the sensing location $\chi_{u,s}$, and the minimum distance between s and any other UAV:

$$w_{u,s} = a_{t,s} + \omega_0 \chi_{u,s} + \omega_1 \min_{v \neq u} \chi_{v,s}. \quad (1)$$

Each UAV gets assigned to the sensing location with the highest value $w_{u,s}$ individually. The weighting parameters ω_0 and ω_1 are determined by an offline optimization algorithm where the parameter space is sampled and the mission is simulated to get the objective value for a particular set of parameters. We adopt this approach to enforce the connectivity constraints by disallowing moves that would result in a disconnected network and denote it as *short horizon* (SH) algorithm. A safe path calculation ensures that every UAV can reach the base station with the remaining energy without violation of the connectivity constraints. More details can be found in [15].

It is shown in [6] that with suitable parameters ω_0 and ω_1 the performance is close to an optimal solution for a single UAV for the scenario considered therein. The optimal solution for the single UAV case can be determined by moving the UAV with maximum speed along the shortest tour through all sensing locations. The shortcoming of the simple SH approach is that constraints that force connectivity and limit the flight time have an impact on the ability of a single UAV for moving to its desired sensing location. The problems that can arise are *oscillation* and *mutual blocking* and are described with an example in Figure 1 (a) and (b), respectively.

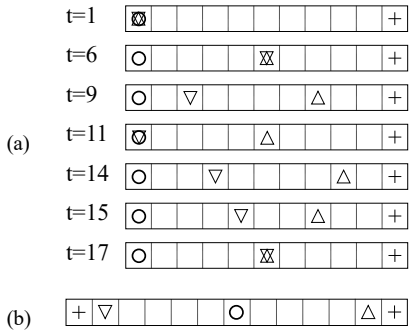


Fig. 1. Limitation of short horizon planning depicted for a one-dimensional surveillance problem: An area consisting of a line of cells with the base station (circle) and sensing locations (cross), which have to be visited by UAVs (triangles) is shown. The communication range is limited to 5 cells. (a) Oscillation: Two UAVs with an energy capacity of 10 and 20 time steps, respectively have to visit the sensing location. The limited communication range requires one UAV sensing and the other UAV relaying the data to the base station. The simple SH approach sends both UAVs towards the sensing location (time steps 1 to 6) and forces one UAV back to the base station because of the flight time limits (time steps 6 to 11). After recharging the UAV continues the mission and both UAVs approach the sensing location (12 to 14) until the other UAV has to return to the base station (after 14). This procedure continues forever and the sensing location will never be reached. (b) Mutual blocking: There are two sensing locations at both ends of the area and the base station in the center. Each of the two UAVs approaches the sensing location but the connectivity constraints prevents each of them from reaching the sensing location.

III. Algorithm description

A. Short horizon cooperative (SHC)

To overcome the mutual blocking problem we developed an extension to SH based on graph matching and formation reconfiguration. The idea is depicted in Figure 2. The goal is that UAVs change their positions from an initial configuration to a final configuration. For this purpose, a configuration is modeled as graph where the nodes are UAVs and the base station and there is an edge between two nodes if the nodes are within communication range. The initial configuration is determined by the positions at a particular time step, and the final configuration are the positions of UAVs at relay positions and sensing locations that should be visited such that no edge length in the graph exceeds R^{com}. Before a reconfiguration of UAVs can happen, a graph matching between the initial graph and the final graph is necessary to determine which UAVs should move to which position. Therefore, we employ the Initial Tree Selection (ITS) [16] using a weighted graph matching algorithm [17] together with a prefix labeling algorithm [18]. The labels of the nodes in Figure 2 are the outcome of this graph matching algorithm. Nodes in the initial graph that do not have a correspondence in the final graph are called *extra* nodes, and nodes in the final graph that do not have a correspondence in the initial graph are called *missing* nodes. The goal is to move the extra UAVs such that the communication graph finally contains the final graph and the UAV can approach their goals while maintaining the final graph.

