Optimisation de plan de vol de drones autonomes pour une recharge rapide de capteurs

Igor Dias da Silva, Yann Busnel, Christelle Caillouet

Optimisation de plan de vol de drones autonomes pour une recharge rapide de capteurs

Igor Dias da Silva[1] et Yann Busnel[2] et Christelle Caillouet[1]

[1] *Université Côte d'Azur, CNRS, I3S, Inria, France*
[2] *IMT Atlantique, IRISA, France*

Nous proposons une solution en deux étapes pour déterminer des trajectoires quasi-optimal de véhicules aériens sans pilote (UAV), ou drones, afin de recharger rapidement un réseau de capteurs au sol, augmentant ainsi la durée de vie de celui-ci. Nous nous concentrons sur une solution de recharge sans fil par radiofréquence (RF), lequel permet notamment de pouvoir charger plusieurs capteurs en parallèle par un même drone. Cependant, il existe une forte déperdition d'énergie lorsque plusieurs drones chargent le même capteur simultanément. Nous optimisons donc le plan de vol de la flotte de drones afin de minimiser le temps nécessaire pour charger tous les capteurs, tout en évitant cette déperdition.

Mots-clefs : Drones, Optimisation, Recharge énergétique, Internet des objets.

1 Introduction

The application of sensor networks has grown significantly in the fields of industry, military, home automation, ecology, precision agriculture, disaster management, *etc*. The majority of the aforementioned use cases require the implementation of sensor systems over long periods of time. Optimising the lifetime of each sensor is a key element in the deployment of these wireless systems and thus, we are interested in recharging the sensors' batteries to improve the lifetime of the systems and their usability.

We focus on a wireless Radio Frequency (RF) power harvesting solution. The unquestionable advantages of these solutions are flexibility, adaptability, and automation. This allows us to use mobile vehicles, such as Unmanned Aerial Vehicles (UAVs), to visit the sensors regularly to recharge them and collect their data. The objective is to define automatic and autonomous flight plans for this fleet of drones to power up energy-constrained sensor nodes.

In this work, we present a short version of the outcomes presented in [DDSBC22]. We propose a two-step optimization framework to determine the UAVs' trajectories to quickly replenish the sensor energy. Given the sensors' positions, their energy requirements, and the number of available UAVs, we first derive a Mixed Integer Linear Program (MILP) to optimise both the placement of the UAVs and the associated hovering times in order to minimise the overall charging time (excluding moving flight time). Then, in the second step, we propose a greedy algorithm to schedule the flight plans while avoiding conflicts that can lead to less efficient loads. We take advantage of the fact that one UAV can charge multiple sensors simultaneously but we avoid the efficiency loss that happens when multiple UAVs charge the same sensor at the same time [AKK16].The output is a set of UAV conflict-free trajectories fulfilling the positions computed by the MILP while minimizing the overall duration for recharging the whole system.

2 Two Step Framework

Our problem consists of determining the trajectories of a set U of UAVs, such that, by following these trajectories, all the ground sensors in a given set S will have their batteries charged. These trajectories are defined for each drone as *(i)* a list of positions the drone must visit and *(ii)* the corresponding amount of seconds the drone must remain in each position. The objective is to charge the sensors as fast as possible. To solve this multi-constraints problem, the first step of our framework consists of a mixed integer linear

program (MILP) that determines which positions the drones must visit and how long the drones should hover in these positions. The second step is to determine the order in which the positions should be visited, and avoid conflicts such as multiple drones visiting the same position at the same time.

2.1 Mixed integer linear program

The information of which positions the drones visit and for how long is modeled respectively by the binary variables y_p^u and the real variables t_p^u. These variables are defined for all available drones $u \in U$ and for all possible positions $p \in P$, where P is the discrete set of positions the drones can visit in a 3D space and U is the set of available drones. The binary variables y_p^u describe whether a drone u should be deployed at position p or not. y_p^u is 1 if the drone u is deployed at position $p \in P$, 0 otherwise. And the real variables t_p^u describe for how many seconds the drone u remains in position p.

