Multi-UAV Monitoring with Priorities and Limited Energy Resources

Vera Mersheeva and **Gerhard Friedrich**
vera.mersheeva@aau.at, gerhard.friedrich@aau.at
Alpen-Adria Universität,
Universitätsstraße 65–67, 9020 Klagenfurt, Austria

Abstract

This paper introduces a monitoring problem with limited energy resources and soft constraints on priorities employing a fleet of unmanned aerial vehicles (UAVs). This monitoring problem is a generalization of application cases ranging from surveillance of open-air events to monitoring crime scenes or disaster sites. In order to compute solutions, we propose an insertion heuristic with a negotiation mechanism for energy resources. The solution quality of the heuristic method is compared to cases where an optimal solution is known resulting in an average deviation of approximately 4%. By using this approach, we are able to plan routes for real-world scenarios which are currently unsolvable by general problem solvers in acceptable time. Moreover, the run-time is within seconds for such scenarios. Therefore, the method can be applied to dynamic environments, where re-planning during the mission is required.

Introduction

This paper presents a problem called *continuous monitoring problem with inter-depot routes and priorities* (CMPIDP). The goal of the problem is to provide a fleet of vehicles with routes to periodically survey points of interest. Vehicles have a limited energy capacity that can be renewed at multiple stations. The problem aims at maximizing the number of visits to the points and minimizing the delays between the visits.

This monitoring problem arises in many real-life scenarios such as aerial continuous surveillance of a disaster scene, a concert or a potential crime location. Our monitoring solution is based on a fleet of small-scale unmanned aerial vehicles that are affordable and man-portable. However, the size and design of these drones impose a number of real-life constraints. In contrast to ground robots, micro-UAVs have a maximal traveling time shorter than their average mission. The number of spare batteries is typically limited. Additionally, a fleet can be heterogeneous, i.e. vehicles have different velocities and battery capacities.

As an example, let us consider a scenario where a team of rescuers arrives at the location during or after a disaster, e.g. an earthquake or a forest fire. Needless to say that any information about the situation is essential. Often it cannot be

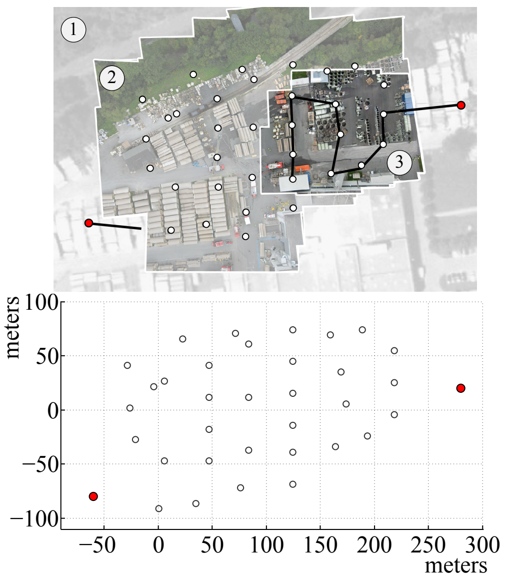

Figure 1: Exploration of an area by UAVs. Picture points and stations are marked as white and filled circles, respectively.

gained from the ground, aerial surveillance is needed. Flight paths for such surveillance have to be computed within five minutes, when the fleet is preparing to take off.

Micro UAVs have a limited flight height and cannot carry wide-angle lens cameras due to the weight. As a result, a single picture of the whole area cannot be taken in one shot. The way to resolve this problem is to split the area into rectangles. UAVs take a picture of each rectangle and transmit this data to the ground station, where received pictures are stitched together into an overview image of high resolution. Locations where photos will be taken are called picture points. The main intention of the aerial exploration is to always have the up-to-date information for every rectangle. Therefore, UAVs have to visit points as often as possible.

Fig. 1 illustrates fire-fighters training at a factory[1]. The input for the route planning algorithm is presented at the bottom of the figure. The image at the top is a partial screenshot of the user interface during the exploration. The background picture from Google Earth™ (Layer 1) was used to define an

[1]More information and videos about our project can be found at http://uav.lakeside-labs.com/

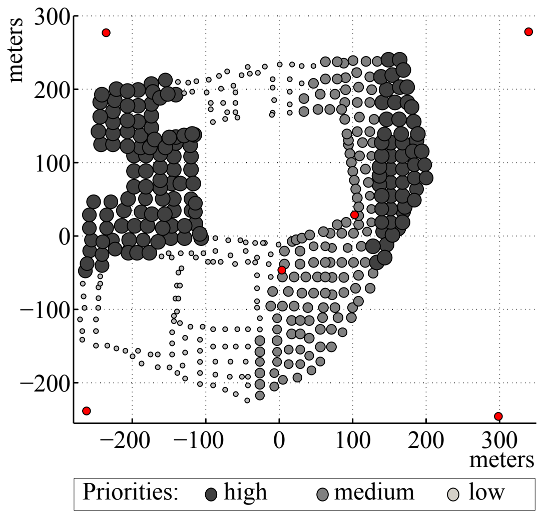

Figure 2: Number of visits depending on priority.

area of interest. During the initial flights, a high-resolution overview image was placed on top (Layer 2). The updates of the overview image are displayed in higher contrast on Layer 3. The black polygonal paths on Layer 3 are the routes which the UAVs have followed so far in their latest flights.

In rescue missions, different subareas can have different importance. For instance, the most important subareas are territories on the edge of a forest fire or those that most likely have people in need of help. These subareas should be visited more often. Fig. 2 depicts picture points with different expectations on the number of visits based on their importance. The greater the number of visits, the larger the circle.

The Operational Research (OR) community has considered a few related problems, e.g. the periodic vehicle routing problem with intermediate facilities (Angelelli and Speranza 2002) or the vehicle routing problem with multiple time windows and multiple visits (Favaretto, Moretti, and Pellegrini 2007). At the same time, some research has been done on patrolling problems with ground robots, e.g. (Almeida et al. 2004), (Stranders et al. 2013), and with areal vehicles, e.g. (Park, Kim, and Jeong 2012), (Yu, Karaman, and Rus 2014). However, these works cannot be directly applied to the monitoring problem with micro UAVs, due to the lack of some essential constraints. The problem of patrolling does not limit battery capacity. The aforementioned OR problems optimize purchased fuel or traveled time instead of gathered information over time.

