Task Allocation Algorithms for a Cooperative Drone Parcel Delivery System

Tomás Miguel Bernardo Bastos
tomas.bastos@tecnico.ulisboa.pt

Instituto Superior Técnico, Universidade de Lisboa, Portugal

December 2020

Abstract

Multi-agent and multi-robot systems have been being widely studied in the last decades, not only due to the practical applications in real quotidian 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, developed in the context of a research project regarding drone parcel transportation systems, addresses a study of optimization algorithms for task allocation in a fleet of drones with the goal of parcel delivery. With this goal, many algorithms are discussed, focusing on the study and implementation of Task Sequential Greedy Algorithm. This algorithm was 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 implemented to decide where and when the relay maneuvers are beneficial to the system.

Keywords: Optimization; Drone transportation systems; Task allocation; Relay maneuvers; Cooperative systems.

1. Introduction

1.1. Motivation

This work, developed within the scope of the project REPLACE, focuses 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, that aims to study new strategies of parcel delivery in urban environments by drones. The inclusion of relay maneuvers in the cooperative parcel delivery is one of the main and disruptive key steps undertaken by the research team during the development of this work.

Numerous applications of Multi-Robot Systems have been emerging in our society and are becoming more and more generalized, mainly in situations where some places cannot be easily reached or there are human lives in danger, such as helping in preventing and fighting wild fires. Further information and a review of the most recent applications and projects regarding this subject can be found in [1].

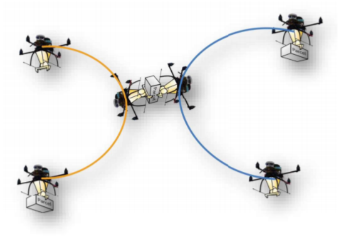
Figure 1: Relay maneuver

1.2. State-of-the-Art

Studies about Multi-agent Systems (MAS) and more specifically Multi-Robot Task Allocation (MRTA), which include UAVs, are also becoming more and more frequent and precise, both with industrial, commercial or security applications.

Some work regarding single task robots (robots that are not able to perform more than one task simultaneously) have used Integer Linear Programming (ILP) or Mixed Integer Linear Programming

(MILP), such as in [2]. 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), as we can see in [3], and [4]. The authors of [5] combine PSO and a Genetic Algorithm for a weapon target task allocation for UAVs in battlefield environment. The work produced in [4] analyses scheduling in an indoor 3D situation in order to minimize the total energy consumed.

Even though the majority of the approaches we see in the literature are single robot tasks (tasks that are performed by only one agent), we will focus mainly in cooperative systems. One article that analyses a cooperative task allocation using also PSO is [3]. Sequential Greedy Algorithm is used in [6] as a baseline comparison, and some changes to that algorithm are proposed to reach better solutions, generating a Task based Sequential Greedy Algorithm that will be very important in the development of this. On top of that, more work have been being dedicated to cooperative task allocation as we can observe in [7], where a Multi-Structure Genetic Algorithm is implemented in a fleet with different types of UAVs.

Many stages and designs can be implemented using decentralized approaches, with the main advantages of time and computational resources efficiency, operation under unknown and uncertain environments and robustness with respect to model uncertainty. On the other hand, centralized algorithms often demand high computational resources for big missions, and so they commonly have a problem regarding scalability. For example, using a specific type of ILP, [8] describes the implementation of Binary Linear Programming and iterative Network Flows and Auction algorithms. Furthermore, a distinct decentralized approach is presented in [9]: Consensus-Based Bundle Algorithm (CBBA). Despite the numerous publications and studies regarding drone cooperative task allocation for parcel delivery, there are few 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, with a change in the carrier during the task performance itself. These approaches are called multi-hop parcel delivery systems and are not very-well studied despite having great advantages and potential if one can create an efficient system. In [10], Chen *et al* 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 - [11]. 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 [12]. In communicative systems, the planning of distance between agents is crucial, once the antennas have a limited transmission and reception radius. Examples of this are the works performed by Kopeikin, Ponda *et al* in [13] and in [14].

It is also worth noting that the implementation of MRTA (Multi-Robot Task Allocation) problems are divided in two main areas: the decision making of the group of agents and its execution. The presented work focuses only on the first part. This means that the drones are assumed to have the capability of performing the trajectories planned, not colliding with each other. The mechanisms to well perform attitude control and collision avoidance systems are assumed to be well implemented in the agents. More over, the drones are considered to be in a 2 dimensions plane.

