Task Allocation Algorithms for Drone Parcel Delivery Systems

Tomás Bastos[1], Bruno Guerreiro[2,1], and Rita Cunha[1]

[1] Institute for Systems and Robotics (ISR/LARSYS), Instituto Superior Técnico, Lisbon, Portugal
[2] DEEC and CTS, NOVA School of Science and Technology, Caparica, Portugal
E-mails: tomas.bastos@tecnico.ulisboa.pt, bj.guerreiro@fct.unl.pt, rita@isr.tecnico.ulisboa.pt

Abstract. This paper addresses optimization strategies for task allocation in a fleet of drones with the goal of parcel delivery. With this goal, several algorithms are discussed, focusing on the study and implementation of Task Sequential Greedy Algorithm. The proposed strategies build on this algorithm, modified to include drone battery limitations and recharge possibilities with a configurable objective function giving the user the possibility of combining and tuning both time and energy consumed. It was also modified to include relays, where parcels can change carrier during an in-flight maneuver. A binary optimization algorithm is also implemented to decide where and when the relay maneuvers are beneficial to the system. A preliminary validation of the proposed algorithms is given based on simple case studies.

1 Introduction

Multi-agent and multi-robot systems have been being widely studied in the last decades, not only due to the practical applications in real life, but also due to the constant growth of application possibilities arising with technological advances. Moreover, delivery systems and urban logistics have also experienced a great investment and effort, given the modern city problems such as excessive pollution, lack of sustainability, increasing road traffic or even due to the financial benefits of modern transportation systems. Both private companies and public entities have found in drones a viable solution not only for urban logistic problems but also for search and rescue and surveillance missions. This work is focused on studying and designing algorithms for the planning and scheduling of a set of tasks for a set of Unmanned Air Vehicles (UAVs), part of the planning and optimization challenges within REPLACE project [1], that aims to study new strategies of parcel delivery in urban environments by drones [2, 3].

Studies about Multi-agent Systems (MAS) are also becoming more frequent and precise, both addressing industrial, commercial, or security applications. Some work regarding single task robots have used Integer Linear Programming (ILP) or Mixed Integer Linear Programming (MILP), such as in [4]. In the first case, air vehicles are expected to identify, classify and target ground objects.

Because of that, the tasks need to follow a specific order, having strict time and precedence constraints. Other articles dedicated to this topic propose an approach based on Particle Swarm Optimization (PSO) [5, 6]. The authors of [7] combine PSO and a Genetic Algorithm for a weapon target task allocation for UAVs in battlefield environment. The work produced in [6] analyses scheduling in an indoor 3D situation in order to minimize the total energy consumed.

Considering cooperative task allocation, a sequential greedy algorithm is used in [8] as a baseline comparison, and some changes to that algorithm are proposed to reach better solutions, generating a Task-based Sequential Greedy Algorithm (TSGA). Further work has been dedicated to cooperative task allocation as we can observe in [9], where a Multi-Structure Genetic Algorithm is implemented in a fleet with different types of UAVs. Regarding decentralized approaches, [10] describes the implementation of Binary Linear Programming and iterative network flows and auction algorithms, whereas [11] proposes a consensus-based bundle algorithm (CBBA). Nonetheless, there are not many authors that incorporate energy limitation and recharge possibilities.

In the context of city logistics, several authors have investigated systems were many agents contribute to perform one single delivery task. In [12],the authors present an approach with a complex ILP formulation solved by two different heuristics, with the goal of using the spare capacity of existing transportation flows, incorporating the option of transfers between drivers. In parallel, relays in multi-robot systems are also under intense study in the context of Industry 4.0, where smart transportation is one of the main focuses [13]. Its applications in literature focus mainly on cooperative agents regarding communication network planning, surveillance or video covering of live events and the research regarding this subject is growing as well as its utility and relevance [14–16].

The main contributions of this paper are the proposal of 2 new algorithms for drone task allocation in a 2-D environment, based on the TSGA algorithm, that enable the use of recharging stations as well as the in-flight parcel relay maneuvers that project REPLACE promotes. The algorithm is described and validated through simple yet insightful case studies.