The monitoring problem described in this paper is a variation of the vehicle routing problem, which is NP-hard. Due to the problem complexity, exact algorithms could not provide solutions in acceptable time. Therefore, our goal was to develop a heuristic for the CMPIDP that uses all available information and is fast enough to react to a number of environmental changes. The developed method is inspired by Solomon's insertion heuristic (IH) (Solomon 1987), the inter-depot insertion heuristic (IDIH) (Mersheeva and Friedrich 2013) and negotiation mechanisms (Almeida et al. 2004). Our new approach, IDIH with battery reservations (IDIH-Reserve), constructs several routes simultaneously by iteratively adding points to a partial solution. In addition, a negotiation mechanism is employed to deal with limited energy resources.

The performance of our method is evaluated in a number of experiments. The first study shows that the IDIH-Reserve returns near-optimal solutions for the scenarios where the optimum can be computed. The average deviation is approximately 4%. Since the optimum *cannot* be found for real-life instances, other evaluations were carried out. First, the proposed method was compared with the IDIH approach on the CMPID instances, i.e. instances with limited resources, fixed mission time and without priorities. The IDIH-Reserve significantly outperforms the IDIH and, thus, demonstrates not only a better efficiency but also its adaptability to related problems. Then the proposed heuristic was evaluated on the patrolling task problem that, similar to the CMPIDP, minimizes the average delay between visits of points and variance of delays. However, in contrast to the monitoring problem, patrolling assumes unlimited battery capacities. Moreover, for this study, we generated instances tailored in favor of patrolling. Even under these discriminating conditions the IDIH-Reserve is only 11% worse than the optimal patrolling solutions regarding the average delay between visits.

The computational time of our method is linear in the number of points, vehicles and visits. Our largest instances (800 points) could be solved within seconds, thus allowing the efficient re-computation of solutions in case of failures and changing environments.

Further we give a formal problem definition followed by a discussion of related work. Then we present our solution method and show its properties by a thorough evaluation.

Problem Description

Input: The *area* is represented as a complete, weighted graph $G = (N, E)$, where N is the set of nodes, E is the set of edges connecting the nodes. The set of nodes N includes a set of base stations N_b and a set of picture points N_p. Every edge $e_{i,j} \in E$ has its weight $d_{i,j}$ that is a distance between nodes i and j that are connected by this edge. In the presence of obstacles, this weight equals the length of the shortest path between the nodes.

Every *picture point* $p \in N_p$ has two properties:

- priority pr_p denotes the importance of point p, i.e. the lower the priority value, the less important this point is;
- time difference lvt_p between the last visit of point p and the start of the mission indicates how long the point remained unvisited before the monitoring started. We introduced lvt_p for re-planning scenarios, where lvt_p and other problem parameters are updated before the re-planning starts. lvt_p is equal to zero for all points, if none of them have been visited before. If only some points have no previous observations, their lvt_p is assigned to the maximal lvt_p value of the visited points.

Every *base station* $b \in N_b$ is characterized by the number of batteries $nBat_{b,t}$ for vehicles of type $t \in T$. The total number of batteries of type t is denoted as $nBat_t$.

Every *vehicle* $v \in V$ is characterized by its type $t_v \in T$, initial remaining battery capacity $inRemCap_v$ and initial location $inLoc_v \in N$. The initial location can be either a base station or a picture point.

Vehicles of *type* $t \in T$ have a certain average speed sp_t, battery capacity $batCap_t$ given as the maximal flight time in time units, service time $servT_t$ (e.g. to take a photo at a picture point) and time to change the battery $tChBat_t$.

Output: A solution to the problem is a sequence of routes for each vehicle $v \in V$, i.e. $R_v = (R_{v,1}, ..., R_{v,nR_v})$, where nR_v is the number of routes of vehicle v. Every route is a sequence of nodes $R_{v,y} = (r_{v,y,1}, ..., r_{v,y,k})$, where k is the number of visits made by vehicle v; $r_{v,y,2}, ..., r_{v,y,k-1}$ are picture points. Nodes $r_{v,y,1}$ and $r_{v,y,k}$ are base stations except for the first point of the first route, which equals the initial location of a vehicle, i.e. $r_{v,1,1} = inLoc_v$.

A feasible solution must fulfill the following requirements: (1) the flight time of each route does not exceed the battery capacity; (2) each node is visited by at most one vehicle at a time; (3) a vehicle can change its battery only at the base stations with available spare batteries of the corresponding type; (4) at the end of the mission all drones must be at the stations.

Additional computations: The travel time of y-th route of vehicle v is a sum of two values, the time needed to travel between its way points and the total service time:

$$time(R_{v,y}) = \left\lceil \sum_{\epsilon=1}^{k-1} d_{r_{v,y,\epsilon}, r_{v,y,\epsilon+1}} / sp_{t_v} \right\rceil + (k-2) \cdot servT_{t_v}.$$

Arrival time at ϵ-th point in route $R_{v,y}$ is a sum of the travel time of the preceding routes, the total time spent on battery changing, and the travel time within route $R_{v,y}$ until ϵ-th point:

$$art(\epsilon, R_{v,y}) = \sum_{y'=1}^{y-1} time(R_{v,y'}) + (y-1) \cdot tChBat_{t_v} + \left(\sum_{\epsilon'=1}^{\epsilon-1} d_{r_{v,y,\epsilon'}, r_{v,y,\epsilon'+1}} / sp_{t_v} + (\epsilon - 2) \cdot servT_{t_v} \right).$$

During route construction, current mission time cmt_v of vehicle v is defined as the time when the vehicle finishes observing its last point in the current partial solution.

