Cooperative route planning of multiple fuel-constrained Unmanned Aerial Vehicles with recharging on an Unmanned Ground Vehicle

Subramanian Ramasamy[1†], Jean-Paul F. Reddinger[2], James M. Dotterweich[2], Marshal A. Childers[2], Pranav A. Bhounsule [1†]

***Abstract*— Multiple small, low cost, multi-rotor Unmanned Aerial Vehicles (UAVs) are ideal for aerial surveillance over large areas. However, their limited battery capacity restricts them to areas in proximity of stationary recharging depots. One solution is to use an Unmanned Ground Vehicle (UGV) to provide a moving recharging depot. The problem is then to find the time- or energy-optimal paths for the multiple fuel-constrained UAVs to visit a set of mission points while being recharged by stopping at the UGV, whose path also needs to be determined. This is a combinatorial optimization problem that is computationally challenging, but may be solved relatively fast using heuristics. In this paper, we present two-level optimization that involves, (1) finding a UGV path by fixing waypoints using K-means and then formulating and solving a traveling salesman problem (TSP), and (2) finding paths for the multiple UAVs using a vehicle routing problem (VRP) formulation with capacity constraints, time windows, and dropped visits. We used constraint programming to solve these problems in less than a minute on a standard desktop computer for up to 25 mission points and 4 UAVs. Our main observation is that increasing the number of UAVs decreases the mission time and refueling stops, but does not decrease the total distance covered or total time taken.**

Keywords: Traveling Salesman Problem, Vehicle Routing Problem, K-means clustering, Fuel Constraints, Constraint Programming, Local Search Heuristics.

1. INTRODUCTION

Recent advances in the design and control of multi-rotor aerial vehicles such as quadcopters or drones have made them practical for applications such as surveillance, reconnaissance, environment and traffic monitoring, search and rescue, and border patrol [5]. The relatively low cost, simple hardware, high speed, and ease of control enable several Unmanned Aerial Vehicles (UAVs) to be deployed for large-scale coverage [1]. However, their limited battery capacity severely restricts their capability to relatively small areas [15].

One way of enabling large-scale coverage with limited battery capacity is to have multiple fixed recharging depots. For example, in case of monitoring the aftermath of a disaster scene, the UAVs may visit the ground base stations for recharging to extend the monitoring time [10]. However, there are several situations such as in non-urban environments and/or hostile environments where it may not be possible to fix the recharging depots [3]. In such situations, a viable alternative is to have an Unmanned Ground Vehicle (UGV) providing recharging support by coordinating with the UAVs. Figure 1 shows an example scenario where two fuel-limited UAVs can visit a set of mission points recharging at strategic positions on the UGV path. Here, we are given only the mission points and the start position of the UAVs-UGV, while we need to determiend the route of the UGV, route of the two UAVs and the recharging stops while minimizing an objective (e.g., minimize the distance travelled by the UAVs). This is a combinatorial optimization problem that is known to be non-deterministic polynomial time or NP-hard. Hence we need to design suitable heuristics to solve the problem in finite time. This paper presents a two-level optimization framework based on formulating a traveling salesman problem and vehicle routing problem and then solving them using constraint programming.

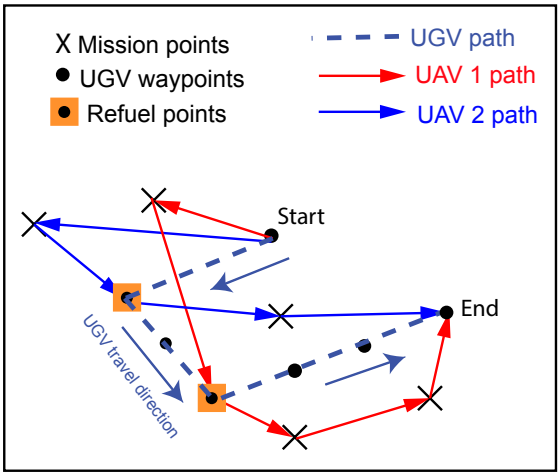

Fig. 1. An example scenario. The objective is to route the UAVs to visit the missions (crosses). Since the UAVs are fuel constrained they cannot visit all missions on a single charge. By planning a suitable route for the UGV (blue dashed lines) with possible recharging locations (small black circles), the 2 UAVs can plan to take a stop each (large orange circles) and visit all missions without running out of fuel.

2. BACKGROUND AND RELATED WORK

We limit the literature review to vehicle routing problems with fuel constraints.

Khuller et al. [7] were the earliest ones to solve vehicle routing problem with fuel constraints. They considered the

[1] Subramanian Ramasamy and Pranav A. Bhounsule are with the Department of Mechanical and Industrial Engineering, University of Illinois Chicago, IL, 60607 USA sramas21@uic.edu pranav@uic.edu [2] Jean-Paul F. Reddinger, James M. Dotterweich, Marshal A. Childers are with DEVCOM Army Research Laboratory, Aberdeen Proving Grounds, Aberdeen, MD 21005 USA. jean-paul.f.reddinger.civ@mail.mil james.m.dotterweich.civ@mail.mil marshal.a.childers.civ@mail.mil. † This work was supported by ARO grant W911NF-14-S-003

problem of finding the cheapest route for a fuel constrained vehicle with a set of fueling stations, each with a different fuel price. They used a dynamic programming (DP) formulation to solve the problem. Kannon et al. [6] considered the problem of finding the route of a fuel-constrained aircraft to visit a set of waypoints with set of aerial refueling waypoints. They compared a mixed-integer linear programming (MILP) formulation with a DP formulation and found that DP outperforms MILP.

Levy et al. [8] and Sundar et al. [13] considered extensions to multiple fuel-constrained Unmanned Aerial Vehicles (UAVs). The goal here was to minimize the distance travelled by multiple fuel-constrained UAVs to visit a set of waypoints once and recharge on ground-based recharging depots. Levy et al. used a variable neighborhood search based on randomization and variable neighborhood descent based on the gradient to search for an optimal solution. Sundar et al. formulated several mixed-integer linear programming (MILP) formulations and solved these using an off-the-shelf MILP solvers.

Maini et al. [9] considered the problem of routing a single fuel-constrained UAV to a set of missions while being recharged by stopping at a UGV traveling on a road network. They solved the problem using a two-stage approach. First, using the UAV range constraints, they found a set of recharging depots. Second, they formulated a mixed-integer linear program and solved for the path of both the UAV and UGV. We consider an extension of the problem considered by Maini et al. These extensions are: we consider multiple fuel constrained UAVs, we use an off-road UGV, both of which add complexity to the problem as we need to plan the path of the UGV, the recharging points for the UAVs, and the paths for the UAVs.