The remainder of this paper starts by introducing some theoretical background in Section 2, while the proposed task allocation algorithm that considers recharge stations is presented in Section 3. The algorithm considering parcel relay maneuvers for task allocation is presented in Section 4, whereas concluding remarks and future work are presented in Section 5.

2 Theoretical Background

Directed acyclic graphs (DAG) have a great relevance in the algorithm definition for the problem of cooperative parcel delivery. Considering a formal definition a graph, $G = (V, E)$ is composed by a set of vertices V (also called nodes) and a set of edges E, where each edge has two nodes associated. Edges can have directions depending on the problem formulation (for examples in a road, the direction in which the traffic flows). If this is the case, it is called a digraph and

the endpoints might be called head and tail, depending on the direction of the edge so that the direction points from head to tail.

Continuing with some useful definitions, with the analysis of a cooperative transportation problem on mind, a graph or sub-graph is a walk in the form of the sequence $\{v_0, e_1, v_2, e_2, ..., v_n\}$ if for every $n = 1, ..., n$, v_{n-1} and v_n are endpoints of e_n. A walk is said to be closed if the initial node coincides with the last node (i.e $v_0 = v_n$). Also, we call trail to a walk that has no repeated edges ($e_i \neq e_j$ with $e, j = 1, ..., n$ and $e \neq j$). Similarly, a path is a trail that does not repeat any internal node (initial and final nodes can be the same and if this is the case, we say it is a closed path). We say that a DAG is a digraph that has no cycles (no closed paths).

As a consequence, we can say that in order to be a DAG, every walk inside a graph must be a path. Notably, we can say that a graph that has directed cycles, and thus it is not a DAG, cannot be a solution of a cooperative task allocation problem.

3 Task Allocation with Recharging

The algorithms proposed in this work build on those presented in [8] and [5] to provide further functionalities. In this section, the basic algorithm and the recharging functionality will be presented.

3.1 Problem Statement

Let us imagine a 2 dimension environment with N agents (drones) and M delivery tasks. Each drone $i \in \{1, 2, 3..., N\}$ is initially characterized by its initial x and y coordinates x_i and y_i respectively and average velocity v_i. Regarding task $k \in \{1, 2, 3, ..., M\}$ initialization, the user has to provide information about its pickup and drop-off locations (x_k^p,y_k^p,x_k^d and y_k^d) and number of agents needed to perform the task Z_k.

Within this framework, $\mathbf{P}$ is defined as the optimization variable that stores the various tasks each agent will perform, in order. We can say, by this definition, that $\mathbf{P}$ is a list or vector of vectors that store the duty of each agent, in the order in which the tasks will be performed. The dimension of $\mathbf{P}$ depends on the number of agents and on the number of agents needed to perform each task. For instance, given an environment with 4 agents, with 10 tasks that need all agents to be performed, $\mathbf{P}$ will be a matrix 4×10. Also, each coalition (group of agents that will perform task k) is denoted as $\mathbf{a_k}$. For example, $\mathbf{a_2}$ refers to the group of agents that are set to perform task number 2. Thus, saying $\mathbf{P}_{2,3} = 4$ means that the task number 4 will be the 3^{rd} one performed by agent 2. Generally, $\mathbf{P}_{i,m} = k$ means that task k will be the m^{th} one performed by agent i. On the other hand saying that $\mathbf{a_k} = (m, n, o)$ means that the agents m,n and o will perform task k. Also, $t(\mathbf{P})$ is the total mission time, corresponding to the latest parcel delivery time. Besides this, G is the graph generated by the dependencies between the tasks.

Given these initializations and definitions, the formulation of our optimization problem is

$$\min_{P} \quad J = t(\mathbf{P}) \tag{1}$$

$$\text{s. t.} \quad n\left(\mathbf{a}_k(\mathbf{P})\right) = Z_k, \forall k \in \mathcal{K} \tag{2}$$

$$\text{isDAG}(G) = 1 \tag{3}$$

The second constraint ensures that G, the graph generated by the task allocation result, is a directed acyclic graph as we have defined before. The conceptual function isDAG() is simply a function that returns 1 if the input graph is a directed acyclic graph and 0 if not.