For every vehicle v, its total mission time mt_v, i.e. the maximal possible mission duration with the batteries it used, is calculated as follows:

$$mt_v = nChange_v \cdot (batCap_{t_v} + tChBat_{t_v}) + inRemCap_v,$$

where $nChange_v$ is the number of times a vehicle has changed its battery.

Objective: The optimization goal is to maximize the number of up-to-date observations with the given batteries. The ideal solution would be if all points were observed all the time. However, due to the limited number of vehicles and large number of points, this solution is not possible in the real world. In this case, the problem requires a solution with maximal number of visits and minimal delays between them. In addition, points with higher priorities have to be observed more often than points with lower priorities.

The aforementioned requirements for an optimal solution are incorporated in the goal function that is minimized. For a feasible solution x, it is computed as follows:

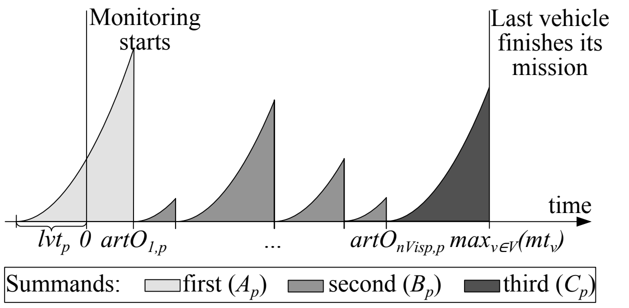

Figure 3: Visual interpretation of penalties for point p.

$$f(x) = \sum_{p \in N_p} \left[A_p + B_p + C_p \right].$$

The three summands are illustrated in Fig. 3. The first summand A_p of the function penalizes the time until the first visit of point p. If the point was observed before the monitoring started ($lvt_p > 0$), then lvt_p is also penalized.

$$A_p = \Big((lvt_p + artO_{1,p}) \cdot pr_p \Big)^2,$$

where $artO_{1,p}$ is the first arrival time at point p.

The second summand B_p summarizes penalties between all visits to the point, where $nVis_p$ is the number of visits:

$$B_p = \sum_{z=1}^{nVis_p - 1} \Big((artO_{z+1,p} - artO_{z,p}) \cdot pr_p \Big)^2.$$

where $artO_{1,p}, ..., artO_{nVis_p,p}$ are arrival times for all visits to point p sorted in ascending order.

Finally, the last summand C_p penalizes the time from the last observation until the overall mission is finished:

$$C_p = \Big((max_{v \in V}(mt_v) + bpenalty - artO_{nVis_p,p}) \cdot pr_p \Big)^2,$$

where *bpenalty* is the sum of penalties for all non-employed batteries and equals their total capacity. If all batteries are utilized, $bpenalty = 0$.

This goal function minimizes the delays between visits of each point. At the same time, it keeps the delays proportional to the priorities of the points.

Related Work

Several related problems exist in the area of Operational Research. One of the most well-known problems in this domain is the vehicle routing problem (VRP) where vehicles have to deliver a required amount of goods to each customer and return to a station. The amount of goods that a vehicle can carry is limited. Typically the total travel time is minimized.

In (Angelelli and Speranza 2002) authors studied a periodic VRP with intermediate facilities. It is a variant of the multi-depot VRP where vehicles can renew their capacity at replenishment facilities. Periodicity is defined as follows: given a set of time units (e.g. days or hours), each target point has a set of possible visit schedules (e.g. a point should be visited either on Monday and Thursday or on Tuesday and Friday). At the end of each time unit vehicles return to their depots. The goal is to select a schedule for each customer and construct a set of routes so that the total distance traveled is minimized. Angelelli and Speranza solved the problem by insertion heuristics and tabu search.

The paper (Favaretto, Moretti, and Pellegrini 2007) describes the VRP with multiple time windows where each customer has a required number of visits and a set of time windows when it can be serviced. The planning horizon is split into sub-periods and every route must start and end within one sub-period. The goal is to minimize the weighted sum of waiting time, travel time and fixed cost of every vehicle in use. An ant colony optimization algorithm is applied.

Related problems were also described in the field of robotics. (Machado et al. 2003) introduced the patrolling task, where robots repetitively visit given points. The planning horizon is typically infinite. In most papers, the capacity constraint is neglected, as typical missions do not outlast the energy capacity of the robots. Commonly the average delay between visits is minimized. One solving approach constructs a TSP-route and places the robots along it (Almeida et al. 2004), (Elmaliach, Agmon, and Kaminka 2009). Another approach is to split the area into clusters, one for each vehicle, and solve a TSP for each cluster (Ahmadi and Stone 2006), (Smith and Rus 2010). Other methods include negotiation-based algorithms (Almeida et al. 2004), (Poulet, Corruble, and Seghrouchni 2012), reinforcement learning and Markov decision process (Santana et al. 2004), (Huynh, Enright, and Frazzoli 2010), divide-and-conquer algorithms (Stranders et al. 2013), an ant-like algorithm (Cannata and Sgorbissa 2011).

(Park, Kim, and Jeong 2012) studied an unmanned vehicle monitoring problem with priorities that minimizes the possibility of enemy infiltration within the fixed time horizon. This problem was solved by simulated annealing.

Only few works on patrolling problems consider capacity renewal, typically with infinite amount of fuel in replenishment facilities. The problem formulated in (Las Fargeas et al. 2012), where every point has a deadline for every visit, aims at minimizing the cost of purchased fuel. The authors suggested a divide-and-conquer algorithm. A UAV routing problem in (Stump and Michael 2011) was modeled as VRP with multiple time windows and multiple visits. This problem minimizes total travel cost and was solved by an adapted Lagrangian branch-cut-and-price approach.

The major difference between the CMPIDP and all previously described problems is the limited number of vehicles as well as energy resources. Due to a number of reasons, in many application cases, infinite recharging of vehicles is not feasible. Therefore, it is of high importance to utilize available resources with maximal information gain.