We present a two-level optimization to solve for the UAV-UGV path. In the first level, we solve for the UGV path by fixing waypoints using K-means and then formulating and solving a traveling salesman problem (TSP). In the second level, we solve for the paths of multiple UAVs by using a vehicle routing problem (VRP) formulation with capacity constraints, time windows, and dropped visits. We solve the optimizations at both levels using constraint programming using Google's OR-Tools™. The novelty compared to previous work is in the deployment of the UGV in off-road environments, extension to multiple UAVs, and solution using constraint programming. Though we used an off-the-shelf solver (Google's OR tools), our approach has the following novelties.

- The framework allows planning of the UGV path in off-road environments using TSP and subsequently the UAV path planning using VRP considers UGV speed through time windows, fuel limits using capacity constraints, and to choose recharge location using dropped visits.
- The use of constraint programming with heuristics leads to a search-based instead of an optimization-based solution approach that enables solution times under a minute for up to 25 missions and 4 UAVs and 1 UGV.

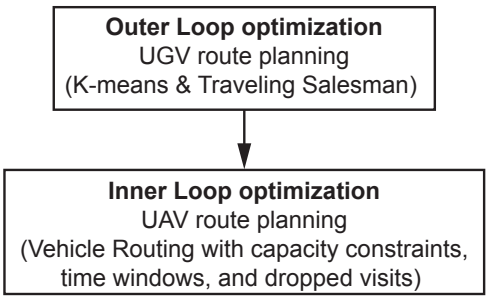

Fig. 2. Two-level optimization for solving UAV-UGV routing problem

3. METHODS

A. Problem statement

The overall objective is to plan the path of K fuel-limited Unmanned Air Vehicle (UAVs) to pre-specified mission points while minimizing the total distance travelled. The UAVs are recharged by docking on a single Unmanned Ground Vehicle (UGV). The path of the UGV and its stopover locations are not pre-specified, but needs to be computed. The velocity of the UAV is fixed, while the velocity of the UGV can be 0 to a maximum value. It is assumed that the UGV has no fuel constraints and can provide unlimited recharges to the UAVs. When the UAV docks on the UGV, it is assumed that the UGV does not move. Thus, the UAV take-off point is the same as the landing point. The total travelled between recharges is fixed at 15 minutes and there is a fixed time, called service time $s = 10$ min. that the UAV spends at the UGV during recharging and also at the mission point.

B. Solution approach

We solve the problem in a hierarchical fashion using a two-level optimization as shown in Fig. 2. At the first-level, we optimize the UGV route. First, we use k-means clustering to find clusters and their centroids. Using these centroids as waypoints, we formulate and solve a traveling salesman problem to get the route for the UGV. We specify the centroids and additional heuristically added points between the centroids as potential stops for the UAV to recharge. At the second-level, we optimize the UAV routes using the mission points and UGV stop locations from the first-level of the optimization. We formulate a vehicle routing problem by adding fuel capacity constraints for the UAVs, time windows for the UGV stops, and allowing the UAVs to drop visits at the fuel stops. We now give more details.

C. K-means clustering

We use K-means clustering to find suitable waypoints for the UGV [18]. K-means clustering is a technique to group n observations into k clusters. Each of these k clusters have a central location, which is the centroid of the cluster. Our goal is to find the k clusters and the centroid. This problem is NP hard so we resort to the heuristic algorithm shown as Algorithm 1 and described next.

Algorithm 1 K-MEANS ALGORITHM

Input: k, $[\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n}]$;

Randomly place k centroid points $\boldsymbol{c_1}, \boldsymbol{c_2}, ..., \boldsymbol{c_k}$
while not StoppingCondition() **do**
  for each observation $\boldsymbol{x_i}$ **do**
    evaluate nearest centroid point $\boldsymbol{c_j}$ by evaluating Euclidean distance **min D** $\boldsymbol{(x_i, c_j)}$;
    Assign observation $\boldsymbol{x_i}$ to the nearest centroid $\boldsymbol{j}$;
  end for
  for each do cluster $\boldsymbol{j = 1, 2, ..., k}$
    recompute $\boldsymbol{c_j = \frac{1}{n_j} \sum_{x_i \to c_j} x_i}$;
  end for
end while

The inputs to the algorithm are the 'n' mission locations $x_1, x_2, ...x_n$ and the number of clusters, k. Initially, we assign the centroid points at random. These centroid points are assigned by picking some 'k' points randomly from location of the existing mission points in the space. Next, we carry a sequential optimization; first, to assign membership for an observation to a cluster, and second, to recompute the centroid of the cluster. To assign membership, we find the distance between an observation and centroid of each cluster and then assign the observation to the cluster with minimum distance. To recompute the centroid of the cluster, we use the observations from the cluster. These two steps are repeated till a *StoppingCondition()* that no observation changes cluster membership.

D. UGV route using traveling salesman problem formulation

Since there is a single UGV with no fuel constraints, we solve for the UGV route using Traveling Salesman Problem formulation.

Consider a directed graph $G' = (V', E')$ where V' is the entire set of vertices $V' = \{0, 1, 2, ..., k\}$, which are the cluster centroids with 0 and k as the 'start' and 'end' vertices, and E' is the set of edges that gives the arc costs between i and j and $E' = \{(i, j)|i, j \in V', i \neq j\}$. The c'_{ij} gives the non-negative arc cost between a particular i and j. The x'_{ij} is a binary variable where the value of x'_{ij} will be 1 if a vehicle travels from i to j, and 0 otherwise. We formulate the TSP problem follows,

$$min \sum_{i \in V'} \sum_{j \in V'} c'_{ij} x'_{ij} \tag{3.1}$$

$$s.t., \sum_{i \in V'} x'_{ij} = 1, \qquad \forall j \in V' \setminus \{0, k\} \tag{3.2}$$

$$\sum_{j \in V'} x'_{ij} = 1, \qquad \forall i \in V' \setminus \{0, k\} \tag{3.3}$$

$$\sum_{j \in V'} x'_{0j} = \sum_{i \in V'} x'_{ik} = 1, \quad \{0, k\} \in V' \tag{3.4}$$

$$\sum_{i \in Q} \sum_{j \in Q} x'_{ij} \leq |Q| - 1, \quad \forall Q \subsetneq \{1, ..., k\}, |Q| \geq 2 \tag{3.5}$$

$$x'_{ij} \in \{0, 1\}, \quad \forall i, j \in V' \tag{3.6}$$

The objective (Eqn. 3.1) is to minimize the total distance of the traveled by the UGV. The constraints (Eq. 3.2) and (Eq. 3.3) denote that each vertex is visited once, that is, balancing the incoming and outgoing number of vehicles at a particular vertex. Constraint (Eq. 3.4) ensures that the vehicle must start from 'start' vertex and ends at the 'end' vertex. Although constraint 3.4 is satisfied by constraints 3.2 and 3.3, we present them separately for the sake of completeness. Constraint (Eq. 3.5) ensures that there are no sub-tours. Constraint (Eq. 3.6) represents the binary type decision variable used.