For the formulation of this problem, it is important to clarify that each task only begins when all agents needed arrive to the parcel pick up location. This means that all the others that might be available at an earlier time are waiting for that agent to arrive.

3.2 Baseline Algorithm

The first algorithm is presented in [5] and called Sequential Greedy Algorithm (ASGA) for cooperative timing missions. The coalitions in this algorithms are chosen in a greedy procedure, among all possible coalitions. For this, the algorithm computes the ETA (Estimated Time of Arrival) of all pairs *(agent,task)*, and chooses the best of these times to decide the next task to be allocated and the coalition leader for that tasks, which are the pair *(agent,task)* with minimum ETA. After this, until the sufficient number of agents, the agent with minimum ETA to that task is allocated sequentially. This procedure is repeated until all tasks have an allocated coalition. This algorithm is purely greedy and might be far from the optimum. Despite this, an important characteristic of this procedure given the cooperative nature of this problem, is that it automatically ensures that the DAG constraint is met, because each task is allocated consecutively in the end of each agent schedule, preventing the appearance of crossed dependencies.

In [8], modifications to this algorithm are proposed, leading to Task Sequential Greedy Algorithm (TSGA). It is worth noting that in ASGA, two major greedy decisions are made: (a) the next task in each step, or the order in which the allocation is performed; (b) the agent allocation within each task. The authors suggest that the first decision is more important than the second one to the overall performance of the algorithm, because it is the one that establishes the order in which each task will be performed. For this reason, the author suggests changing the order in which the tasks are being considered inside the algorithm, performing an allocation in all possible combinations, storing at each step the best result so far. This means that the first greedy decision disappears.

As stated, the second greedy decision inside each allocation is maintained and this keeps the automatic satisfaction of the DAG constraint. In essence, this algorithm runs $M!$ (M factorial) ASGAs with a specific and constrained

allocation order. For this reason, there is no need to demonstrate that this algorithm improves ASGA results, because the original ASGA allocation is for sure a subset of TSGA result, noting that the greedy order chosen for ASGA is for sure one of the $M!$ permutations. This ensures that TSGA is in the worst case as good as ASGA. Nevertheless, some numerical simulation results are shown in [8].

In Figure 1a the development data set can be observed in a graphical representation of the environment, where the red circles are the drones initial position, and the crosses are the initial (P) and final (D) points of each parcel delivery task. Regarding the number of agents needed in each task, task 1 and 4 need only one agent to be performed, task 0, 2 and 5 need two agents while three agents are necessary to perform task 3. and the velocities of the drones are set to be 0.02 distance units per time units.

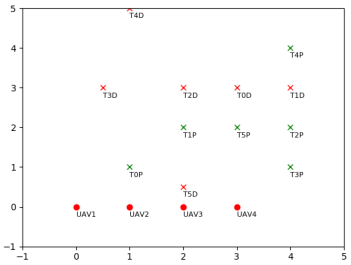
The task allocation result with the TSGA algorithm for this environment is presented in Table 1b. This notation for the task allocation result will be used along all this work, and can be interpreted as each row corresponding to each one of the agents (identified in the left column), with the result of the task allocation in the right column. The task allocation list represents the tasks performed by each drone in order. Meaning that, as an example, agent 1 in this problem will perform task 0, then 5 and finishes with task 3. In Figures 1c and 1d a graphical representation of the scheduling of the agents and the time interval of tasks execution is respectively presented.

3.3 Proposed Solution with Recharge

The next step of our implementation procedure is the inclusion of drone battery. This implementation increases the degree of reality of the environment given that all real agents have energy limitation and also enables the study of more complex missions, where the drone team does not have enough energy to complete all tasks.

Concerning how energy consumption is implemented in the code, an "energy left" parameter was created internally associated to each drone (and restored in each permutation analysis). This parameter is updated every time a task is allocated to the drone schedule. Also, before the allocation of any task, the drone is checked to determine if it has sufficient energy to both travel to the pick up point of the task and to perform the delivery itself. If not, the drone is considered unsuitable for that task as it will not be able to complete the entire task. For this reason, some modifications to the problem formulation might be needed as described in the following paragraphs. It is also worth noting that the mission is considered concluded if all the tasks are performed or alternatively, when all drones are considered not assignable to any of the remaining tasks, meaning that they do not have enough energy to perform any of them.