At the beginning of the planning horizon (iteration) the algorithm calculates the desired final configuration of the UAVs and the reconfiguration procedure determines the

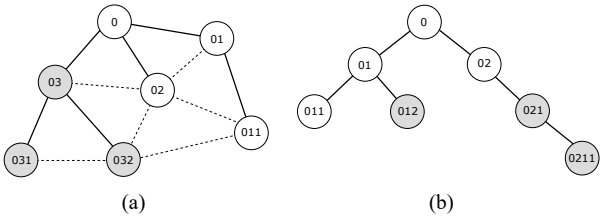
length of the iteration in time steps. One iteration is outlined in Algorithm 1. Here P represents a set of paths, one for each UAV, where a path is a sequence of cells which describes the moves of a UAV. The moves are synchronized over all UAVs, i.e. all UAVs perform the next move on their path at the same time. The UAV state vector p consists of the positions of the UAVs at the beginning of the iteration and is indexed by u, i.e. $p(u)$ is the position of UAV u. The matrix A determines the assignment of a UAV to a sensing location, i.e. if $A(u) = s$ then UAV u is assigned to s. This matrix is calculated in the same way as for the SH algorithm based on the weighing matrix W, where $W(u, s) = w_{u,s}$.

First, the assigned sensing locations in A are ordered according to their values in W. Then an approximation of a minimum node Steiner Tree which contains the base station s_0 and as many sensing locations (terminals) as possible such that the number of nodes (terminals and intermediate nodes) in the tree is not larger than $U + 1$ (available UAVs plus base station) is calculated. We use the algorithm in [19] and set the node weight to one. The input for the algorithm is the graph containing all the cells as vertices and an edge between two vertices if the distance between the cells is not larger than R^{com}, together with the terminal sensing locations. The resulting tree is the desired final tree T_f and is labeled with the prefix labeling algorithm and matched with the connectivity graph of the UAVs and the base station given by the actual positions of the UAVs and the base station (p_0). This results in a partially labeled graph T_t with unlabeled nodes that could not be matched. Then the *extra* and *missing* nodes are determined. After that, the shortest paths in T_t between all pairs of extra nodes and the parents of nodes of missing nodes that are also in the actual tree T_t are calculated. A minimum weight matching assigns an extra node to a missing node and the path between them is stored in SP. Afterwards, the UAVs move along the paths and the tree T_f is relabeled after a UAV reaches its goal. When the trees match ($compare\_graphs$ returns $true$) the UAVs move to their final goals (the cells determined by T_f). The iteration is finished when $s_1, \ldots, s_k$ sensing locations have been reached, where $k \leq U$ is an input to the algorithm (this means a new iteration can start before all U selected sensing locations have been reached).

B. TSP based partitioning (TSPP)

The TSP (Traveling Salesperson Problem) based partitioning algorithm partitions the sensing locations based on the number of UAVs available and the communication range in disjoint sets divided by rays that origin at the base station. In every partition a TSP tour through all sensing locations is generated which a set of UAVs have to follow where one UAV acts as leader and supporter UAVs act as relays. An example of an partition is shown in Figure 3.

The algorithm for partitioning is shown in Algorithm 2. The function $get\_partitions$ adds sensing locations with increasing distance to the base station to $\mathcal{S}_0$ until there are no more sensing locations that can be reached with the number of UAVs provided in the second argument or $MST(\mathcal{S}_0)$ exceeds $MST(\mathcal{S})/K$. Here, MST denotes the weight of the minimum spanning tree of the partition, which serves an estimate of the TSP tour. The weight of the edges between two sensing locations is the length of the shortest path between them. Then the function $get\_partitions$ orders the remaining sensing locations by the angle of the line between the base station and the sensing location and adds them in that order to a partition $\mathcal{S}_k$, $k > 0$ (these are referred to as *angular partitions*). A new angular partition $k+1$ is created if $MST(\mathcal{S}_k)$ exceeds $MST(\mathcal{S})/K$.



(a) (b)

Fig. 2. Outcome of the graph matching algorithm. The nodes are the base station (node '0') and the UAVs. The initial graph (a) models the state of the UAVs and the base station at a particular time step with a line between two nodes if they are within communication range. A dashed line indicates the communication link and a solid line indicates the selected tree in the initial graph. The graph (b) models the final desired UAV configuration, where each UAV is assigned a relay position or a sensing location. For example, UAVs labeled with '011', '012', and '0211' are assigned to sensing locations but can reach them only with help of relays labeled with '01', '02', and '021'. The shaded nodes are the extra nodes (a) and the missing nodes (b), respectively.