Our main goal is to charge the ground sensors, $s \in S$, where each sensor s has an amount of energy required E_s. Therefore, the main constraint of the MILP ensures that the drones' positioning and the hovering times described by the variables y_p^u and t_p^u must allow the sensors to harvest enough energy to replenish their batteries. We use the energy model described in [ZRGD16] to compute how much energy a sensor can harvest from a drone u in a position p during t_p^u seconds. Then, we ensure that the total energy received by each sensor $s \in S$ is equal or greater than the sensor's energy requirement E_s.

Hence, a feasible solution to the MILP describes in which positions the UAVs must be deployed and for how long, while ensuring that all sensors are charged. The MILP's objective function is to minimize both the time the UAVs spend in the air charging the sensors and the number of positions they visit.

2.2 Wait-Time Scheduling Algorithm

The output of the previous linear model is a set of tasks, where a task k is given by a drone u_k, a position p_k, a duration t_k in seconds and a set of sensors S_k that are charged during the task's execution. Each drone can have multiple tasks to complete, and our goal here is to determine the order in which the drones complete their respective tasks without conflicts.

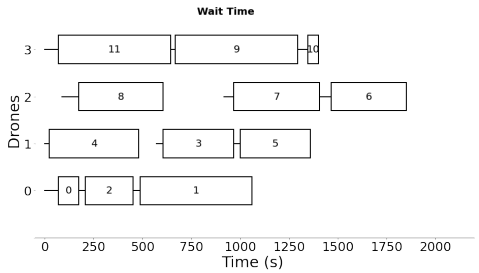
There is a *conflict* between two tasks k and k^* if *(i)* they use the same drone $(u_k = u_{k^*})$, *(ii)* they use the same position $(p_k = p_{k^*})$, or *(iii)* they charge some common sensors at the same time $(S_k \cap S_{k^*} \neq \emptyset)$ (to avoid the multi-charger constraints [AKK16]). A task is said to be *conflict-free* at a given moment if it has no conflict with any of the tasks being currently executed. We ignore possible collisions while the drones move.

Even if we ignored the possible conflicts between tasks, determining the fastest order of tasks for each drone is the same as solving the Traveling Salesman Problem (TSP) that is known to be NP-hard. We therefore present a greedy algorithm that assign the tasks in order to minimise the total time the drones have to wait hovering to avoid conflicts. More precisely, the algorithm runs as follows :

1. We order the tasks in increasing order of the wait-time (how long the UAV must wait until it can execute the task). If the task k is conflict-free, then the wait-time corresponds to the time of flight (ToF), that is, the necessary time for drone u_k to reach p_k from its previous position. However, if the task k is in conflict with some task k^* being currently executed, then its wait-time is $max(ToF, f_{k^*})$, where f_{k^*} is equal to the remaining duration before k^* ending. In some cases, when k^* uses the same drone as k, the wait-time is $f_{k^*} + ToF$, as the drone u_k will only be able to move when k^* is finished.
2. Once the tasks are ordered, if a task is conflict-free, we assign it immediately. This changes the wait-time of the remaining tasks, and they must be reordered. If a task k is not conflict-free and u_k is not executing any task, then u_k moves to p_k in advance so that it can execute k as soon as it becomes conflict-free.

An illustration of the wait-time algorithm is presented in Figure 1. The linear program has computed 12 tasks assigned to 4 drones to cover 40 sensors. The lines represent the time a drone spends flying, and the rectangles represent the period of execution of the tasks when the drones send RF signals to recharge the sensor batteries.