After implementing a limitation of 15 energy units to the agents, it was noted that the algorithm retrieved a task allocation without completing task 3. Note that task 3 is not completed. This happens because some of the allocation orders tested in the TSGA do not allow every task to be completed, but in the

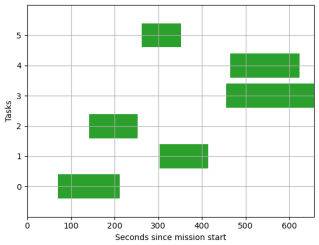


(a) Data set environment.

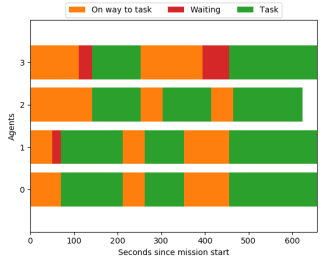
Agent	TA
1	[0,5,3]
2	[0,5,3]
3	[2,1,4]
4	[2,3]

Mission time: 656.905

(b) Allocation results.



(c) Task execution times.



(d) Agent scheduling.

Fig. 1: TSGA allocation results.

considered formulation and objective function, we are only minimizing mission time. For this reason, given all the task allocations correspondent to all the possible orders (with and without all tasks completed), the algorithm chooses the minimum mission time that logically corresponds to a mission where less tasks are performed. A quick analysis to every task allocation retrieved during the algorithm process, confirms that other allocation orders enable the completion of all the tasks. Given this particularity, triggered by the inclusion of the battery limitation, a modification to the actual formulation is proposed, because the main objective of a task allocation procedure is to complete all tasks.

Instead of minimizing the mission time, the algorithm will now be set to maximize a score function

$$f(\mathbf{P}) = k_1 \times s_{task}(\mathbf{P}) - k_2 \times s_t(\mathbf{P}) - k_3 \times s_e(\mathbf{P}) \tag{4}$$

where both the number of completed tasks and total energy consumed are taken into account. In this score function, $s_{task}(\mathbf{P})$, $s_t(\mathbf{P})$ and $s_e(\mathbf{P})$ are measures of performance regarding task completion, mission time and energy consumed respectively. The algorithm will maximize the number of tasks completed, having the mission time and energy as penalties to the score. Note that k_1, k_2 and k_3 are the weight of the task completion, mission time, and energy consumption, respectively. We can continue studying a pure TSGA by setting $k_1, k_3 = 0$, for instance. With this modification, the algorithm was able to achieve exactly the same result as before, represented in Table 1b.

The recharging of batteries is implemented with some initial considerations: the agents recharge in specific places on the map called bays; the recharge is instantaneous (e.g. battery swaps); the bays do not have a limited space for drones to recharge; when a recharge happens, the drone battery becomes full; an agent is not capable of recharging while performing a task.

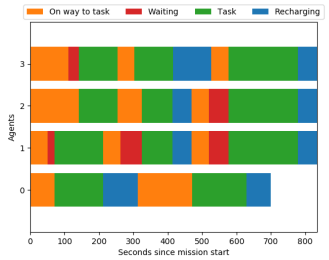
Considering the decision of when to send a drone to a recharging bay, the proposed implementation relies on the evaluation, at each time that the a task is appended to a drone schedule, if the drone will have enough energy to perform the task that the algorithm is trying to allocate and then to go to the nearest bay (relatively to the task drop off point). If that verification is false, the drone will go recharge in the nearest bay. It is ensured that the drone can reach the nearest recharge bay because it was evaluated before appending this task.

Reducing now the energy limitation of each drone to 10 energy units (that would result in a task allocation again without completion of task 3 but this time even with the modifications of the objective function), and setting the 2 bays with geographical location of the points $(x = 3; y = 1)$ and $(x = 0; y = 4)$, and with the implementation of the recharge option, results in the task allocation are shown in Table 2a and Figure 2b, where "r" represents a recharge task. Also note that the algorithm (as an implementation option) forces the drones to end the mission in a bay.