In particular, in the $while$ loop in Algorithm 2 the number of angular partitions K is increased until the number of UAVs necessary to cover all partitions at the same time exceeds $|\mathcal{U}|$. The function get_min_U returns the minimum number of UAVs necessary to reach any of the sensing locations in the set provided. In the example for Figure 3 $get\_min\_U = 4$. Note, that the second argument of the call to $get\_partitions$ (UAVs for $\mathcal{S}_0$) is zeros in the loop (and therefore, $\mathcal{S}_0$ will be empty) since this number is not known at this moment. After the loop finishes the remaining UAVs available for $\mathcal{S}_0$ are calculated and the final partition is calculated. Now, $get\_partitions$ is called with U_0 as second argument.

After the area is partitioned, the paths for the UAVs for each partition are generated. This is shown in Algorithm 3. For each angular partition a TSP tour on the sensing locations $\mathcal{S}_k$ is calculated and the paths for $get\_min\_U(\mathcal{S}_k)$ $(k > 0)$ UAVs are generated, where one leading UAV follows the tour and the supporting UAVs leave the base station as necessary (see Figure 3 (b)). If necessary, the leader (and therefore also the supporters) returns to the base station for recharge and continues with the next sensing location on the path such that the maximum flight time is not exceeded. Additionally, the tour can be shortcut if sensing locations have already been visited by any supporter UAV or during the return to the base station. The set $\mathcal{V}$ is the set of all sensing locations that have been visited by leader or supporter UAVs. This set can be ignored for the calculation of the paths for $\mathcal{S}_0$. In every iteration it is checked whether the path for $\mathcal{S}_0$ or the longest

Algorithm 1 *shc_iteration*

Input:
UAV states p, path P, assignment A, weight matrix W, number of intended sensing locations to reach before a new iteration starts k
Output:
extended path P

$(s_1, \ldots, s_U) \leftarrow$ sort SL in A according to values in W
for $i = 1$ **to** U **do**
  $T'_f \leftarrow calc\_steiner\_tree(\{s_0, s_1, \ldots s_i\})$
  if $|V(T'_f)| > (U+1)$ **then** exit **for** loop
  $T_f \leftarrow T'_f$
$L_f \leftarrow label\_trie(T_f)$
$G_c \leftarrow calc\_conngraph(p \cup p_0)$
$(T_t, L_t) \leftarrow calc\_graph\_matching(G_c, T_f, L_f)$
$L_t \leftarrow complete\_labels\_trie(T_t, L_t)$
$(\mathcal{V}_e, \mathcal{V}_m) \leftarrow calc\_extra\_missing(L_t, L_f)$
for $i \in \mathcal{V}_e$, $j \in \mathcal{V}_m$ **do**
  $D(i,j) \leftarrow$ length of shortest path in T_t from node i to the closest parent of j that is also in T_t
$M \leftarrow calc\_matching(D)$
for $u \in \mathcal{V}_e$ **do** $SP(u) \leftarrow$ shortest path in T_t from u to $M(u)$
while true **do**
  for $u \in \mathcal{U}$ **do**
    if $u \in \mathcal{V}_e$ **then**
      $p'(u) \leftarrow$ make a move along $SP(u)$
      if u reached goal on $SP(u)$ **then**
        $L_t \leftarrow update\_labels(L_t)$
        $\mathcal{V}_e \leftarrow \mathcal{V}_e \setminus \{u\}$
    $G_c \leftarrow calc\_conngraph(p' \cup p_0)$
    if $compare\_graphs(G_c, L_t, T_f, L_f)$ **then**
      $p'(u) \leftarrow$ make move towards SL which has same label as i
  $P \leftarrow P$ extended with p'
  if all s_1 to s_k have been reached by any UAVs **then** exit **while** loop

Algorithm 2 *partition*

Input:
sensing locations $\mathcal{S}$, UAVs $\mathcal{U}$
Output:
partition $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K)$