2. Theoretical Background

2.1. Directed Acyclic Graphs

Directed acyclic graphs 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. In the example, Figure 2, nodes are represented by the letters u, v, w and x while edges by the letters from a to f. Endpoints are defined to be the nodes that delimit each edge, e.g u and v are endpoints of a. Two edges are adjacent if they have an endpoint in common. 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 the head to the tail.

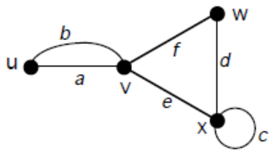

Figure 2: Graph example [15]

Continuing with some useful definitions, with the analysis of a cooperative transportation problem on mind: we say that 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. In other words, a graph is a continuous sequence of adjacent edges. A walk is said to be closed if the 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 directed acyclic graph (also 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.

Let's now depict an useful example related to a task allocation problem. Let's assume we have 2 drones a_1, a_2 and 3 tasks t_1, t_2, t_3 to be performed. Task 1 only needs 1 agent to be performed whilst in tasks 2 and 3, both drones are needed to complete the tasks (because of the weight of the parcel for instance). Let's say that the vector V_n stores the output of the ordered tasks to be performed by the agent n after some task allocation decision was made: $V_1 = (1, 2, 3)$ and $V_2 = (3, 2)$, meaning that drone 1 performs task 1, then task 2 and in the end task 3 (represented in red), and that drone 2 performs task 3 and then task 2 (represented in green).

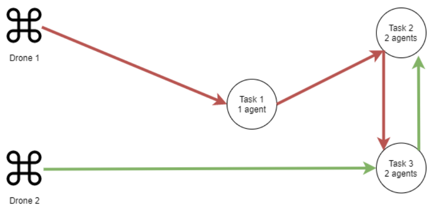

Figure 3: Example of task allocation representation

We can directly analyze the dependency of the tasks: from drone 1 path, task 3 depends on task 2 to be executed, and task 2 depends on task 1. On the other hand, given drone 2 task allocation, task 2 is dependent on task 3. We can easily observe that task 2 and task 3 have a mutual dependency and thus, it is an impossible solution for our problem. In the following image we can observe the time dependency of the tasks on this example, knowing that an edge directed from t_n to t_m means that task m needs task n finished to begin (time hierarchy).

This graph has directed cycles and thus it is not a DAG, which is the reason why it cannot be a solution of a cooperative task allocation problem, has we have seen before.

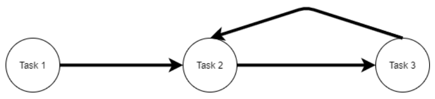

Figure 4: Task dependency graph

3. Task Allocation for Parcel Delivery with Recharging

After careful review of the state-of-the-art and selected literature, it was considered that a good first step would be to understand and implement the algorithms presented in [6] and [3] by Oh, Kim, Ahn and Choi.

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, to each coalition (group of agents that will perform task k) the author calls $\mathbf{a_k}$. 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 as analysed in Figure 4.

Given these initializations and definitions, the formulation of our optimization problem is as follows, according to [3]:

$$\begin{array}{ll} \text{minimize} & J = t(\mathbf{P}) \\ \text{subjected to} & n\left(\mathbf{a}_k(\mathbf{P})\right) = Z_k, \forall k \in \mathcal{K} \\ & \text{isDAG}(G) = 1 \end{array} \tag{1}$$

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. Proposed Solutions in the Literature

The first algorithm is presented in [3] 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 [6], 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. Those are:

- The next task in each step, and thus the order in which the allocation is performed;
- 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 [6].

In the next figure 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.

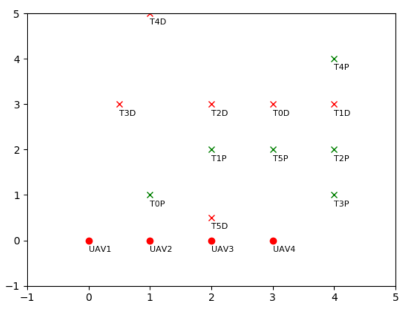

Figure 5: Development data set environment

The task allocation result with the TSGA algorithm for this environment is as follows:

Task Allocation Result - TSGA

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

Mission time: 656.905

Table 1: Task allocation 1

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 6 and 9 a graphical representation of the scheduling of the agents and the time interval of tasks execution is presented.