Agent	TA
1	[0,r,4,r]
2	[0,5,r,3,r]
3	[2,5,r,3,r]
4	[2,1,r,3,r]

Mission time: 834.29

(a) Allocation results.



(b) Agent scheduling.

Fig. 2: TSGA-recharge allocation results.

4 Relay Maneuvers

In this section, relay maneuvers will be implemented in the task allocation algorithm. With the relay points, the tasks are from now on considered to be composed by various sections that will be processed as separated tasks.

4.1 Problem Statement

Take in consideration the examples depicted in Figure 4b we are able to see several tasks, where task 0 is divided in 3 different minor tasks by the means of 2 relay points, whereas task 1 does not have relay points and thus, is not divided.

In order to ensure language coherence, let us detail some useful definitions. From this point on the total task will be called *task* (green dashed line in the graphical environment), and the referred minor tasks will be named *relay task*. A *relay point* is the place where a relay maneuver is set to happen (represented with blue crosses in the graphical environment). All relay tasks are associated to a original task, and the notation used will be "X.Y" where "X" relates to the original task and "Y" represents that the relay task is the Y-th inside of the original task.

In order to better formulate the problem, some assumptions should be made. We consider 100% success of performed relay maneuvers, relay maneuvers are only performed in tasks that require only one agent, and the receiving drone should reach the relay point simultaneously or earlier than the drone that is providing the parcel. Given these assumptions, the optimization problem relies on finding the best task allocation for each drone given all the relay tasks.

In addition to the task allocation problem formulated in the previous section, we have now the decisions of where to locate the relay points as well as how many relay points in one single task are necessary.

4.2 Proposed Solution

Two of the main focuses of the proposed solution are to maintain a centralized algorithm and to create a possibility for the utilization of the work performed developed before, namely the TSGA algorithm.

As stated before, the two decisions of where and when to perform the relay must be found by the algorithm. However, it is possible for the programmer to give some possibilities of relay points locations to the algorithm and these two decisions would be taken on top of those possibilities. This is of course a relaxation of the problem and its constrains, narrowing the possible locations of the relay points, and reducing the optimality of the result. Nonetheless, the quality of the solution can be increased with the refining of the grid as long as the necessary computational power is available.

This strategy can be implemented using binary optimization algorithms, where a binary variable would be created to encode an active or inactive relay point. The decision variable vector would be the vector made of all these binary variables to be set by the binary optimization algorithm, resulting in the best places in the map for performing a relay maneuver. The proposed algorithm is depicted in Figure 3.

4.3 Implementation and Discussion

It is further assumed that for one task with m relay points, the same drone is not able to perform more than one relay task. This decision has the consequence

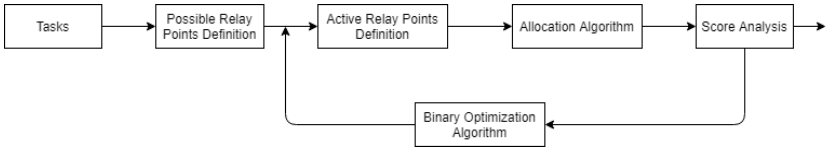


Fig. 3: Relay Point Binary Optimization Diagram

that the number of drones of the mission should be higher than the maximum number of relay points in a single task. So, the first analysis will focus on a simple case of a single task with 2 relay points with 3 drones, as shown in Figure 4a, where a single task with 2 relay points and 3 UAVs in their initial positions are represented. For the analysis and comparison of time of the missions, let us assume that the speed of the drones is 0.02 distance units per time units.

(a) Basic environment

(b) Multi-task environment

Fig. 4: Relay allocation development environments.

In a first approach to create the algorithm that will dictate which drone is going to perform each one of the segments of the tasks, a greedy procedure was implemented from the beginning to the end of the task. This means that the procedure is to sequentially look for the closest drone for the relay task and allocate that segment to that agent. With this approach, the task allocation result would be drone 1 to perform relay task 0.0, drone 2 to perform relay task 0.1 and drone 3 to perform relay task 0.2, with a total mission time of 200 time units and a total distance of 7.618 distance units.