Papers (Mersheeva and Friedrich 2012) and (Mersheeva and Friedrich 2013) study two similar problems without priorities and with fixed mission time, i.e. (1) the continuous monitoring problem (CMP) and (2) CMP with inter-depot routes (CMPID). The first problem was solved by variable neighborhood search (VNS) with either an insertion heuristic or a modified savings algorithm. The second paper applied VNS with an inter-depot insertion heuristic (IDIH).

To summarize, the state of the art in AI and OR communities considers several related problems. However, the important problem of monitoring with limited energy resources is not sufficiently covered. The aim of this paper is to fill this gap and to draw more attention to this significant problem.

Solution Method

The IDIH-Reserve heuristic suggested in this paper is based on the insertion heuristic (Solomon 1987), and negotiation mechanisms (Almeida et al. 2004).

Solomon's insertion heuristic was proposed for the VRP with time windows. It constructs one route at a time by inserting one point after the other until all points have been visited. At each step, a point to insert is selected by an evaluation function, e.g. time of arrival at a point. It should reflect the optimization criteria of a problem but is not equal to it.

Due to several significant differences between the problems, Solomon's method cannot be applied for the CMPIDP. First of all, in the CMPIDP, a fleet of vehicles is heterogeneous, and every vehicle can make multiple trips. Therefore, the IDIH-Reserve constructs routes for all vehicles in parallel. Secondly, in the CMPIDP, every point is visited multiple times and has no time windows. To cope with the larger search space, we insert points only at the end of a vehicle's last route. Finally, in the CMPIDP, vehicles can make multiple trips, limited by the number of batteries. For limited resources, the IDIH-Reserve has a negotiation mechanism.

The main steps of the IDIH-Reserve are shown in Algorithm 1. The heuristic first creates a route for every vehicle v by adding its initial location $inLoc_v$ as the first element (Line 2). Then it assigns batteries to vehicles with a reservation mechanism that is described later. Afterwards the main loop in Lines 3–25 inserts points to the solution one at a time. It terminates when no point-vehicle pair is selected for insertion (Line 19), i.e. no new insertion is possible. Before the termination the algorithm adds the final base stations.

The reservation procedure RESERVEBATTERIES(v) performs a negotiation between the vehicles to ensure that all available batteries are shared by the whole fleet. Vehicle v is assigned one battery from each station that 1) is reachable by v and 2) has a non-reserved battery of the corresponding type. If vehicle v is unable to reserve a battery, it will not be able to continue its mission after it uses all energy capacity on board. Therefore, to receive a battery for the next flight, it initiates a negotiation for vehicles of the same type. During the negotiation such vehicles offer every reserved battery for which the following conditions are fulfilled. First, an offering vehicle must have more than one reserved battery. Secondly, it offers every battery that is at a station reachable by the initiating vehicle v. Then vehicle v selects the best offer that is a battery located at the closest station. In case of ties, offering vehicles with more reserved batteries are preferred.

The main loop starts by initializing the variables. Point $bPoint$ will be added to the last route of vehicle $bVehicle$. Their value of the evaluation function is $minValue$. If required, vehicle $bVehicle$ will renew its capacity at station $bStation$. Otherwise, $bStation$ equals $null$.

The *for*-loop iterates through all possible combinations of vehicles and points to select a feasible combination with the minimal evaluation value. Infeasible combinations are

Algorithm 1: IDIH-Reserve

```
input  : problem description
output: a set of routes $R = \{R_v | v \in V\}$
1  for $v \in V$ do
2      $R_{v,1}$.ADD($inLoc_v$);  RESERVEBATTERIES($v$);
3  repeat
4      $bVehicle \leftarrow null$;  $bPoint \leftarrow null$;
5      $bStation \leftarrow null$;  $minValue \leftarrow \infty$;
6      for $v \in V$, $p \in N_p$ do
7          if $p \neq lastPoint(v)$ then
8              if NEEDBATCHANGE($v$,$R_v$,$p$) then
9                  $st \leftarrow$COMPUTESTATION($v, R_v, p$);
10                 if $st \neq null$ and NOCOLLISION($v, R_v, p$)
                   and $g(p, v) < minValue$ then
11                     $bVehicle \leftarrow v$;  $bPoint \leftarrow p$;
12                     $bStation \leftarrow st$;
13                     $minValue \leftarrow g(p, v)$;
14             else
15                 if NOCOLLISION($v, R_v, p$) and
                   $g(p, v) < minValue$ then
16                     $bVehicle \leftarrow v$;  $bPoint \leftarrow p$;
17                     $bStation \leftarrow null$;
18                     $minValue \leftarrow g(p, v)$;
19     if $bVehicle = null$ then
20         ADDFINALDEPOTS();  return $R$;
21     if $bStation \neq null$ then
22         $R_{bVehicle}$.INITIALIZENEWROUTE($bStation$);
23     $R_{bVehicle,last}$.ADD($bPoint$);
24     UPDATERESERVATIONS($bVehicle, bStation$);
25 until false;
```

not considered due to two conditions. First, the algorithm avoids loops at the same point (Line 7). Second, the constraint NEEDBATCHANGE(v, R_v, p) checks if the vehicle v must change its battery before visiting point p to avoid exceeding maximal flight time. The change is required if 1) the vehicle v cannot reach any station after the visit or 2) it cannot change its battery in any of the reachable stations.

If a battery change is needed, an intermediate station is selected by COMPUTESTATION(v, R_v, p). The selected station must be reachable, closest to the point p and have a battery reserved by vehicle v. If there is no such station, the function returns $null$ and this point-vehicle pair is not considered.

After analyzing the need for a battery change, the point-vehicle pair is examined for collision avoidance. For this, function NOCOLLISION(v, R_v, p) checks if no other vehicle visits point p at the same time as vehicle v. Then the value of the evaluation function is computed for the point-vehicle pair. If it is lower than the best value, the pair is selected.

As mentioned, the evaluation function reflects the optimization criterion but is not equal to it (Solomon 1987). This function measures the quality of a partial solution. The goal function, on the contrary, evaluates the complete solution and, therefore, cannot be used for constructing the routes.