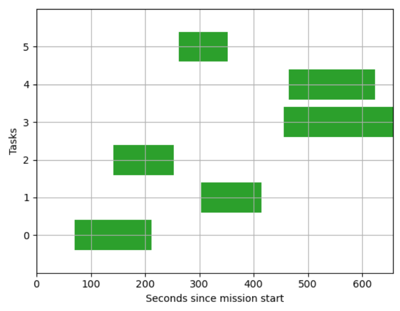

Figure 6: Task Execution Times

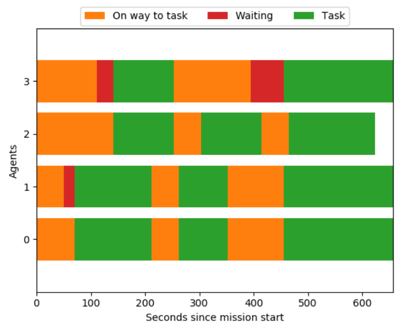

Figure 7: Agent Scheduling

3.3. Modifications discussion and implementation

The next step of our implementation procedure is the inclusion of a battery limitation on the drones. This implementation increases the degree of reality of the studied environment given that there are no real agents with no 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, to 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 Task Sequential Greedy Algorithm do not allow every task to be completed, but in the 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 $f(\mathbf{P})$, where both the number of completed tasks and total energy consumed are taken into account, such that:

$$f(\mathbf{P}) = k_1 \times s_{task}(\mathbf{P}) - k_2 \times s_t(\mathbf{P}) - k_3 \times s_e(\mathbf{P}) \quad (2)$$

where $s_{task}(\mathbf{P})$, $s_t(\mathbf{P})$ and $s_e(\mathbf{P})$ are measures of performance regarding task completion, mission time and energy consumed respectively. Given this score function, the algorithm is maximizing 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 and that can be tuned to the preferences of the user. 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 1 showing that the implementation was successful.

Following the line of thought of trying to reach simulation conditions as close as possible to the real ones, the next step is to create the possibilities for the drone to recharge batteries.

This feature will be implemented with some initial considerations: the agents recharge in specific places on the map called bays; the recharge is instantaneous (one can think of a battery swap, as a parallel to a real process) and do not have operational time of approaching and leaving the bay; this means that the drone recharges just by equalling its coordinates to the coordinates of a charging bay; the bays do not have a limited space for drones to recharge, meaning that any bay is always available for any drone to recharge and there are no queues or waiting on the recharge process; when a recharge happens, the drone battery becomes full; an agent is not capable of recharging while performing a task.

The biggest decision at this point is when to send a drone to recharge, i.e, when should the drone be considered with "low battery" and a recharge task added to the allocation. Two hypothesis were considered: the first one was the definition of a minimum threshold below which the drone would be sent to recharge. For example, every time a drone drops below 3 energy units, it would go to a recharge bay. This option clearly lowers the optimality of the algorithm. To prove this, one might just imagine a small task that would require less energy than the minimum energy threshold, beginning exactly where the drone is and in the direction of the recharge bay. Also, this option gives the chance to a drone to run easily without any energy left. Considering these limitations, 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 (relatively to the point he is at). 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 2, 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.

Task Allocation Result

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

Table 2: Task allocation w/ recharge

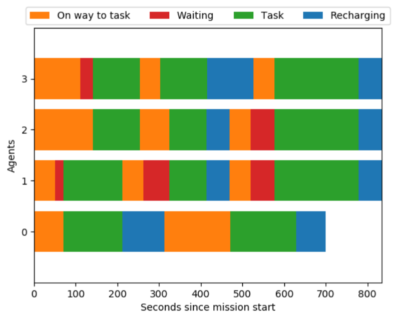

Figure 8: Agent Scheduling with recharge

4. Relay Maneuvers

4.1. Problem Description

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. Take in consideration the next example.