$\mathcal{S}_1 \leftarrow \mathcal{S}, \mathcal{S}_0 \leftarrow \emptyset, \mathcal{S}_2 \leftarrow \emptyset, \ldots$
$K \leftarrow 1$
while $\sum_{k=1}^{K} get\_min\_U(\mathcal{S}_k) \leq |\mathcal{U}|$ **do**
  $(\mathcal{S}'_0, \mathcal{S}'_1, \ldots) \leftarrow (\mathcal{S}_0, \mathcal{S}_1, \ldots)$ //save for later
  $K \leftarrow K + 1$
  $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K) \leftarrow get\_partitions(\mathcal{S}, 0, K)$
$K \leftarrow K - 1$
$U_0 \leftarrow |\mathcal{U}| - \sum_{k=1}^{K} get\_min\_U(\mathcal{S}'_k)$
$(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K) \leftarrow get\_partitions(\mathcal{S}, U_0, K)$

path in any $\mathcal{S}_k$, $k > 0$, is longer and sensing locations are moved from $\mathcal{S}_0$ to $\mathcal{S}_k$ (path for $\mathcal{S}_0$ is longer than longest path in any $\mathcal{S}_k$), or from $\mathcal{S}_k$ to $\mathcal{S}_0$ (longest path in any $\mathcal{S}_k$ is longer than path for $\mathcal{S}_0$).

Given the existence conditions described in Section II this algorithm will generate a solution that visits every sensing location at least once. For a persistent mission the solution can be repeated since all paths start and end at the base station.

IV. Simulation results

In this section we compare four algorithms: (i) short horizon (SH), (ii) short horizon cooperative (SHC), (iii) TSP based partitioning (TSPP), and (iv) tree traversal (TT). The

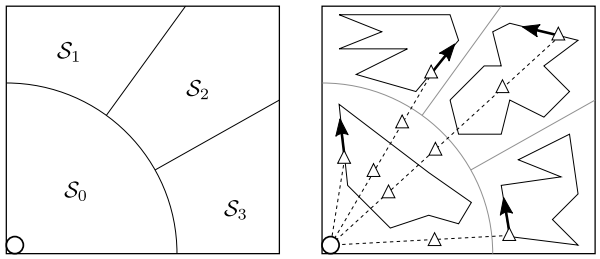


Fig. 3. TSPP algorithm: Example of an partitioning of an area and path generation. The base station is located at the bottom left corner of the area, there are 14 UAVs available, and the minimum number of UAVs necessary to reach all the sensing locations is 4. (a) The set of UAVs can be divided by the number of minimum UAVs necessary to create three partitions that can be patrolled by three teams of UAVs simultaneously. For the three partitions $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ (angular partitions) there are 12 UAVs necessary. For the remaining two UAVs a partition $\mathcal{S}_0$ can be created. (b) Teams of UAVs with a leader and supporter UAVs follow the generated tours.

TT algorithm is based on the concurrent tree traversal algorithm [20] using the union of the shortest paths from every sensing location to the base station as input. This algorithm traverses a given tree placing relays when necessary and traversing branches concurrently if possible. All algorithms have been implemented in Matlab. The parameters ω_0 and ω_1 are determined with the pattern search algorithm provided by Matlab. The TSP tours for TSPP are determined with a simple genetic algorithm. The simulations are run on an area of 21x21 cells where the base station is at the lower left corner and the upper right 20x20 cells are sensing locations.

First, we investigate the effect of decreasing communication range R^{com} with increasing number of UAVs U and unlimited energy capacity E. The number of UAVs is increased from 2 to 10, whereas the communication range decreases such that all UAVs are necessary to reach the top right cell. Figure 4 depicts the coverage time which is the number of time step it takes until every sensing location has been visited at least once. This number serves as an indicator for the performance of persistent surveillance scenarios. It can be seen that the values are almost constant for the approaches with increasing number of UAVs. The only exception is the SH algorithm for which no parameters ω_0 and ω_1 could be found for 10 UAVs such that the area is covered within the planning horizon (2000 time steps). This is an indication of the mutual blocking problem which gets more prominent as the number of UAVs increases and the communication range decreases. Since the genetic algorithm has a probabilistic part, the simulations for TSPP are run 5 times and the results also include the standard deviation. It can be seen by the slight decrease of the coverage time that TSPP can exploit the increasing number of UAVs since the supporting UAVs visit sensing locations that have not to be visited by the leading UAV which results in a shortcut route. SHC cannot make use of the increasing number of UAVs because the reconfigurations to the final desired trees consume more time with increasing fleet size but it can outperform the others with small number of UAVs.