We solve the above problem using Constraint Programming [11], [12], more specifically, using Google's OR-Tools™[4].

E. UAV route using vehicle routing formulation

Table I describes the sets used to describe the UAV vehicle routing problem. Consider a directed graph $G = (V, E)$ where V is the entire set of vertices $V = \{0, 1, 2, ..., m, m+1, ..., n\}$ and E is the set of edges that gives the arc costs between i and j and $E = \{(i, j)|i, j \in V, i \neq j\}$. Let c_{ij} be the non-negative arc cost between a particular i and j. Let x_{ij} be the binary variable where the value of x_{ij} will be 1 if a vehicle travels from i to j, and 0 otherwise. We formulate the VRP problem with capacity constraints, time windows, and dropped visits as follows,

$$min \sum_{k \in K} \sum_{i \in V} \sum_{j \in V} c_{ijk} x_{ijk} + \sum_{d \in D \setminus \{0, m\}} \alpha p_d, \quad D \subset V \tag{3.7}$$

$$s.t., \sum_{k \in K} \sum_{i \in V} x_{ijk} = 1, \quad \forall j \in V \setminus \{0, m\} \tag{3.8}$$

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1, \quad \forall i \in V \setminus \{0, m\} \tag{3.9}$$

$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{im} = K, \{0, m\} \in D \tag{3.10}$$

$$C(r_k) \leq C_k, \quad \forall k \in K \tag{3.11}$$

$$t_j^k = t_i^k + s_i^k + t_{ij}^k, \quad \text{where} \quad \forall i \in V, j \in V \setminus \{0\}, k \in K \tag{3.12}$$

$$t_{start} \leq t_j^k \leq t_{end}, \quad \text{where} \quad \forall j \in V \setminus \{0\}, k \in K \tag{3.13}$$

$$f_j^k = f_i^k - c_{ij}^k, \quad \forall k \in K, i, j \in M \tag{3.14}$$

$$f_j^k = N^k, \quad \forall k \in K, j \in D \setminus \{0, m\} \tag{3.15}$$

$$x_{ij}^k \in \{0, 1\}, \quad \forall i, j \in V, \forall k \in K \tag{3.16}$$

$$p_d \in \{0, 1\}, \tag{3.17}$$

$$p_d + ActiveVar(d) = 1-, \quad \forall d \in D \setminus \{0, m\} \tag{3.18}$$

$$s_i^k \geq 0 \tag{3.19}$$

$$\alpha \geq 0 \tag{3.20}$$

$$C_k > 0, \quad C_k \in \mathbb{R}_+ \tag{3.21}$$

$$T^k > 0, \quad T^k \in \mathbb{R}_+ \tag{3.22}$$

The objective, Eqn. 3.7, is to minimize the sum of the total travel distance of all the UAVs and the penalty costs for dropping the visits. The concept of dropping the visits will be explained in detail below.

The constraints in Eq. 3.8 and Eq. 3.9 denote that an UAV visit each vertex only once, with exception of the starting depot 0 and ending depot m. Thus, the first two constraints are made in such a way that the two UAVs do not coincide simultaneously in the same charging station (UGV stops) as well. Constraint in Eqn. 3.10 denotes that the number of UAVs leaving the depot 0 is equal to the number of UAVs reaching the depot m. Constraint in Eq. 3.11 denotes that the total cost of a particular UAV's route r_k ($C(r_k)$) is always less than or equal to the corresponding UAV's maximum allowable distance C_k that it can travel. Constraint in Eq. 3.12 denotes that the cumulative arrival time at j, t_j^k, is equal to the sum of cumulative time at the mission i, the service time at the mission i, s_i^k, and the travel time from i to j, t_{ij}^k. Constraint in Eqn. 3.13 ensures that the UAV's cumulative time at a particular vertex (either missions from M or depots from D) is always within the range of the time window, $[t_{start}, t_{end}]$. This $[t_{start}, t_{end}]$ represents the time window at which UGV arrives and stays at a particular UGV stop. This time window values are obtained from the ratio of the distance between any two UGV stops and the velocity with which UGV travels. Constraints Eq. 3.14 and Eq. 3.15 represents the fuel constraints. Constraint 3.14 denotes that when any UAV 'k' is visiting a set of missions, the fuel of the UAV decreases proportional to the distance traveled from node i to node j. Once the UAV comes to the recharge at a UGV stop, i.e., a node which belongs to set $D \setminus \{0, m\}$, it recharges to N^k, where N^k is the full fuel capacity for a UAV k. This is represented by the constraint Eq. 3.15. It then completes rest of the missions till it reaches the end depot m, $m \in D$. In short, f^k is equal to N^k during the start of the problem. Constraints given by Eq. 3.16 to Eq. 3.22 represents constraints on the free variables. Here p_d, Eq. 3.17, is a binary decision variable corresponding to the $Indices$ set, which is mainly operated by the constraint 3.18. This binary variable p_d is responsible for dropping UGV stops. In constraint 3.18, ActiveVar(d) means that if the node 'd' is visited by a vehicle, then ActiveVar(d) is equal to 1, else ActiveVar(d) is 0. Intutively, set $D \setminus \{0, m\}$ in 3.18 can be interpreted as the nodes in the D set which are to be optionally dropped. Each optional node in D can only be visited a maximum of a single time according to the constraint 3.18. That is, if a node d is visited, then p_d must be 0 to satisfy 3.18, thereby penalty is 0 as the node is visited. If a node d is not visited, then p_d must be equal to 1 to satisfy 3.18 which leads to adding penalty α in the objective function. The penalty α, Eq. 3.20, is a user chosen real number which decreases the dropped visits as we increase its value and vice versa. This α value can be tuned in such a way that if a smaller number is chosen, then the solver tries to drop as many nodes as possible and if a large number of penalty is chosen, the solver tries to avoid dropping as many nodes as possible because the solver has to pay such a huge penalty if a node is to be dropped. he C_k in Eq. 3.21 is the user chosen positive real number which defines the allowable maximum distance that a particular UAV can travel. The total distance of the UAV in operation must be lesser than or equal to C_k. Similarly, the T^k in Eq. 3.22 is the user chosen positive real number which defines the allowable maximum duration that a particular UAV can fly. The total duration of the UAV in operation must be lesser than or equal to T^k.