The evaluation function of the IDIH-Reserve is based on the following four parameters:

- distance from the current vehicle position to the point, thus, minimizing traveling distance;
- arrival time of the vehicle at the point, which gives preference to the vehicle with the shortest mission;
- time of the last visit to select the longest waiting point;
- number of visits of the point to evenly distribute observations among all points.

The time-based parameters increase with the planning horizon, whereas the other two criteria almost do not change. To eliminate this difference, we introduce relative time parameters, which relate to other time-dependent measures. Relative arrival time $\Delta art(v, p)$ is computed as $art(v, p) - min_{v' \in V} [cmt_{v'}]$, the difference between the actual arrival time $art(v, p)$ and minimal current mission time among all vehicles (see Fig. 4). Relative last visit time $\Delta\tau_p$ equals $\tau_p - min_{p' \in P} [\tau_{p'}]$, the difference between the last visit time τ_p and the minimal last visit time among all points (see Fig. 5).

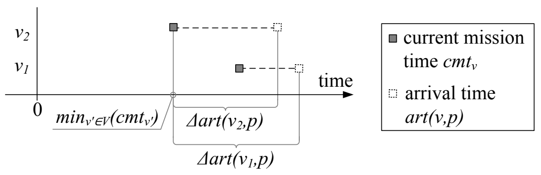
v_2
v_1
time
0
$min_{v' \in V}(cmt_{v'})$
$\Delta art(v_2,p)$
$\Delta art(v_1,p)$
■ current mission time cmt_v
□ arrival time $art(v,p)$

Figure 4: Illustration of the relative arrival time $\Delta art(v, p)$ for vehicles v_1 and v_2.

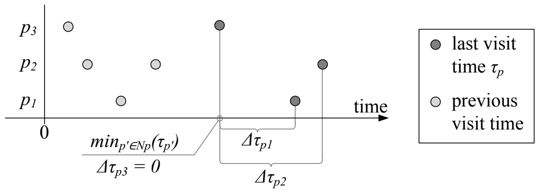

Figure 5: Illustration of the last visit time $\Delta\tau_p$ for points p_1, p_2 and p_3.

The evaluation function is computed as follows:

$$g(p, v) = \alpha_1 \cdot d(v, p) + \alpha_2 \cdot \Delta art(v, p) + \alpha_3 \cdot \Delta\tau_p/(pr_p)^\beta + \alpha_4 \cdot scale \cdot nVis_p/(pr_p)^\beta \;;$$
$$0 \leq \alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta \leq 1; \alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 = 1,$$

where $d(v, p)$ is the distance that vehicle v has to travel to point p, $nVis_p$ is the number of visits made at point p. Weight coefficients $\alpha_1, \alpha_2, \alpha_3$ and α_4 define the influence of the mentioned parameters on the final decision. Coefficient $scale$ transforms the number of visits to the same order of magnitude as the other parameters. Coefficient β determines the impact of priorities. Increasing β enhances the preference for points with higher priorities. The next section suggests the best performing values for these coefficients.

Finally, if a point-vehicle pair was selected, the heuristic performs insertion and updates the battery reservations for the selected vehicle. If a battery change is required, before adding the point, the vehicle terminates its last route at $bestStation$, and starts a new one from there (Line 22).

UPDATERESERVATIONS($bestVehicle, bestStation$) is a procedure similar to RESERVEBATTERIES(v) with the only difference that all previous reservations of *bestVehicle* must be canceled before making new ones. Since the vehicle $bestVehicle$ will use a battery from the station $bestStation$, one battery is removed from this station.

The IDIH-Reserve can be applied to a related problem without priorities and with a fixed mission time (CMPID) with minor changes. Given the maximal mission time mt, the IDIH-Reserve terminates vehicles' routes as soon as they reach this limit. If there are not enough batteries for all vehicles to fly until mt, the batteries should be used s.t. the maximal number of vehicles ($maxNVeh_t$) of type t can fly until mt. For that, the IDIH-Reserve first estimates the number $maxNVeh_t$ for each type t as $nBat_t/rNBat_t$, where $nBat_t$ is the total number of batteries of type t, $rNBat_t$ is the number of batteries required for a UAV to cover the mission time. Then the heuristic performs its basic steps with just one change. The first $maxNVeh_t$ vehicles, which started their second route, are allowed to change the batteries. The others stop traveling after exhausting their initial on-board energy capacity.

Computational Results

This section starts with a description of the used test instances and then proceeds with results of the following studies: selection of coefficients values for the proposed heuristic, comparison with optimum, and evaluation using the larger instances. All tests were performed on Intel Core i5 2.50 GHz system with 8GB RAM running Windows 7.

Test instances. To conduct a thorough evaluation of our method, four sets of instances[2] were generated: 10 small *optimum* instances, 12 *patrolling*, 48 *life* and 60 *random* scenarios. Fig. 6 depicts examples of the instances.

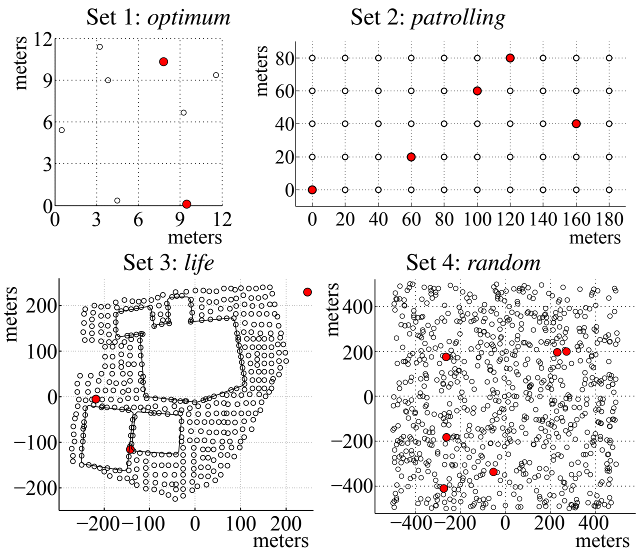

Figure 6: Examples of instances from the four test sets.