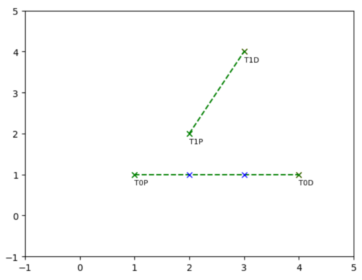

Figure 9: Agent Scheduling with recharge

In Figure 9 we are able to see 2 different tasks. 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). A more rigorous definition of *relay task* can be: a relay task is a task limited by at least one relay point. Additionally, to enumerate and refer the relay tasks, a new notation will be used. As noted, all relay tasks are associated to a original "big" task, and the notation used will be "X.Y" where "X" relates to the original task associated with the relay task (global numeration) and "Y" represents that the relay task is the Yth inside of the original task (internal numeration in each original task).

In order to better formulate the problem, some assumptions should be made: in the first place, it is considered that 100% of the performed relay maneuvers are successful; it is also assumed that relay maneuvers are only performed in tasks that require only one agent; this consideration does not come with the intention of decreasing the complexity of the problem but instead some increasing the real applications of the study since it would be very difficult to create the transmission mechanisms for transferring a parcel carried by more that one drone; finally, it is also supposed that for a relay maneuver to take place, the receiving drone should reach the relay point earlier than the drone that is providing the parcel (or simultaneously). Given these assumptions, the optimization problem relies on finding the best task allocation for each drone given all the relay tasks.

It is worth recalling the inputs that the algorithm is expected to receive that are only a set of tasks and the characterization of the drones, meaning that the user is not expected to define the relay points. As a consequence, the algorithm is expected to (optimally) define the relay points it will use for each one of the tasks (if any) according to the drones limitations and the specific geometry of the mission.

The more severe problems that arise are resultant of the algorithm decision of where to locate the relay points as well as how many relay points in one single task and , given the relay points, when to trigger or ask the relay maneuver. These problems were by far the most difficult to address and to produce the final solution, and so these two problems are considered to be the core of an algorithm with relay maneuvers and where many of them can have significant contributions.

After careful thinking and numerous tests about the implementation and correctness of some strategies, the conclusion is that it is difficult to establish *a priori* a generic rule, or chain of rules or even a deterministic strategy that would be successful tackling the two aforementioned problems, given that deterministic algorithm would need to decide how many relay points in each task, the position of the relay points and the reason to call the relay maneuver. It would be extremely difficult or almost impossible to build a deterministic algorithm that would ensure that the decision making was the best for the global mission (one is able to imagine a mission with a big number of tasks) and not only to a small system near to the point where the decision is being taken. For this reason, a non-deterministic approach will be proposed.

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. However, the programmer is in practice creating a grid, or a mesh, on top of which the algorithm is going to optimize. This means that the quality of the solution can be increased with the refining of the grid as long as computational power is available.

This strategy would be programmed to be solved using binary optimization algorithms. For this, after the definition of the possible location to the relay points, associated with each one of those points, a binary variable would be created, encoding an active or inactive 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. For the study to be conductive to an appropriate solution, the programmer should run some simulations with different possible relay points (considering the available computational power) in order to understand if the task division is not limiting the optimality of the solution.

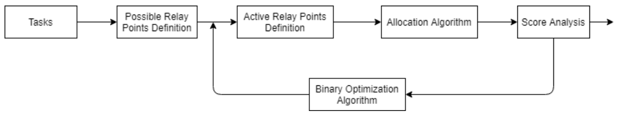
Tasks
Possible Relay Points Definition
Active Relay Points Definition
Allocation Algorithm
Score Analysis
Binary Optimization Algorithm

Figure 10: Relay Point Binary Optimization Diagram

4.3. Implementation and Discussion

First of all, an important assumption was made: for one task, with m relay points, the same drone is not able to perform more than one relay task, i.e, more than one segment of the task. One drone is not able to perform the path from the pick up point to the first relay point, for example, and after picking up again the parcel in any of the ensuing segments. This decision was taken bearing in mind that it is considered that the velocity of the drone is constant (with or without parcel). This implies that the drone that transported the parcel from the pick up point to the first relay point, in order to be able to be on time in another relay point forward, would have to assume the trajectory of the task itself (the straight line) because the other drones, with the same velocity, will carry the parcel in that trajectory. Note that "being on time" means that the reaching drone will not have to wait for another drone in the relay point in order to perform the relay maneuver. Thus, it would not make sense in a way that if the drone is doing the parcel trajectory itself, that drone should be carrying the parcel and the relay maneuver should not have happened in the first place. This decision has a considerable consequence which is that the number of drones of the mission, should be no lower than the maximum number of relay segments in a single task, or in other words, 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 the next figure, 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.