Sets	Description	Set elements
V	Set of Vertices	$V = \{0, 1, 2, ..., m, m+1, ..., n\}$
E	Set of Edges	$E = \{(i, j) \mid i, j \in V, i \neq j\}$
D	Set of UGV stops	$D = \{0, 1, 2, ..., m\}$
M	Set of missions	$M = \{m+1, ..., n\}$
K	Number of UAVs	$K = \{1, 2, ..., k\}$
F	Full fuel i.e., Fuel capacity of UAV	$F = \{f_1, f_2, ..., f_k\}$

TABLE I
SETS USED IN THE UAV ROUTING PROBLEM

The concept of dropping UGV stops is vital in our problem as dropping as many refueling nodes as possible will save the total distance traveled by the UAVs thereby avoiding unnecessary refueling stops made by the UAVs on the UGV.

We solve the above problem using Constraint Programming [11], [12] using Google's OR-Tools [4]. In this formulation, sets D and M are proper subsets of V. Set D represents the number of UGV stops including starting depot and ending depot, and set M represents the number of missions that UAVs need to visit once.

F. Solution using Constraint Programming (CP)

We solved the UGV routing (TSP) (see Section 3-D) and UAV routing (VRP) (see Section 3-E) problems using Constraint Programming (CP). Constraint programming or constraint optimization is a tool for solving hard combinatorial optimization problems by searching for solutions that satisfy a list of constraints.

In particular, we used Google's OR-Tools solver. OR-Tools uses a search tree, local search, and meta-heuristics to find feasible and subsequently, the most optimal solutions. At the heart of OR-tools is a CP-SAT solver [4]. The solver uses *DecisionBuilder* that has as its input, the decision variables, rules to choose the next variable to assign a value, rules for choosing the value to assign to the variable. Using the *DecisionBuilder*, we use the *Path Cheapest Arc* strategy to find an initial feasible solution (see algorithm in [14]). Starting with the "start" node, the decision builder connects the node that has the shortest distance from the previous node and iterating till the end. While doing the connections, it checks the feasibility of the solution.

Then OR-Tools uses a local search to find the best solution in the neighborhood of the current solution. These local search proceeds by a move operator that rewires the nodes

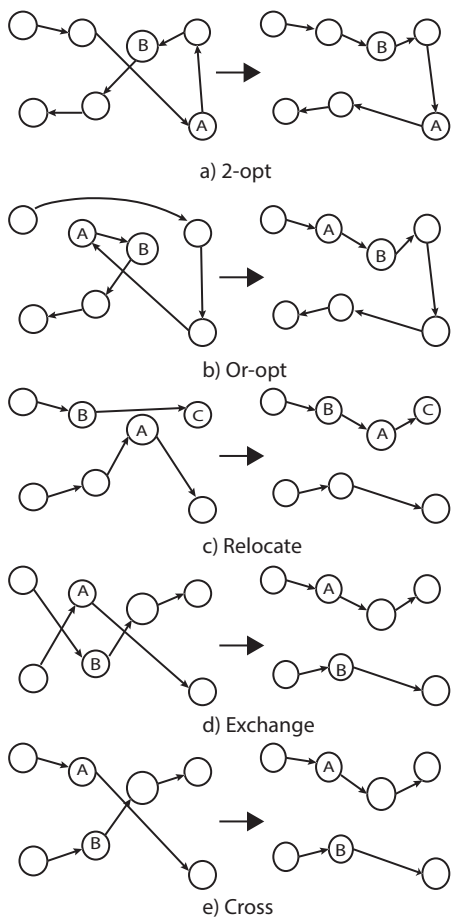

Fig. 3. Move operators [2]

and checks for feasibility and cost. These moves are repeated till a termination criteria such as no improvement of the objective. There are 5 move operators. These are listed next and shown in Fig. 3 and is taken from [2].

1) **2-opt** interchanges the sub-part of a tour by removing two arcs, and then connects them interchangeably so that the objective value gets reduced.
2) **Or-opt** moves the sub-part of a tour if there are a maximum of 3 contiguous visits to that sub-part of the tour.
3) **Relocate** connects a visit of one tour to another tour if the reduction in objective value is seen.
4) **Exchange** involves swapping two visits between each other from either the same tour or two different tours.
5) **Cross** involves exchange of a visit at the end of one to another tour. The difference between Exchange and Cross is that the Exchange move can be done in any part of tour/tours, but Cross can be done only to the end portions of two tours.

In order to escape a local optimum solution, OR-Tools use meta-heuristics. We use the Guided Local Search (GLS) in our problem [16]. In GLS, we add a penalty term to the objective function O leading to an augmented objective O'

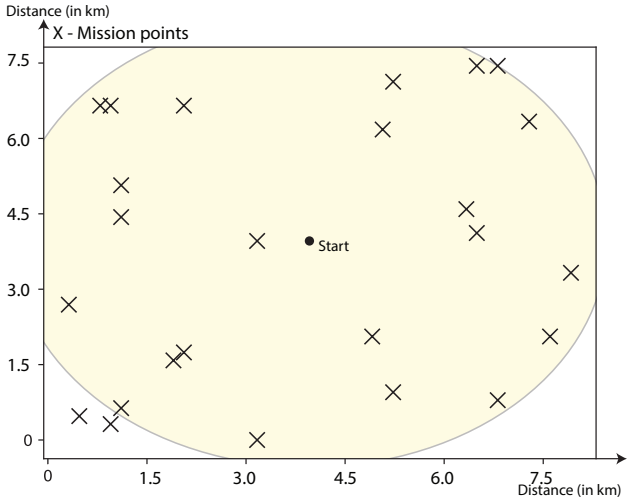

Fig. 4. Graphical representation of the 25 mission points on a coverage area of 8×8 sq. km. Both, the UGV and UAV are assumed to start at the center of the coverage area shown as 'start'. The circle represents the UAV round trip coverage on a single battery capacity.