Algorithm 3 *tspp*

Input:
sensing locations $\mathcal{S}$, partition $(\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K)$
Output:
path P

$resume \leftarrow true$, $obj^{min} \leftarrow \infty$
while $true$ **do**
  $\mathcal{V} \leftarrow \emptyset$, $obj_1 \leftarrow 0$
  for $k = 1$ **to** K **do**
    $T \leftarrow solve\_tsp(\mathcal{S}_k)$
    $(P'_k, \mathcal{V}') \leftarrow generate\_pathes(T)$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \mathcal{V}'$
    $obj_1 \leftarrow \max\{obj_1, length(P'_k)\}$
  if $|\mathcal{S}_0| > 0$ **then**
    $T \leftarrow solve\_tsp(\mathcal{S}_0 \setminus \mathcal{V})$
    $P'_0 \leftarrow generate\_pathes(T)$
    $obj_0 \leftarrow length(P'_0)$
    if $\max\{obj_0, obj_1\} > obj_{min}$ **then** exit **while** loop
    else
      $obj^{min} \leftarrow \max\{obj_0, obj_1\}$
      if $obj_0 > obj_1$ **then**
        for $k = 1$ **to** K **do**
          $s \leftarrow$ closest SL in $\mathcal{S}_0$ to any SL in $\mathcal{S}_k$
          if $get\_min\_U(\{s\}) \leq get\_min\_U(\mathcal{S}_k)$ **then**
            $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{s\}$, $\mathcal{S}_0 \leftarrow \mathcal{S}_0 \setminus \{s\}$
      else
        for $k = 1$ **to** K **do**
          $s \leftarrow$ closest SL in $\mathcal{S}_k$ to any SL in $\mathcal{S}_0$
          if $get\_min\_U(\{s\}) \leq get\_min\_U(\mathcal{S}_k)$ **then**
            $\mathcal{S}_0 \leftarrow \mathcal{S}_0 \cup \{s\}$, $\mathcal{S}_k \leftarrow \mathcal{S}_k \setminus \{s\}$
      if $\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_K$ did not change **then** exit **while** loop
  else exit **while** loop
$P \leftarrow combine\_paths(P'_0, P'_1, \ldots P'_K)$

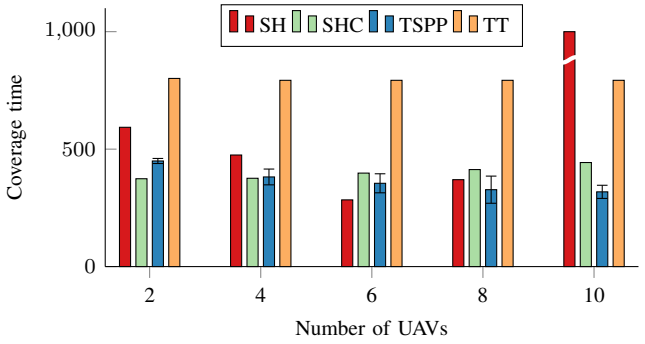


Fig. 4. Comparison of the first coverage time with increasing number of UAVs and decreasing communication range, and unlimited energy capacity E. Due to mutual blocking with a high number of UAVs and small communication range R^{com}, SH is not able to visit all sensing locations in the last scenario.

Figure 5 shows the effect of increasing fleet size with fixed communication range $R^{com} = 8$ (4 UAVs are necessary). It can be seen that SH can exploit the increasing number of UAVs when the communication range is large enough. The drop in the coverage time for TT with 12 UAVs results from the fact that there are enough UAVs to traverse the tree with two teams simultaneously.

Next, we investigate the long term performance by plotting the maximum age of all sensing locations over the time. Figure 6 shows the result for 4 and 8 UAVs with unlimited energy capacity. SHC is not able to maintain the persistent

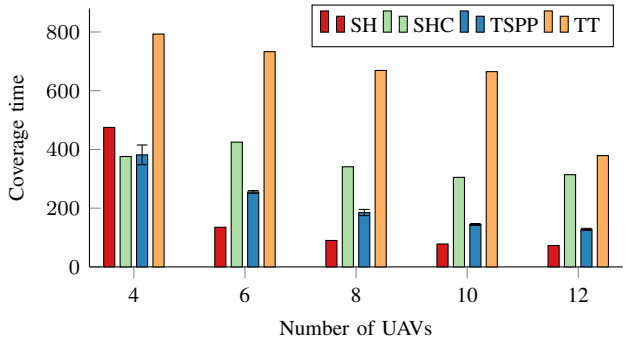


Fig. 5. Comparison of the first coverage time with different number of UAVs, fixed communication range $R^{com} = 8$, and unlimited energy capacity E. The simple SH strategy can outperform the others because the movement restrictions get attenuated with an increasing number of UAVs.