Optimum instances are small instances where it is feasible to compute an optimal solution. This set consists of 10 instances with 6 points and 2 stations. Their coordinates were selected randomly in the interval $[0, 12]$. All points are split into two clusters so that the first and the second clusters have priorities 1 and 2, respectively.

Every *optimum* scenario has 5 batteries and 2 vehicles. Due to the problem complexity, these vehicles are homogeneous. In contrast, instances of other sets have a heterogeneous fleet. The batteries are randomly allocated at the stations. Their capacity is sufficient for several overview images. The vehicles have average speed, battery change time, and service time equal to 1. Their initial location and remaining energy capacity as well as the delay lvt_p of the points are selected randomly.

Since an optimum cannot be found for common real-life scenarios, the IDIH-Reserve solutions are compared with the optimal solutions for the patrolling task. It is a related problem that relaxes some constraints, thus, providing a lower-bound for the CMPIDP optimum. The vehicles are homogeneous with *unlimited* battery capacity. The goal of patrolling is to compute a set of routes for a fleet of vehicles to repeatedly visit a set of points with minimal average delay. The *patrolling* instances are generated so that the TSP-based strategy by (Almeida et al. 2004) is optimal:

1. *Placing the points* The points are placed in a grid with 20 m step size. The total number of points is within 45–375. Each point has the same priority, and the parameter lvt_p is equal to zero.
2. *Placing the stations* The first station is placed at the left-most bottom point with coordinates $(0, 0)$. The remaining four stations are placed along the optimal TSP route with equal distances between them. Fig. 7 shows an example of a TSP-route and the corresponding stations.

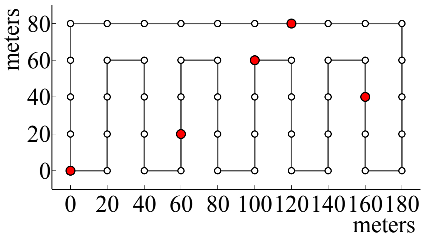

Figure 7: An example of a TSP route for generating the *patrolling* instances.

3. *Generating vehicles and batteries* One vehicle is initially located in each station. The vehicles are homogeneous and move with the average speed 1 m/s. Since patrolling ignores battery capacity limitation, this constraint is relaxed as follows. The battery capacity is equal to the travel time between the stations. The service time and the time to change the battery are equal and set to 0 s.
4. *Setting the planning horizon* The IDIH-Reserve was developed for the problem with limited number of energy resources. The patrolling strategy, on the contrary, does not set any upper bound on the planning horizon. For a fair comparison, the mission time is set to several values: the travel time of the TSP route multiplied by 2, 4, 6 or 8.

The third set (*life scenarios*) represents several scenarios that we observed in practice. To generate these instances,

[2]All sets are available online at http://uav.lakeside-labs.com/publications/test-data/planning/

first, the size of an area that can be covered by one picture is derived from the flight altitude and camera resolution. Then the minimal number of points is spread equally over the area of interest as described in (Quaritsch et al. 2010). The number of points in these scenarios ranges from 46 to 441.

The picture points of the *random* scenarios are placed randomly with coordinates in the following intervals: $[-300, 300]$ for 200–300 points, $[-400, 400]$ for 301–600 points, $[-500, 500]$ for 601–800 points.

The remaining parameters of the *life* and *random* scenarios are generated as follows. Three or six base stations are randomly allocated with coordinates in the mentioned intervals. Each scenario has 4, 7 or 8 vehicles of 2, 3 or 4 types, respectively. The number of batteries is chosen s.t. the area can be covered 3 or 6 times. The batteries are either assigned to the stations randomly or distributed among them evenly. Depending on the type, the maximal flight time with one battery is either 1200 s or 2400 s. For simplicity all vehicles have the same average speed of 2.5 m/s. Their initial location and remaining energy capacity, as well as the delay lvt_p of the points are generated randomly.

The *life* and *random* instances include 3 priority levels. To assign them, the area is split into either 3 clusters (for scenarios with up to 200 points) or 6 clusters. All points in a cluster have the same priority from the set $\{1, 2, 3\}$. For the instances with 6 clusters each priority level has to be assigned to exactly two clusters.

Selection of the coefficients values. Typically, parameter tuning results in noticeable improvements. In the following, we describe the process of selecting the best-performing coefficients for the IDIH-Reserve heuristic for both the CMPIDP and CMPID problems. For this process we focused on *life* and *random* instances. In order to avoid overfitting, every third instance was excluded. To select values of the coefficients for the CMPID, several values of the mission time were used. Tuning of coefficients is typically performed by comparing the performance of a heuristic with all possible combinations of coefficient values from a fixed domain. All coefficients (except $scale$) took values from the interval $[0, 1]$ with step 0.1. $scale$ was set to 100. The performance of a combination on an instance was measured by the cost deviation from the best obtained solution on this instance. The best combination has minimal average and minimal maximal deviation among all instances: for the CMPID $\alpha_1{=}0.4, \alpha_2{=}0.5, \alpha_3{=}0.1, \alpha_4{=}0$; for the CMPIDP $\alpha_1{=}0.2, \alpha_2{=}0.6, \alpha_3{=}0.1, \alpha_4{=}0, \beta{=}0.7$ for instances with less than 100 points and $\alpha_1{=}0.3, \alpha_2{=}0.4, \alpha_3{=}0.2, \alpha_4{=}0, \beta{=}0.7$, otherwise.

Comparison with optimum. In order to solve our *life* instances in acceptable time we have to apply heuristics which may result in non-optimal solutions. Therefore, we evaluate how far from optimum the proposed heuristic can get. For this evaluation we used the *optimum* instances. To compute the optimum, the CMPIDP problem was modeled in MiniZinc[3] and solved by Gecode[4].

The achieved solution costs are reported in Table 1. They show that the proposed heuristic is on average only 4.09 % and at most 13.66 % worse than the optimum. Moreover, the heuristic requires less than a second to solve each instance instead of minutes or hours required by the solver.