function. The penalty term is dependent on the neighborhood of the solution x through a set of features F. The augmented objective function is [2]

$$O'(x) = O(x) + \lambda \sum_{i \in F} f_i(x) p_i c_i \tag{3.23}$$

where the indicator function for the corresponding feature i that belongs to F is f_i. We define $f_i(x) = 1$, if the feature i is in solution or 0 otherwise. Also, λ is the penalty factor that can tune the search for the solutions. For example, a larger λ increases the diversity of the solutions (also see [17]), p_i is the number of times the particular feature i has been penalized, and c_i is the cost for the feature f_i. Using the augmented objective O' increases the cost of the objective with respect to the neighborhood, thus enabling the solver to get unstuck from a local optimum solution. Subsequently, a local search is used to continue the search.

4. RESULTS

We generated 25 mission points randomly in an area of 8×8 kms as shown in Fig. 4. We solve using single UGV, but vary the number of UAVs, $1, 2, 3, 4$ to compare the solutions as we add UAVs. The UAV and UGV both start at the center of the area. Since the UAV travels at a constant speed of 10 m/s, the distance is proportional to time and is interchangeable after suitable conversion. The fuel capacity in terms of time is 15 minutes, which is equivalent to a distance coverage of 9 km. We fix the service time at mission and at the UGV to be 10 minutes. We assume that there is no fuel consumption when the UAV is at the UGV and that the UGV does not move when the UAV has landed on it. The minimum and maximum speed of the UGV is used to set the time windows in the UAV route optimization.

First of all, note that if the UGV is stationary at the middle, then there are no solutions as there are some missions that are far enough that the UAV does not have enough charge

TABLE II

RESULTS FOR MULTIPLE UAVS FOR 2 CLUSTERS

# UAV	UAV 1			UAV 2			UAV 3			UAV 4			Total Distance (kms.)	Total Time (min.)	Mission time (min.)
	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission			
1	40.34	4	25										40.34	357.23	357.23
2	38.99	4	19	8.69	0	6							47.68	369.47	294.98
3	24.76	3	13	17.76	1	7	8.28	0	5				50.8	374.67	201.27
4	14.99	1	5	15.13	1	7	8.85	0	6	16.06	1	7	55.03	371.72	107.77

TABLE III

RESULTS FOR MULTIPLE UAVS FOR 3 CLUSTERS

# UAV	UAV 1			UAV 2			UAV 3			UAV 4			Total Distance (kms.)	Total Time (min.)	Mission time (min.)
	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission			
1	42.73	6	25										42.73	381.22	381.22
2	35.74	4	16	17.6	1	9							53.34	388.9	259.58
3	17.65	1	8	27.44	3	12	8.88	0	5				53.97	379.95	195.73
4	8.01	0	3	20.81	2	9	7.56	1	6	21.2	1	7	57.58	385.97	144.68

TABLE IV

RESULTS FOR MULTIPLE UAVS FOR 4 CLUSTERS

# UAV	UAV 1			UAV 2			UAV 3			UAV 4			Total Distance (kms.)	Total Time (min.)	Mission time (min.)
	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission	Distance (kms)	#refuel	#mission			
1	49.13	6	25										49.13	391.88	391.88
2	8.89	0	6	40.75	5	19							49.64	382.73	307.92
3	9.03	0	3	18.07	1	6	35.23	4	16				62.33	403.88	258.72
4	9.03	0	3	8.27	0	2	6.26	0	4	35.23	4	16	58.79	387.98	258.72

take a round trip. Figure 4 yellow region shows the range of the UAV . Thus, it is necessary for the UGV to move in order to have feasible solutions.

We used constraint programming solver in Google's OR-Tools™[4]. We coded the optimization problem in Python 3.9.0 and performed the computations on an 2.2 GHz Quad-Core Intel Core i7 processor with 16 GB RAM 1600 MHz DDR3 memory. All optimizations took less than 60 seconds to complete.

Figure 5 shows the clusters for cluster size $k = 4$ using the K-means algorithm. The large circles show the centroid of the cluster, while we show the mission points with smaller circles. We show mission points that belong to the same cluster using the same color.

Figure 6 shows the results of solving for the UGV route. We solve the problem using the centroid locations found using k-means as the waypoints and with the Euclidean distance as the objective in the traveling salesman problem. The solution gives the UGV route, including the direction of motion of the UGV. The 'end' location is the final node and depends on the number of clusters chosen. Finally, we add 3 more waypoints between two UGV waypoints as possible stopping locations for the UAV for recharging. Although we show 3 waypoints here, we found that adding additional waypoints did not change the optimum appreciably.

Figure 7 shows the UAV routes for a different number of UAVs, $K = 1, 2, 3, 4$. The 'start' and 'end' denotes the start and end position of both the UGV and all UAVs. We show the UGV route for using a blue dashed line with the direction as shown. We depict the UAV route by a solid line of a different color. The arrows on the dashed and solid line indicate the direction of motion of the UGV and UAV, respectively. We show the recharging spots with an orange dot. For example, in Figure 7 (a) the UAV has 6 orange dots indicating that the UAV stops at the UGV 6 times. We can see that there is no instance when over one UAV is recharging at a particular UGV stop as there multiple UAVs do not cross at the orange dot. Also, when the number of UAVs is greater

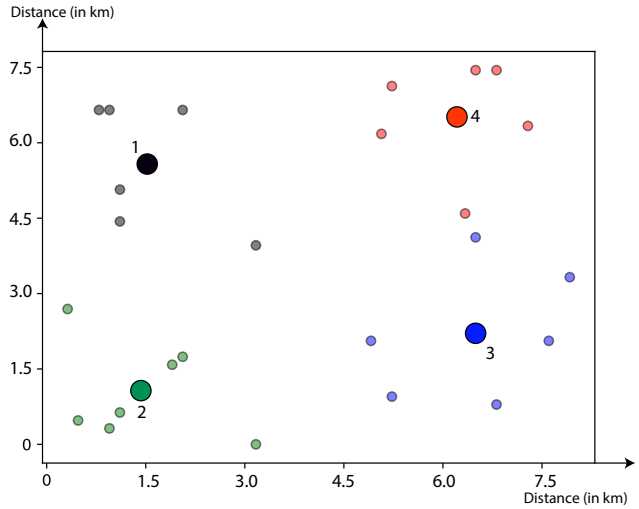

Fig. 5. Results for cluster size of 4. Large circle indicates the centroid of the cluster and small circles represent the mission points. Missions that belong to a cluster are in the same color.