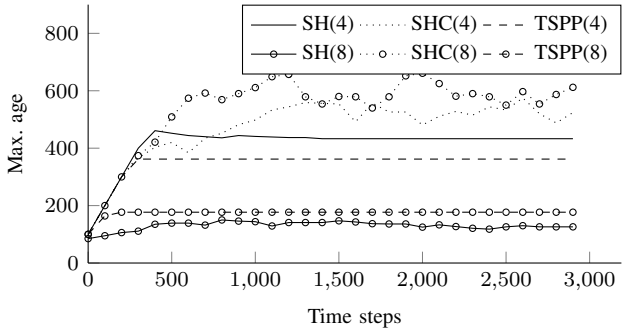


Fig. 6. Comparison of the persistence performance with 4 and 8 UAVs, fixed communication range $R^{com} = 8$, and unlimited energy capacity E. SH can outperform the other algorithms with 8 UAVs. SHC is not able to employ the additional UAVs.

performance after the first coverage (time step 326 and 367 for 4 and 8 UAVs, respectively) because sensing locations with high age get more distributed on the area over time and the reconfiguration time of the UAVs increases and stays high. As seen in Figure 5, SHC is not able to exploit the increasing number of UAVs.

To see the effect of the energy constraints we compare SH and TSPP with 4 UAVs, and set E to 50 and 100 (40 time steps are necessary to reach the sensing location at the top right corner and return to the base station). Because of the bad performance of SHC and TT we do not investigate in these algorithms with limited energy. Capacity values higher than the first coverage time do not decrease the performance considerably compared to the case with unlimited energy. The results are plotted in Figure 7. SH needs considerably longer for the first coverage with $E = 50$ but can improve its performance over time. The reason is that sensing locations are visited while the UAVs are returning to the base station for recharge. This limits the age for the sensing locations which then do not have to be considered for some time thereafter. This reduces the number of sensing locations that effectively have to be considered for visiting and the performance in terms of maximum age increases over time. This effect declines with increasing energy since the interval between returns to the base station is longer. In Figure 8

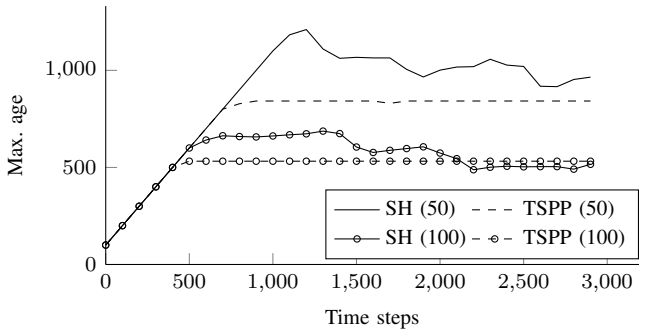


Fig. 7. Comparison of the persistence performance with 4 UAVs, fixed communication range $R^{com} = 8$, and limited energy capacity E of 50 and 100 time steps. With small energy budget it takes longer for SH for the first coverage (first peak of the plot).

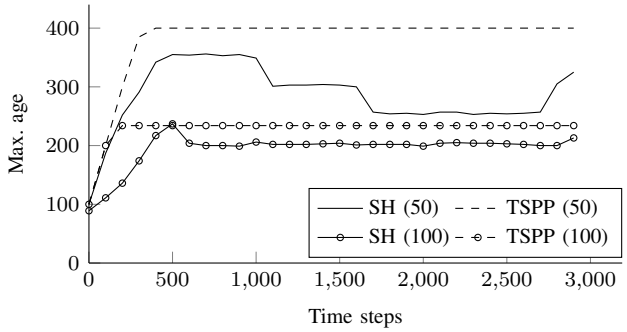


Fig. 8. Comparison of the persistence performance with 8 UAVs, fixed communication range $R^{com} = 8$, and limited energy capacity E of 50 and 100 time steps. With 8 UAVs SH can outperform TSPP in contrast to the scenario with 4 UAVs.

the results of the same scenario with 8 UAVs are plotted. In contrast to the scenario with 4 UAVs, SH is able to outperform TSPP for all time steps.