[3]see http://www.minizinc.org/

[4]see http://www.gecode.org/

Table 1: Comparison with optimum

Instance Nr.	Optimal cost	IDIH-Reserve cost	IDIH-Reserve deviation in %
1	30364	30884	1.71
2	23441	23961	2.22
3	17267	17983	4.15
4	19073	19497	2.22
5	18378	19156	4.23
6	23795	24239	1.87
7	25038	26074	4.14
8	19348	20426	5.57
9	23158	26322	13.66
10	20111	20335	1.11

Solving the CMPIDP. In the presence of priorities, areas of higher importance should be visited more often. Our study shows that the IDIH-Reserve provides such solutions with an update frequency proportional to the priority level.

The *life* and *random* instances were solved twice. In the first run, average delays for each of the three priorities were determined. The second run ignored the priority information and, thus, all points were equally important.

Fig. 8 demonstrates the results of the study. Average visit frequencies of the first run are shown as percentages of the visit frequency of the second run. These charts also show the variations among the measurements.

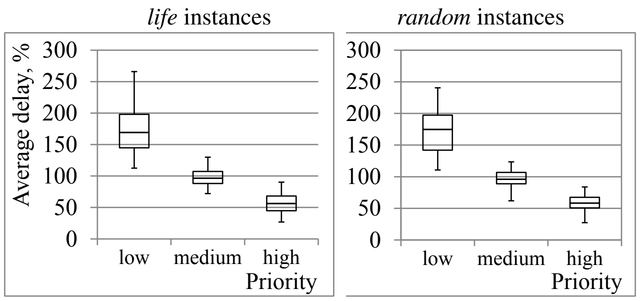

Figure 8: The average delay of each priority in the first run.

As expected, delays of the first run are approximately proportional to the priorities, i.e. around 60, 100 and 170 % for low, medium and high priorities, respectively. Moreover, delays of the medium priority are close to the delays of the second run. This means that delays between visits are not significantly affected by use of priorities. It was observed that, as the number of points increases, delays within one priority level deviate more. This is an expected consequence of the increasing complexity of the problem. Moreover, there was no significant difference between the performance on the *life* and *random* scenarios.

The computational time increases linearly with the number of points, vehicles and visits. The upper bound on the number of visits is $maxCap/minTrTime$, where $maxCap$ is the maximal total capacity of all batteries of the same type, $minTrTime$ is the minimal travel time between two points. Then the complexity of the heuristic is

$O(|N_p| \cdot |N_v| \cdot maxCap/minTrTime)$. According to the evaluation results, even the largest instances are solved in seconds, e.g. the instances with 800 points are solved at most in 10.69 s. These results show that the IDIH-Reserve heuristic can be used in time-critical applications.

Solving the CMPID. Comparison with a CMPID method allows to evaluate how efficiently the IDIH-Reserve exploits limited energy resources. The used CMPID method is the IDIH heuristic (Mersheeva and Friedrich 2013).

The study was conducted on the *life* and *random* instances. They were extended by adding the fixed mission time mt equal to $0.75 \cdot mt_{base}$, mt_{base} or $1.5 \cdot mt_{base}$. mt_{base} is the maximal possible mission time if all UAVs of the same type use the available batteries evenly. Performance of the methods is evaluated by the goal function of the CMPID.

The results for the mission time $0.75 \cdot mt_{base}$ are shown in Fig. 9. Results with other mission time values are similar and, therefore, omitted. The X-axis depicts percentage of instances that were solved by the IDIH-Reserve with the corresponding improvement over IDIH (Y-axis). The IDIH-Reserve outperformed the IDIH on almost all instances with up to 87.6 % improvement. The improvement is significant regardless of the instance parameters.

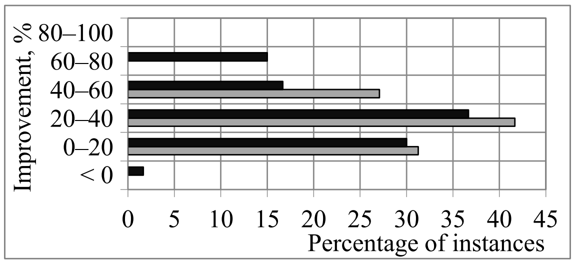

Figure 9: Comparison of the IDIH and IDIH-Reserve on *life* (black) and *random* (gray) instances with $mt_{base} \cdot 0.75$.

There are two major reasons for such significant improvement. First, due to a reservation policy in the IDIH-Reserve, all vehicles have a spare battery at every step. In contrast, the IDIH can extend a vehicle's route relying on the last battery at a station that can be taken by another drone at the next step. In this case, the first vehicle has to finish its mission as soon as its battery is discharged. This might happen at any time during the mission leading to a non-optimal use of batteries. The second reason is that the evaluation function includes relative time-based parameters that do not increase with the mission time unlike absolute parameters.

Solving the patrolling task problem. The TSP-based patrolling strategy and the tailored patrolling instances guarantee solutions with equal and minimal delays between consecutive visits. We employ these instances as stress tests to evaluate the solutions of our more general method.

The performance of the IDIH-Reserve and the TSP-based strategy is compared regarding the average delay between the visits. The box plot in Fig. 10 shows the results achieved by the IDIH-Reserve for the smallest and largest instances. The bold lines depict the delays of the TSP-based strategy. As expected due to the setup of the study, our heuristic deviates from the patrolling solution. In particular, this deviation

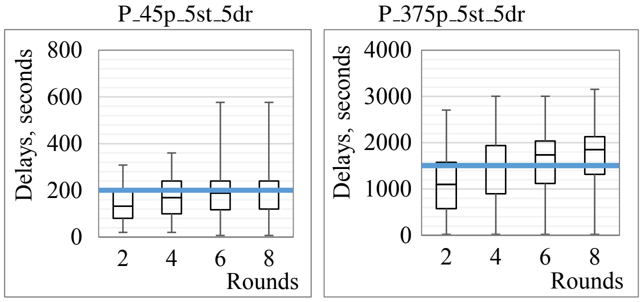

Figure 10: Average delays for the *patrolling* instances.

increases only linearly with the number of visits. The average delay is only 10.72 % above the optimum on the largest scenario with 8 rounds corresponding to 4 hours mission time.