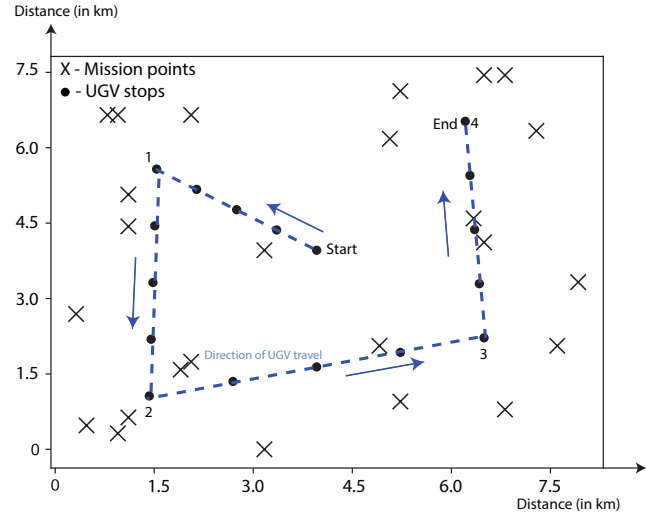

Fig. 6. UGV route obtained by solving the traveling salesman problem with distance as the objective using the 4 centroids as waypoints. The centroids are numbered. Additional stops are shown with small filled circles as potential locations for the UGV to stop.

than 1, we can see that one UAV travels substantially more distance compared to the others. We elaborate on this in the discussion section.

Figure 8 shows the the solution for different cluster sizes $k = 2$ and $k = 3$ for 2 UAVs. Also, note that Fig. 7 (b) shows the solution for $k = 4$ for 2 UAVs. The use of different cluster sizes leads to substantially different UGV routes. For 2 clusters (Fig 8 (a)), the UGV starts towards the right corner and then takes a turn to move towards the left corner. For 3 clusters (Fig 8 (b)), the UGV starts toward the bottom left corner before turning top left and then going horizontally. For cluster size of 4 shown in Fig. 7 (b) respectively, the UGV starts toward the top left before going vertically down then horizontally to the right and eventually going vertically up to the end. These UGV routes lead to widely different UAV solutions with different missions point served at different times.

Tables II, III, and IV are for cluster sizes of 2, 3, and 4 respectively. They provide a comparison of the distance travelled, number of recharging stops, mission travelled for each UAV and total distance and total travel time. For each cluster, we solved for 1, 2, 3, and 4 UAVs. Our overall objective was to minimize the total distance travelled (third to last column). It can be seen that for a specific cluster size the distances are the least for 1 or 2 UAVs. Within the clusters, the least distance is 40.34 for cluster size 2 and 1 UAV (Table II). The total time is given by the travel time from one mission to another (= distance in km/UAV speed in km/min) and the service time. The service time s is given by

$$s = 10 \times \#\text{ refuel} + 10 \times \#\text{ mission}$$

where 10 is the number of minutes taken by the UAV at the recharging stop and mission. There is no clear trend in the minimum time across these clusters. For cluster size of 2, 3, and 4, the minimum time is for 1, 3 and 2 UAVs respectively. The total stops are obtained by summing the individual stops in each row. For example, for Table III there are 6, 5, 4, 4 for 1, 2, 3, and 4 UAVs respectively. Thus, as the number of UAVs increases the recharging stops decrease. The same is true for other clusters. In the individual table it can be seen that there is one UAV that travels the most distance compared to the others. For example, in Table IV, for 4 UAVs, the UAV 4 travels 35.23 km, refuels 4 times and serves 16 missions while the others serve 6.26, 8.27, and 9.03 km, do not refuel, and serve 4, 2, and 3 missions respectively. This happens because our objective of reducing the total distance covered does not specifically try to balance out the distances travelled by UAVs. Moreover, as seen in Fig. 7 (d), UAV 4 serves bottom left set of missions while the other serve the mission on the top right side indicating that the specific spread of the missions contributes to the imbalanced distance coverage.

The mission time which is the time from start to end is given in the last column of Tables II, III, and IV. It can be seen that as the number of UAVs is increased the mission time decreases. For example, for 2 clusters, the mission time is 357.23 min for 1 UAV, but it reduces to 107.77 min for 4 UAVs, a factor of 3 reduction. One can also see that the number of refueling decreases from 4 for 1 UAV to only 1 for 4 UAVs. The reduction in mission time and refueling is an advantage of using more UAVs.

5. DISCUSSION

In this paper, we have presented a methodology to solve for routing of a multiple fuel-limited Unmanned Aerial Vehicles (UAVs) and an Unmanned Ground Vehicle (UGV) such that the UAVs visit a set of missions and recharge as needed by landing on the UGV. We solved the problem using a two-level optimization; first, we solve for the UGV route; and second, we use the UGV route to solve for the UAV route. To compute a route for the UGV, we used K-means clustering to decide a set of waypoint and then formulated