However, one can observe that it is possible to achieve a solution that delivers an equal mission time of 200 time units with a lower total distance traveled by the agents, which implies a reduced cost if we consider both the goals of minimizing time and consumed energy. This solution would be allocating UAV3 to the first relay point (second segment of the task) and UAV2 to the second relay point (third and final segment). To achieve this result based on the conceptualized algorithm, some changes must be implemented. The algorithm will from now on, before allocating the relay tasks to each drone, choose the best coalition, i.e

the group of drones, to perform the task. This is going to be done by choosing the closest available agents to each segment of the task, forming the coalition. Inside this coalition, the algorithm will test all permutations of the chosen agents to each relay point and choose the best allocation, by minimizing the objective function

$$f_1(\mathbf{a_k}) = k_1 \times s_{time}(\mathbf{a_k}) + k_2 \times s_{energy}(\mathbf{a_k}) \tag{5}$$

This optimization level, where within 1 single task the best coalition of drones is chosen and allocated to the relay tasks, is referred to as *micro allocation cycle*.

The allocation result after the implementation of the *micro allocation cycle* is drone 1 to perform relay task 0.0, drone 2 to perform relay task 0.2 and drone 3 to perform relay task 0.1, with a total mission time of 200 time units and a total distance of 6.920 distance units.

The next step in complexity is adding more than one task. As we have analyzed, the order in which tasks are allocated is one of the key factors for the performance of this allocation strategy and for this reason, an optimization cycle iterating the order in which the tasks are considered is going to be implemented, following the idea behind TSGA. This cycle of task order iteration is referred to as *macro allocation cycle*. The objective is to minimize an objective function that will take into account the total mission time, total distance covered by the drones, as the aforementioned f_1, given by

$$f_2(\mathbf{P}) = k_3 \times s_{time}(\mathbf{P}) + k_4 \times s_{energy}(\mathbf{P}) \tag{6}$$

Considering the multi-task environment depicted in Figure 4b, the impact of the *macro allocation cycle* can be analyzed in the results presented in Table 1, where we have the results with and without the *macro allocation cycle*.

Table 1: Relay Task Allocation Results: no *macro* cycle

(a) Without *macro* cycle

Agent	TA
1	[0.0; 1.0; 3.0; 4.1; 5.1]
2	[0.2; 2.1; 4.0; 5.2]
3	[0.1; 2.0; 3.1; 5.0]

Mission time: 937.63
Total distance: 45.06

(b) With *macro* cycle

Agent	TA
1	[4.0; 0.0;1.0; 5.0]
2	[3.1; 0.2; 2.1; 5.2]
3	[3.0; 4.1: 0.1; 2.0; 5.1]

Mission time: 761.73
Total distance: 37.66

After observing these results, the last optimization cycle regarding the relay points was considered. It is worth mentioning that this binary optimization of the relay points is performed over a finite set of possible relay points, predefined by the user and that the intention of this optimization phase is to find the ones that optimize mission time and energy. The chosen algorithm to perform the binary optimization is a genetic algorithm. The global architecture of the proposed solution is given in Figure 5.

Considering no battery limitations and with an homogeneous fleet there is no incentive for the system to call for relay maneuvers as there is no time or energy saving in this procedure. However, this algorithm can prove its utility in the study of heterogeneous fleets, where the algorithm tends to maximize the relay tasks performed by the fast or more efficient drones. Furthermore, it is possible to establish a minimum number of relay maneuvers, and an example considering at least 3 relay maneuvers was tested for algorithm validation. The result of the 3 best relay points, optimized over the possible points is shown in Figure 6.

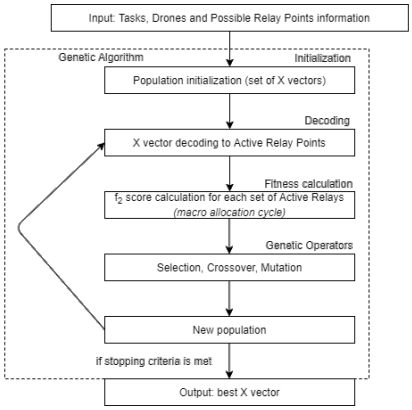


Fig. 5: TSGA-relays algorithm.