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.

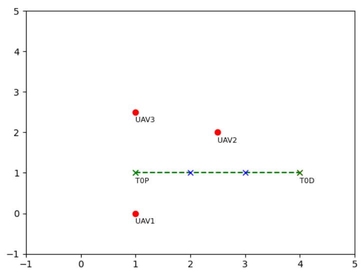

Figure 11: Relay development environment

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.

It is worth noting that the algorithm was coded to wait for the limiting agent to deploy the task. This means that if one of the above agents was far away from the task (and no other was closer) the algorithm computes the time that the farthest agent takes to reach the attributed relay point and only deploys the task at the right time to assure the time synchronization of the mission. This means that the algorithm will not leave any task not performed, as long as the unlimited battery of the drones is kept, and no feasibility condition is broken.

However, given the above map, 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 travelled 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). Note that concerning elapsed time this allocation would not modify the result as UAV3 reaches the first relay point before UAV1 with the parcel and UAV2 reaches the second relay point also before the parcel. In any case, it "saves" energy.

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, an optimization procedure will be run: the algorithm will test all permutations of the chosen agents to each relay point and choose the best allocation, by the means of a minimization of an objective function:

$$f_1(\mathbf{a_k}) = k_1 \times s_{time}(\mathbf{a_k}) + k_2 \times s_{energy}(\mathbf{a_k}) \quad (3)$$

This optimization level, where within 1 single task the best coalition of drones is chosen and allocated to the relay tasks will be called *micro allocation cycle* from now on to denote the difference to the optimization cycle that will be discussed next.

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 analysed, 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. To this cycle of task order iteration we will, from now on, call *macro allocation cycle* as opposed to the *micro allocation cycle*, defined before. To sum up the differences, and it is important that the reader notes that they are two independent cycles and levels of optimization, *micro allocation cycle* is done to allocate the agents to the best relay points inside one single task, whereas *macro allocation cycle* tests the order in which to allocate the tasks, given a set of tasks.

In this allocation cycle, the objective will also be minimizing an objective function that will take into account the total mission time, total distance covered by the drones, as the aforementioned f_1, such that:

$$f_2(\mathbf{P}) = k_3 \times s_{time}(\mathbf{P}) + k_4 \times s_{energy}(\mathbf{P}) \qquad (4)$$

Next one can see another environment example and the impact of the *macro allocation cycle* in the task allocation algorithm by analysing two task allocation results with and without it.

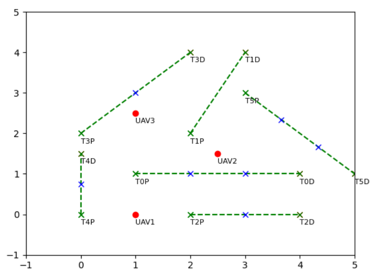

Figure 12: Multi task relay development environment

Task Allocation with no *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

Table 3: Relay Task Allocation 1

Task Allocation 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

Table 4: Relay Task Allocation 2

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, given a set of specific locations where relay maneuvers can happen, find the ones that optimize mission time and energy. The chosen algorithm to perform the binary optimization is a genetic algorithm as it is an algorithm that is well studied and has numerous optimization parameters that the user can change in order to perform relevant studies. In Image 13, the global architecture of the problem can be observed.

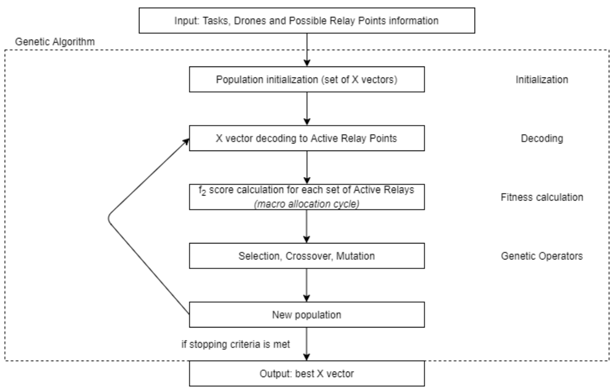
Input: Tasks, Drones and Possible Relay Points information
Genetic Algorithm
Population initialization (set of X vectors)
Initialization
X vector decoding to Active Relay Points
Decoding
f_2 score calculation for each set of Active Relays
(macro allocation cycle)
Fitness calculation
Selection, Crossover, Mutation
Genetic Operators
New population
if stopping criteria is met
Output: best X vector