Conclusion

This paper introduced the continuous monitoring problem with limited energy resources and priorities (CMPIDP). This problem arises in many real-life scenarios such as continuous aerial surveillance of a disaster site or a crime scene.

The CMPIDP is a variation of a well-known NP-hard problem. Therefore, we proposed a heuristic approach, the inter-depot insertion heuristic with reservations (IDIH-Reserve) to solve this problem.

The IDIH-Reserve heuristic was extensively evaluated in several studies. First, the algorithm is on average only 4 % and at most 14 % far from optimum for cases where an optimum was computable.

Second, the approach achieves good results for the real-life CMPIDP instances in short time, e.g. in less than 11 s for the largest instance with 800 points. Moreover, the method is linear in the number of points, vehicles and visits.

In a further study we evaluated IDIH-Reserve for solving a related problem (CMPID) with fixed mission time and no priorities. Our results show that IDIH-Reserve significantly outperforms the original method developed for CMPID.

Because the optima for real-life instances are not known and alternative methods for comparison are not available, the final study compares the performance of the proposed heuristic with the optimal solutions of the patrolling problem. The used instances were generated by relaxing the essential energy constraints. Even though these instances are favorable for patrolling, the average delay of the IDIH-Reserve solutions deviates by only 11% from the optimum for the largest scenarios.

We envision several directions for future work. First, the proposed approach can be enhanced further with dynamically adjusted coefficients. Another direction is to consider moving charging stations and different types of service that can be performed at points of interest by different vehicles.

Acknowledgments

This work was carried out in the scope of the research project HINT (Heuristic Intelligence) and was funded by FFG FIT-IT (grant number 840242).

We would like to thank Sergey Alatartsev and John NA Brown for their valuable contributions.

References

Ahmadi, M., and Stone, P. 2006. A multi-robot system for continuous area sweeping tasks. In *IEEE International Conference on Robotics and Automation*, 1724–1729.

Almeida, A.; Ramalho, G.; Santana, H.; Tedesco, P.; Menezes, T.; Corruble, V.; and Chevaleyre, Y. 2004. Recent Advances on Multi-agent Patrolling. In *Advances in AI SBIA*, volume 3171 of *LNCS*. Springer Berlin Heidelberg. 474–483.

Angelelli, E., and Speranza, M. G. 2002. The periodic vehicle routing problem with intermediate facilities. *European Journal of OR* 137(2):233–247.

Cannata, G., and Sgorbissa, A. 2011. A Minimalist Algorithm for Multirobot Continuous Coverage. *IEEE Transactions on Robotics* 27(2):297–312.

Elmaliach, Y.; Agmon, N.; and Kaminka, G. A. 2009. Multi-robot area patrol under frequency constraints. *Annals of Mathematics and AI* 57(3–4):293–320.

Favaretto, D.; Moretti, E.; and Pellegrini, P. 2007. Ant colony system for a VRP with multiple time windows and multiple visits. *Journal of Interdisciplinary Mathematics* 10(2):263–284.

Huynh, V. A.; Enright, J.; and Frazzoli, E. 2010. Persistent patrol with limited-range on-board sensors. In *IEEE Conference on Decision and Control*, 7661–7668.

Las Fargeas, J.; Hyun, B.; Kabamba, P.; and Girard, A. 2012. Persistent Visitation with Fuel Constraints. *Procedia - Social and Behavioral Sciences* 54:1037–1046.

Machado, A.; Ramalho, G.; Zucker, J.-D.; and Drogoul, A. 2003. Multi-agent Patrolling: An Empirical Analysis of Alternative Architectures. In *Multi-Agent-Based Simulation II*, volume 2581 of *LNCS*. Springer Berlin Heidelberg. 155–170.

Mersheeva, V., and Friedrich, G. 2012. Routing for Continuous Monitoring by Multiple Micro UAVs in Disaster Scenarios. In *European Conference on AI*, 588–593.

Mersheeva, V., and Friedrich, G. 2013. Variable Neighborhood Search for Continuous Monitoring Problem with Inter-Depot Routes. In *German Conference on AI*, 106–117.

Park, C.-H.; Kim, Y.-D.; and Jeong, B. J. 2012. Heuristics for determining a patrol path of an unmanned combat vehicle. *Computers & Industrial Engineering* 63(1):150 – 160.

Poulet, C.; Corruble, V.; and Seghrouchni, A. 2012. Working as a Team: Using Social Criteria in the Timed Patrolling Problem. In *IEEE International Conference on Tools with AI*, volume 1, 933–938.

Quaritsch, M.; Kruggl, K.; Wischounig-Strucl, D.; Bhattacharya, S.; Shah, M.; and Rinner, B. 2010. Networked UAVs as Aerial Sensor Network for Disaster Management Applications. *e&i Elektrotechnik und Informationstechnik, Special Issue on Wireless Sensor Networks* 127(3):56–63.

Santana, H.; Ramalho, G.; Corruble, V.; and Ratitch, B. 2004. Multi-agent patrolling with reinforcement learning. In *International Joint Conference on Autonomous Agents and Multiagent Systems*, 1122–1129.

Smith, S., and Rus, D. 2010. Multi-robot monitoring in dynamic environments with guaranteed currency of observations. In *IEEE Conference on Decision and Control*, 514–521.

Solomon, M. 1987. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research* 35:254–265.

Stranders, R.; Munoz De Cote, E.; Rogers, A.; and Jennings, N. R. 2013. Near-optimal Continuous Patrolling with Teams of Mobile Information Gathering Agents. *Artificial Intelligence* 195:63–105.

Stump, E., and Michael, N. 2011. Multi-robot persistent surveillance planning as a Vehicle Routing Problem. In *IEEE Conference on Automation Science and Engineering*, 569–575.

Yu, J.; Karaman, S.; and Rus, D. 2014. Persistent Monitoring of Events with Stochastic Arrivals at Multiple Stations. In *IEEE International Conference on Robotics and Automation*.