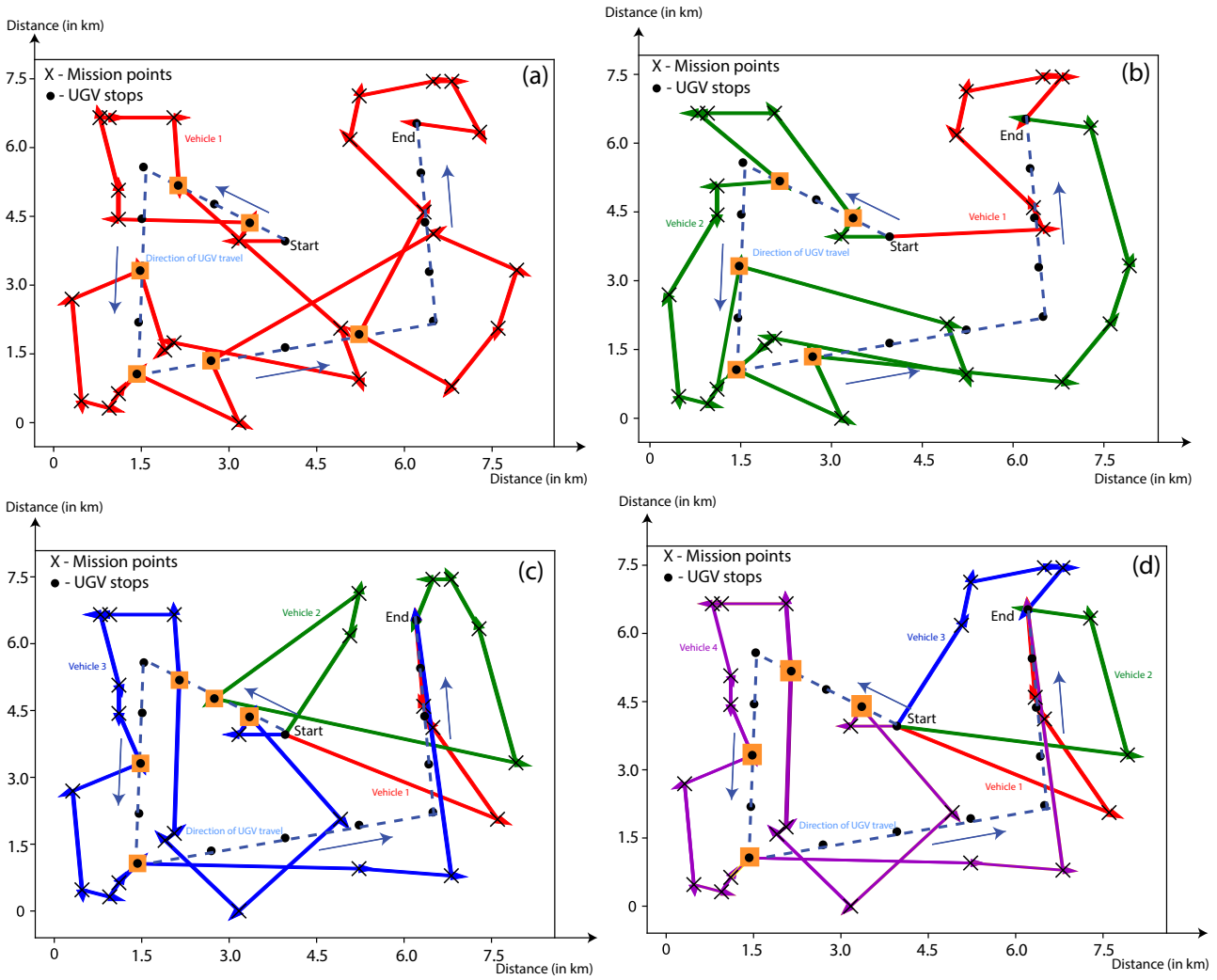

Fig. 7. Routes obtained for different number, k, of UAVs, (a) k = 1, (b) k = 2, (c) k = 3, and (d) k = 4

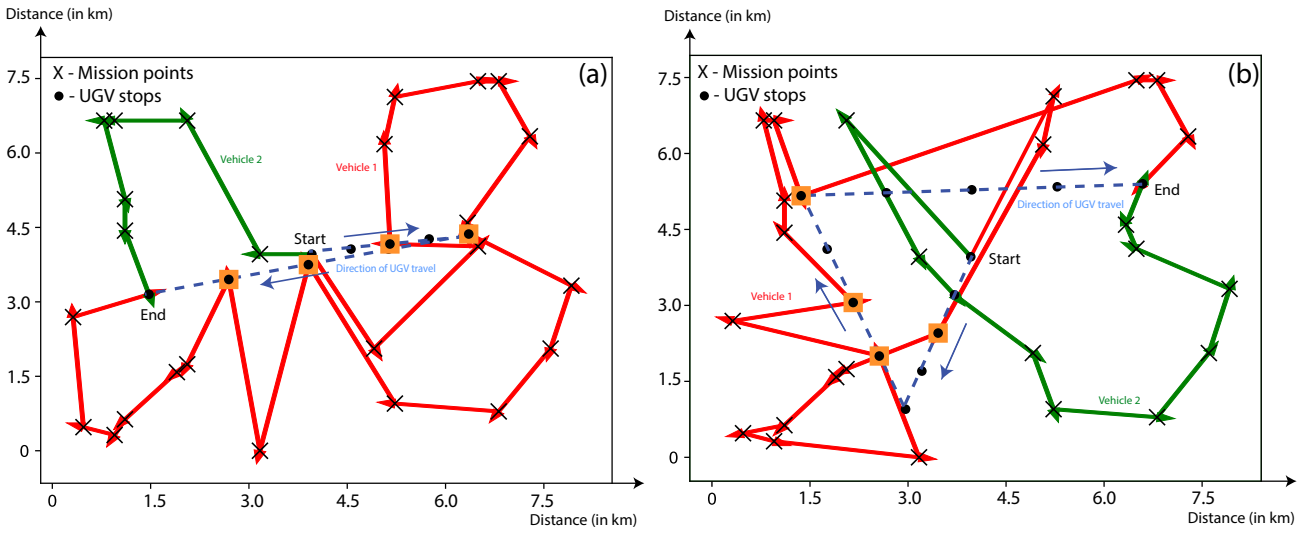

Fig. 8. Routes obtained for 2 UAVs with different cluster size for UGV route selection (a) 2 clusters, (b) 3 clusters. Also, compare with 4 clusters in Fig. 7 (b).

and solved a Traveling Salesman Problem. To compute the route of the UAVs, we formulated and solved a Vehicle Routing Problem with capacity constraints (fuel limits), time windows (to match UGV meeting points for recharging), and dropped visits (allow UAV to stop at some, bit not all the UGV waypoints).

The cluster points define points to which the UGV must travel to extend the range of the fuel-limited UAV. The minimum number of clusters to get a feasible solution is 1. This corresponds to moving a short distance toward $0, 0$ as shown in Fig. 4 such that all mission points are eventually covered by the UAV. As the number of clusters increases, it enables the UGV to have a larger coverage area which in turn can reduce the total travel distance of the UAV. However, larger coverage does not imply better solutions for the UAV hence we need to find the optimum number of clusters. In this scenario, we found that for 4 clusters we get the optimum UAV solution.