All drones start at the same location corresponding to a base station located at position $(0,0)$ in our experiments but our model can take into considerations multiple starting positions. The tasks are ordered



Tasks by drone

d_0	$[0,1,2]$
d_1	$[3,4,5]$
d_2	$[6,7,8]$
d_3	$[9,10,11]$

Sensors by task

0	$[35,9,18,34,26,17,12,19]$
1	$[5,13,23.1,25,28,39,22,37,27.29,321$
2	$[13,23,3,25,28,8,19,36,14]$
3	$[35,0,18,34,26,17,15,12]$
4	$[10,0,38,15,4]$
5	$[35,9,18,21,26,17,2]$
6	$[5,11,13,23,1,3,25,28,27.8,19,36,32,14]$
7	$[10,0,38,34,15,12]$
8	$[35,9,18,21,26,17,2]$
9	$[11,31,16,30,7,6,33,24,14]$
10	$[5,13,23,1,25,28,39,22,27,29,36,32]$
11	$[20,16,30]$

FIGURE 1 – Example using the wait-time algorithm to schedule the tasks in a 5x5 grid with 40 sensors and 4 drones.

following the wait-time algorithm. Task 4 has the shortest wait-time (at the beginning the wait-time is simply the time of flight), so it is the first one to be assigned. Now we must reorder the remaining tasks because their wait-time may have changed if they have a conflict with Task 4. For example, the wait-time for Task 7 is no longer its time of flight, but f_4, the time remaining for Task 4 to finish. Tasks 0 and 11 once again have the same wait-time because they are both conflict-free and have the same time of flight. Once Task 0 is assigned, we reorder the tasks and, as the wait-time for Task 11 remains the same and the smaller one, it is then assigned to drone 3.

The tasks are reordered once more, and, Task 8 is the one with the smallest wait-time since it only needs to wait for Task 0 to finish. Task 2 also needs to wait for Task 0 to finish, but since they use the same drone, it can only travel after Task 0 finishes, so its wait-time is larger than Task 8. The execution of the algorithm continues consecutively, until all tasks have been assigned and the chronogram shown in Figure 1 is obtained.

3 Results

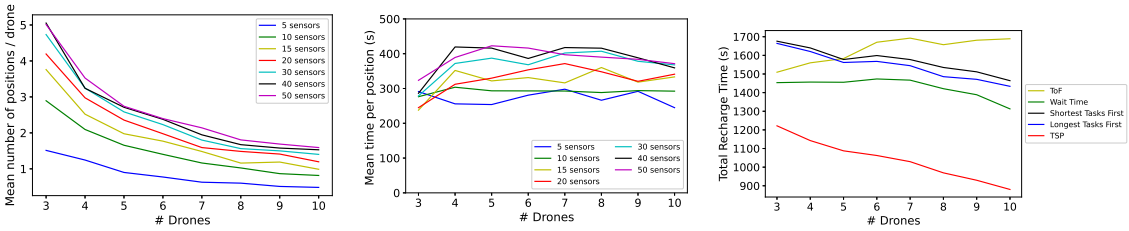
We consider a 2-dimensional area of size $50m \times 50m$ where sensors are randomly placed. We divide the area as a 5×5 grid with 5 possible altitudes, totaling 125 possible positions to deploy the drones. The number of sensors varies between 5 and 50 while the number of drones available to charge them varies from 3 to 10. Each sensor requires 150 mJ of energy.

The first step of our framework is the MILP determining the optimal positions that should be visited by the drones and for how long to recharge the battery of the sensors using RF-signals. As one can observe in Figure 2a, the more sensors to charge, the more positions the drones must visit. On the other hand, the more drones we have, the fewer positions they need to visit on average.

Figure 2b depicts the average time a drone spends in the same position. With a higher density of sensors, each drone can charge more sensors at once, but it requires a higher altitude to provide more coverage, which increases the distance between the UAV and the sensors, and consequently increases the time required in each position to charge the sensors.

Given the results of the MILP, we then apply the wait-time algorithm presented in Section 2.2 to schedule the trajectory (*e.g.* order of tasks) for the drones in order to minimise the total time of the energy replenishment problem (including time of flight and waiting time due to conflict avoidance). We also proposed other scheduling algorithms such as the optimal, the TSP, the Time of Flight (ToF) and the Shortest/Longest Tasks First algorithms.