V. Conclusion

We formulate and compare different algorithms for the persistent surveillance problem with energy and communication constraints that use different planning horizons. Since energy limitations and communication constraints have an impact on the possible movements of the UAV, we formulate existence conditions for a solution. We show that, given enough UAVs attenuating the constraints, a simple short horizon algorithm can make use of an increasing number of UAVs to improve the mission performance although it cannot guarantee to produce a valid solution. In other situations, where the movement of the UAVs is severely limited by the constraints, a patrolling based approach that finds a solution given the existence conditions can outperform the other approaches.

ACKNOWLEDGEMENT

The work was supported by the ERDF, KWF, and BABEG under grant KWF-20214/24272/36084 (SINUS). It has been performed in the research cluster Lakeside Labs.

References

[1] J. Scherer, B. Rinner, S. Yahyanejad, S. Hayat, E. Yanmaz, T. Andre, A. Khan, V. Vukadinovic, C. Bettstetter, and H. Hellwagner, "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*, 2015, pp. 33–38.

[2] V. Mersheeva and G. Friedrich, "Multi-UAV monitoring with priorities and limited energy resources," in *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling*, Apr. 2015.

[3] D. Mitchell, M. Corah, N. Chakraborty, K. Sycara, and N. Michael, "Multi-robot long-term persistent coverage with fuel constrained robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1093–1099.

[4] S. L. Smith, M. Schwager, and D. Rus, "Persistent robotic tasks: Monitoring and sweeping in changing environments," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 410–426, Apr. 2012.

[5] F. Pasqualetti, J. W. Durham, and F. Bullo, "Cooperative patrolling via weighted tours: Performance analysis and distributed algorithms," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1181–1188, Oct. 2012.

[6] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple UAVs for persistent surveillance: Algorithm and flight test results," *IEEE Trans. Contr. Syst. Technol.*, vol. 20, no. 5, pp. 1236–1251, Sept. 2012.

[7] C. Franco, G. Lopez-Nicolas, C. Sagues, and S. Llorente, "Persistent coverage control with variable coverage action in multi-robot environment," in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Dec. 2013, pp. 6055–6060.

[8] C.-I. Vasile and C. Belta, "An automata-theoretic approach to the vehicle routing problem," in *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, July 2014.

[9] E. Yanmaz, "Connectivity versus area coverage in unmanned aerial vehicle networks," in *Proceedings of the IEEE International Conference on Communications (ICC)*, June 2012, pp. 719–723.

[10] M. Zavlanos, A. Ribeiro, and G. Pappas, "Network integrity in mobile robotic networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 3–18, 2013.

[11] E. I. Grøtli and T. A. Johansen, "Task assignment for cooperating UAVs under radio propagation path loss constraints," in *Proceedings of the American Control Conference (ACC)*, June 2012, pp. 3278–3283.

[12] G. A. Hollinger and S. Singh, "Multirobot coordination with periodic connectivity: Theory and experiments," *IEEE Transactions on Robotics*, vol. 28, no. 4, pp. 967–973, Aug. 2012.

[13] J. Banfi, N. Basilico, and F. Amigoni, "Minimizing communication latency in multirobot situation-aware patrolling," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2015, pp. 616–622.

[14] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. a. Di Caro, "Connectivity-aware planning of search and rescue missions," in *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013.

[15] J. Scherer and B. Rinner, "Persistent multi-UAV surveillance with energy and communication constraints," in *Proceedings of the IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2016, pp. 1225–1230.

[16] Z. Kan, L. Navaravong, J. M. Shea, E. L. Pasiliao, and W. E. Dixon, "Graph matching-based formation reconfiguration of networked agents with connectivity maintenance," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 1, pp. 24–35, Mar. 2015.

[17] S. Umeyama, "An eigendecomposition approach to weighted graph matching problems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 5, pp. 695–703, 1988.

[18] L. Navaravong, Z. Kan, J. M. Shea, and W. E. Dixon, "Formation reconfiguration for mobile robots with network connectivity constraints," *IEEE Network*, vol. 26, no. 4, pp. 18–24, 2012.

[19] P. Klein and R. Ravi, "A nearly best-possible approximation algorithm for node-weighted Steiner Trees," *Journal of Algorithms*, vol. 19, no. 1, pp. 104–115, July 1995.

[20] A. R. Mosteo and L. Montano, "Concurrent tree traversals for improved mission performance under limited communication range," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2009, pp. 2840–2845.