Since we first solved the UGV route followed by the UAV route, the quality of the UAV solution is highly depended on the UGV route solution. In particular, the choice of the number of clusters, which determine the number of waypoints for the UGV route selection, results in substantially different UGV route (compare the dashed lines in Fig. 8 (a) with Fig. 7 (b), and Fig. 8 (b)) and subsequently the UAV solution. For a specific cluster size, we found that using 1 or 2 UAVs results in the least total distance travelled compared to using 3 or 4 UAVs. On close examination of the solutions, we found distances travelled by the UAVs are not similar. Usually, one UAV travels substantially more distance compared to the others. There are probably two reasons for this: one, the geographic distribution of the missions favors this solution, and two, we did not enforce that all UAVs finish their missions in the same time, thus allowing some UAVs to travel shorter distance.

Although we obtained minimum travel distance using a single UAV, there are benefits of using multiple UAVs. With more UAVs, the time from start to end or mission time, decreases. Also, as we add more UAVs, the number of recharging stops decreased.

Our methodology has limitations, which we list next. In the two-level optimization, the results from the first level significantly affect the results of the second level. Thus, it is important to vary the different free parameters in the first level to understand their influence on the second level of optimization. The k-means clustering assigns clusters based on distance between mission points, but not the number of missions. There could be a scenario where the distance and the number of missions within a cluster needs to be factored in to arrive at a feasible solution (e.g., highly packed clusters). In our formulation, we use the average speed of the UGV of 1.1 mph to fix the time windows for UAV route optimization. However, our resulting solution does not lead to a uniform speed of the UGV from one recharging depot to the next. That is, the UGV might have to travel faster or slower than average speed between some recharging depots. This is potentially problematic if the UGV cannot travel faster because of terrain conditions. Our UGV-UAV solution is computed offline and executed in an open-loop fashion. Thus, if the missions and/or UGV paths change due to dynamically changing scenarios, we do not make modifications to the plan. However, given the short computation times of < 1 min, it is conceivable that we can recompute the solutions either on a ground station or on the cloud and relay the updated route to the UGV and UAV.

6. CONCLUSION AND FUTURE WORK

We conclude that the problem of routing multiple fuel-constrained Unmanned Aerial Vehicle (UAV) with recharging on a single Unmanned Ground Vehicle (UGV) can be formulated as a two-level optimization and solved effectively using constraint programming. In the first level of optimization, the UGV route is generated by using k-means to generate waypoints and then formulated and solved as a traveling salesman problem. In the second level of optimization, the UAV route is generated by using a vehicle routing formulation with capacity constraints, timed-windows, and dropped visits. We are able to generate high-quality solution for 25 mission points, 1 UGV and up to 4 UAVs in less than a minute on a standard desktop computer. Our main observation is that increasing the number of UAVs decreases the mission time and refueling stops, but does not decrease the total distance covered or total time taken.

There are a multiple directions to extend this work

1) Increasing the number of UGVs. This might need a different heuristic for generating the waypoint for the UGV and using a vehicle routing formulation to solve for UGV route.
2) Increasing the number UAVs to several orders and subsequently the number of UGVs. This might entail creating new heuristics and creating new search algorithms within the constraint programming formulation.
3) Allowing period re-computation of the routing problem for UGV and UAV as the mission progresses based on information updates.
4) Allowing multiple or periodic visits of the UAV to some mission points.
5) Adding varies uncertainties such as the travel time and service time to the formulation.

REFERENCES

[1] Yaniv Altshuler, Alex Pentland, and Alfred M Bruckstein. Optimal dynamic coverage infrastructure for large-scale fleets of reconnaissance UAVs. In *Swarms and Network Intelligence in Search*, pages 207–238. Springer, 2018.

[2] Bruno De Backer, Vincent Furnon, Paul Shaw, Philip Kilby, and Patrick Prosser. Solving vehicle routing problems using constraint programming and metaheuristics. *Journal of Heuristics*, 6(4):501–523, 2000.

[3] Michael Freed, Will Fitzgerald, and Robert Harris. Intelligent autonomous surveillance of many targets with few UAVs. In *Proceedings of the Research and Development Partnering Conference, Department of Homeland Security, Boston, MA*, 2005.

[4] Google. Google OR-tools. `https://developers.google.com/optimization`, 2021. Online; accessed Feb 2, 2021.

[5] Stephen R Griffiths. Remote terrain navigation for unmanned air vehicles. 2006.

[6] Tanya E Kannon, Sarah G Nurre, Brian J Lunday, and Raymond R Hill. The aircraft routing problem with refueling. *Optimization Letters*, 9(8):1609–1624, 2015.

[7] Samir Khuller, Azarakhsh Malekian, and Julián Mestre. To fill or not to fill: The gas station problem. *ACM Transactions on Algorithms (TALG)*, 7(3):1–16, 2011.

[8] David Levy, Kaarthik Sundar, and Sivakumar Rathinam. Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. *Mathematical Problems in Engineering*, 2014, 2014.

[9] Parikshit Maini, Kaarthik Sundar, Sivakumar Rathinam, and PB Sujit. Cooperative planning for fuel-constrained aerial vehicles and ground-based refueling vehicles for large-scale coverage. *arXiv preprint arXiv:1805.04417*, 2018.

[10] Vera Mersheeva. *UAV routing problem for area monitoring in a disaster situation*. PhD thesis, PhD thesis, 2015.

[11] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[12] Paul Shaw, Vincent Furnon, and Bruno De Backer. A constraint programming toolkit for local search. In *Optimization software class libraries*, pages 219–261. Springer, 2003.

[13] Kaarthik Sundar, Saravanan Venkatachalam, and Sivakumar Rathinam. Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. In *2016 American Control Conference (ACC)*, pages 6489–6494. IEEE, 2016.

[14] Christopher Alexander Arend Tatsch. *Route Planning for Long-Term Robotics Missions*. West Virginia University, 2020.

[15] Mirco Theile, Harald Bayerlein, Richard Nai, David Gesbert, and Marco Caccamo. UAV coverage path planning under varying power constraints using deep reinforcement learning. *arXiv preprint arXiv:2003.02609*, 2020.

[16] Christos Voudouris and Edward PK Tsang. Guided local search. In *Handbook of metaheuristics*, pages 185–218. Springer, 2003.

[17] Christos Voudouris, Edward PK Tsang, and Abdullah Alsheddy. Guided local search. In *Handbook of metaheuristics*, pages 321–361. Springer, 2010.

[18] Gregory A Wilkin and Xiuzhen Huang. K-means clustering algorithms: implementation and comparison. In *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, pages 133–136. IEEE, 2007.