In the optimal algorithm we consider all possible permutations of tasks and, for each permutation, we assign the tasks accordingly. From all the possible schedulings we select the optimal one with the minimum total recharge time. As for the TSP algorithm, given the set of visited positions for each drone provided by the MILP, we compute the optimal TSP solution by brute-force. This solution gives us a lower bound for the global minimum time to recharge the sensors. However, this solution is not conflict-free, meaning that the computed trajectories can allow multiple simultaneous UAV chargers for a sensor, therefore limiting



(a) The mean number of positions each drone must visit.

(b) The mean time each drone spends in the same position (tasks' mean duration).

(c) Total recharge time for all 2800 scenarios. Optimal cannot be depicted.

FIGURE 2 – MILP and scheduling results

the efficiency of the harvesting system. The ToF algorithm is similar to the wait-time algorithm except that the tasks are ordered only by their time of flight in increasing order. Finally, in the Shortest/Longest Tasks First algorithms, we order the tasks by their duration in ascending/decreasing order. For each task, if it is conflict-free, we assign it immediately. Otherwise, the drone moves at the last moment to start charging as soon as possible (*i.e.* when it becomes conflict-free).

Figure 2c compares the performance of the different algorithms. The figure illustrates how long it takes to charge all sensors by scheduling the tasks with each algorithm. As expected, we see that no greedy algorithm can charge the sensors as fast as the TSP because the TSP solution allows conflicts. But then, the wait-time algorithm outperforms all the other naive greedy approaches.

Out of the 2800 scenarios, we manage to solve the Optimal algorithm on 870 instances with less than 10 tasks because taking into consideration all possible tasks permutations consumes a lot of time. In these 870 scenarios, the optimal algorithm charges all sensors in 803.92 seconds on average. The wait-time algorithm takes on average only 3.28 % longer to charge the sensors than the optimal scheduling. The ToF algorithm takes 7.87 % longer on average, while the Longest Tasks First and Shortest Tasks First take 4.80% and 5.39% longer, respectively. In fact, we observed that all algorithms can reach the optimal solution in some instances. The wait-time algorithm reaches the optimal solution in 63.9 % of the instances. The ToF algorithm reaches the optimal solution in 59.54% of the instances, while the Longest Tasks First and Shortest Tasks First reach the optimal solution in 51.14 % and 47.01 % of the cases.

4 Conclusion

We presented a two-step framework for optimal energy replenishment of a set of sensors, using UAV as RF sources. We first solve a mixed integer linear program to determine the drones' placement in a 3D space such that it minimizes the time required to charge all ground sensors. Then, we proposed a wait-time heuristic to compute the order in which the drones should visit the deployment positions while avoiding conflicts and taking advantage of the parallelism between tasks. Avoiding these conflicts, such as multiple drones charging the same sensors simultaneously, ensure a higher energy harvesting efficiency. In the future, we would like to take the number of conflicts between tasks as a metric to be minimized in the MILP itself, to push for more parallelism which would allow us to benefit even more from a higher number of drones. The full work is presented in [DDSBC22].

Références

[AKK16] D. Altinel and G. Karabulut Kurt. Energy harvesting from multiple rf sources in wireless fading channels. *IEEE Transactions on Vehicular Technology*, 65(11) :8854–8864, 2016.

[DDSBC22] I. Dias Da Silva, Y. Busnel, and C. Caillouet. Trajectory optimization for fast sensor energy replenishment using uavs as rf sources. In *IEEE GLOBECOM*, pages 2176–2181, 2022.

[ZRGD16] D. Zorbas, P. Raveneau, and Y. Ghamri-Doudane. Assessing the cost of rf-power harvesting nodes in wireless sensor networks. In *IEEE GLOBECOM*, Washington, DC, USA, Dec 2016.