Figure 13: Architecture of TSGA with Relays with Relay Points Genetic Optimization

After some testing on different data sets, it was concluded that with no battery limitation of the drones and with an homogeneous fleet, since 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. Another example of application can be seen below, where a minimum of 3 relay maneuvers was established. The result of the 3 best relay points, optimized over the possible points shown in Figure 14, is presented below.

5. Conclusions

5.1. Contributions

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 since an already known algorithm, Task Sequential Greedy Algorithm, was implemented and modified to have not only recharge possibilities but also the relay maneuvers.

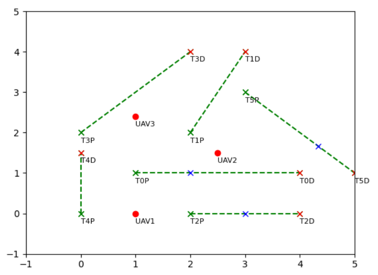

Figure 14: Result of the optimization

This work also had the objective of producing a generic algorithm, meaning that the algorithm and the framework are coded in a way that any type or size of the data sets can be considered as inputs. In practice, any fleet of drone (both homogeneous and heterogeneous) and any type of tasks (with diverse weights or lengths) can be studied in this algorithm. Both stages of the algorithm were characterized from a evaluative point of view, meaning that the user has the information about the performance of the algorithm in certain conditions and also its limitations.

5.2. Further work

Firstly, a modification that aims to provide the algorithm with the possibility to evaluate the task according to their priority in time can be implemented such as in the baseline article. Another good implementation for the algorithm would be time varying velocity and consumption. Another modification that would provide a more realistic simulation environment would be to consider a 3D space or the inclusion of a real drone model.

One of the biggest limitations of the algorithm we implemented and modified is its combinatorial nature. Data clustering techniques can be implemented in order to tackle this problem. Also, an interesting approach would be given by a multi-objective optimization procedure.

To conclude, the study of a decentralized approach to this algorithm would be of interest, not only to study the differences regarding the system response but also because this modification would allow for the usage of bigger data sets without escalating in computational effort or run time, thus improving the scalability of the strategy.

References

[1] Robin Kellermann, Tobias Biehle, and Liliann Fischer. Drones for parcel and passenger transportation: A literature review. 2020.

[2] Corey Schumacher. UAV Task Assignment with Timing Constraints. 2003.

[3] Gyeongtaek Oh, Youdan Kim, Jaemyung Ahn, and Han Lim Choi. PSO-based Optimal Task Allocation for Cooperative Timing Missions. 2016.

[4] Yohanes Khosiawan and Izabela Nielsen. Indoor UAV scheduling with Restful Task Assignment Algorithm. 2017.

[5] Jia-lei Liu, Zhi-guang Shi, and Yan Zhang. A New Method of UAVs Multi-target Task Assignment. 2018.

[6] 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.

[7] Nuri Ozalp, Ugur Ayan, and Erhan Oztop. Cooperative multi-task assignment for heterogonous UAVs. 2015.

[8] Phillip R. Chandler, Meir Pachter, Steven R. Rasmussen, and Corey Schumacher. Multiple task assignment for a UAV team. 2002.

[9] Andrew K. Whitten, Han Lim Choi, Luke B. Johnson, and Jonathan P. How. Decentralized task allocation with coupled constraints in complex missions. 2011.

[10] Wenyi Chen, Martijn Mes, and Marco Schutten. Multi-hop driver-parcel matching problem with time windows. 2018.

[11] Ocident Bongomin and Aregawi Yemane. The hype and disruptive technologies of industry 4.0 in major industrial sectors: A state of the art. 2020.

[12] 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.

[13] 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.

[14] 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.

[15] Jonathan Gross, Jay Yellen, and Ping Zhang. *Handbook of Graph Theory*. CRC Press, 2014.