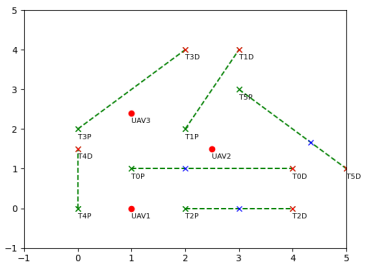


Fig. 6: TSGA-relays algorithm results with 3 relay points.

5 Conclusion

The present work combines multi-agent systems study applied with modern urban logistics, given that it is aimed to study a multi-drone cooperative parcel delivery system. The main objectives were accomplished by proposing and implementing two new algorithms that have recharge possibilities and allow the relay maneuvers. A preliminary validation of the algorithms was accomplished through simple yet informative study cases.

Regarding future work, several simple modifications can be considered, such as task priority, more evolved energy consumption models, or considering a 3-D environment. Another opportunity for further improvement is to consider decentralized variations of the current algorithm, as the one limitation of the proposed algorithms is their computational complexity.

Acknowledgments

This work was partially funded by FCT project REPLACE (PTDC/EEIAUT-/32107/2017) which includes Lisboa 2020 and PIDDAC funds, project CAPTURE (PTDC/EEI-AUT/1732/2020), and also projects CTS (UIDB/00066-/2020) and LARSYS (UIDB/50009/2020).

References

1. Bruno Guerreiro. Replace project. URL: http://replace.isr.tecnico.ulisboa.pt (accessed: 01.03.2022).
2. João Pinto, Bruno J Guerreiro, and Rita Cunha. Planning parcel relay manoeuvres for quadrotors. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 137–145. IEEE, 2021.
3. Francisco Matos and Bruno Guerreiro. Model predictive control strategies for parcel relay manoeuvres using drones. In *2021 International Young Engineers Forum (YEF-ECE)*, pages 32–37. IEEE, 2021.
4. Corey Schumacher. UAV Task Assignment with Timing Constraints. 2003.
5. Gyeongtaek Oh, Youdan Kim, Jaemyung Ahn, and Han Lim Choi. PSO-based Optimal Task Allocation for Cooperative Timing Missions. 2016.
6. Yohanes Khosiawan and Izabela Nielsen. Indoor UAV scheduling with Restful Task Assignment Algorithm. 2017.
7. Jia-lei Liu, Zhi-guang Shi, and Yan Zhang. A New Method of UAVs Multi-target Task Assignment. 2018.
8. Gyeongtaek Oh, Youdan Kim, Jaemyung Ahn, and Han-Lim Choi. Task Allocation of Multiple UAVs for Cooperative Parcel Delivery. In *Advances in Aerospace Guidance, Navigation and Control*. 2018.
9. Nuri Ozalp, Ugur Ayan, and Erhan Oztop. Cooperative multi-task assignment for heterogonous UAVs. 2015.
10. Phillip R. Chandler, Meir Pachter, Steven R. Rasmussen, and Corey Schumacher. Multiple task assignment for a UAV team. 2002.
11. Andrew K. Whitten, Han Lim Choi, Luke B. Johnson, and Jonathan P. How. Decentralized task allocation with coupled constraints in complex missions. 2011.
12. Wenyi Chen, Martijn Mes, and Marco Schutten. Multi-hop driver-parcel matching problem with time windows. 2018.
13. Ocident Bongomin and Aregawi Yemane. The hype and disruptive technologies of industry 4.0 in major industrial sectors: A state of the art. 2020.
14. Jesús Sánchez-García, JM García-Campos, Mario Arzamendia, D Gutierrez Reina, SL Toral, and D Gregor. A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications. 2018.
15. Sameera S Ponda, Luke B Johnson, Andrew N Kopeikin, Han-Lim Choi, and Jonathan P How. Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. 2012.
16. Andrew N Kopeikin, Sameera S Ponda, Luke B Johnson, and Jonathan P How. Dynamic mission planning for communication control in multiple unmanned aircraft teams. 2013.