Noname manuscript No.
(will be inserted by the editor)

Coordinated route planning of multiple fuel-constrained unmanned aerial systems with recharging on an unmanned ground vehicle for mission coverage

Subramanian Ramasamy[†] · **Jean-Paul F. Reddinger**[‡] · **James M. Dotterweich**[‡] · **Marshal A. Childers**[‡] · **Pranav A. Bhounsule** [†]

Received: date / Accepted: date

Abstract Small Unmanned Aerial Systems (sUAS) such as quadcopters are ideal for aerial surveillance because of their runway independence, terrain-agnostic maneuverability, low cost, and simple hardware. However, battery capacity constraints limit the effective range and endurance of sUASs, requiring human intervention to replace batteries or perform manual recharging for longer operations. To increase their range, an Unmanned Ground Vehicle (UGV) may provide recharging depot as needed. The problem is then to find optimal paths for the UGV and sUASs to visit mission points and sUAS-UGV rendezvous points for recharging. We present a three-tiered heuristics to solve this computationally hard combinatorial optimization problem: (1) K-means clustering is used to find UGV waypoints, (2) a traveling salesman formulation (TSP) is used to solve the optimal UGV route, and (3) vehicle routing problem formulation (VRP) with capacity constraints, time windows, and dropped visits is used to solve for sUAS routes. We use constraint programming for optimization of the TSP and VRP, achieving a solution for 25 mission points and up to 4 sUASs in about a minute on a standard desktop computer. We also found that constraint programming solvers are $7 - 30$ times faster, but $4 - 15\%$ sub-optimal compared to mixed-integer solvers, which provide exact solutions. Further, we used constraint programming solvers in a Monte-Carlo approach to evaluate the role of mission spread, number of clusters, and number of sUASs on the optimal solution. Our contribution is the development of heuristics for route selection of sUAS-UGVs that produces high quality solutions as more mission points and sUASs are added without substantially increasing the computational time.

This work was supported by ARO grant W911NF-14-S-003

[†] Subramanian Ramasamy and Pranav A. Bhounsule are with
Dept. of Mechanical and Industrial Engineering, University of Illinois Chicago, IL, 60607 USA.
E-mail: sramas21@uic.edu and E-mail: pranav@uic.edu

[‡] Jean-Paul F. Reddinger, James M. Dotterweich, Marshal A. Childers are with
DEVCOM Army Research Laboratory, Aberdeen Proving Grounds, Aberdeen, MD 21005 USA.
E-mail: jean-paul.f.reddinger.civ@army.mil and E-mail: james.m.dotterweich.civ@army.mil and E-mail: marshal.a.childers.civ@army.mil

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

1 INTRODUCTION

Aerial vehicles such as quadcopters are ideally suited for missions such as surveillance, reconnaissance, environment and traffic monitoring, search and rescue, and border patrol [8]. Their relatively low cost, simple hardware, runway independence, and terrain-agnostic maneuverability enables one to deploy several small Unmanned Aerial Systems (sUASs) for large-scale area coverage [2]. However, a key bottleneck is their limited battery capacity, which typically restricts them to about $15 - 20$ minutes of flight time [29].

The flight time and subsequently the coverage area of sUASs can be increased by having several recharging depots spread across the area [17]. However, in non-urban environments (e.g., rural areas) and in hostile environments (e.g., battlefields) it would be expensive or tactically impossible to have recharging depots placed in advance [6]. A viable alternative is to have multiple sUASs recharge on multiple Unmanned Ground Vehicles (UGV) that can provide sUAS recharging: when the sUAS is low on fuel it coordinates with the UGV to find a rendezvous point, lands on the UGV, recharges itself, and continues the mission.

Given a set of mission points, the number of sUASs, and the fuel-constraints and other constraints on the speed and service time, the problem is to plan an optimum route (e.g., minimum distance) for the sUAS and the UGV such that all mission points are served and the sUASs never run out of battery charge. This is a combinatorial optimization problem that is non-deterministic polynomial time or NP-hard. Such problem can be solved in finite time by designing heuristics that are problem dependent. This paper presents heuristics for planning the route of the UGV and sUAS to achieve coordinate route planning while minimizing a suitable objective (distance travelled by the sUASs) and meetings mission constraints (e.g., fuel constraints, service time constraints, vehicle speed constraints).

Khuller et al. [11] were the earliest ones to solve vehicle routing problem with fuel constraints. They considered the 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. [10] considered the problem of finding the route of a fuel-constrained aircraft to visit a set of waypoints with a set of aerial recharging waypoints. They compared a mixed-integer linear programming (MILP) formulation with a DP formulation and found that DP outperforms MILP.

Levy et al. [13] and Sundar et al. [27] considered extensions to multiple fuel-constrained Unmanned Aerial Systems (sUASs). The goal here was to minimize the distance travelled by multiple fuel-constrained sUASs 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. Similar to these works, Bung Duk Song et al. [26] considered a multiple heterogeneous sUAS path planning problem in which automatic Logistics Service Stations (LSS) are used for sUAS fuel replenishment.

Younghoon Choi et al. [33] implemented a new Coverage Path Planning (CPP) problem for solving the routes of a fleet of sUASs with energy constraints. The authors' novelty comes in formulating a column generation approach to deal with non-linear energy consumption, where traditional arc-based optimization approaches do not accurately estimate such function.

Radzki et al. [20] proposed a robust approach to delivery planning of small Unmanned Aerial Systems (sUAS) for disaster relief missions. The authors formulated an algorithm for planning of the routes of sUASs such a way that it ensures the proper execution of a certain delivery mission irrespective of any change in the weather conditions.

Andrew et al. [12] solved a two-tier route optimization problem of homogeneous sUASs and repair vehicles for network exploration and failure repair. Their problem first solves for the optimal route of sUASs to explore the potential failure locations followed by the optimal route of repair vehicles to address the failure regions located by the sUASs.

Andres et al. [1] proposed a novel approach for global path optimization of sUASs by combining Traveling Salesman Problem and continuous optimal control formulations where the latter takes the sUAS dynamics and constraints into consideration. Sung et al., addressed the optimal zoning problem of Unmanned Aerial Vehicles using Genetic Algorithm to optimize package delivery services.

Maini et al. [15] considered the problem of routing a single fuel-constrained sUAS 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 sUAS 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 sUAS and UGV. We consider an extension of the problem considered by Maini et al. These extensions are: we consider multiple fuel constrained sUASs, 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 sUASs, and the paths for the sUASs.

We list some other works that are related to sUAS-UGV coordination in other settings and/or use clustering approach.

Manyam et al., [16] solved the problem of cooperative routing of sUAS-UGV to visit a set of mission points while being within a radius of each other to enable communication. They cast the resulting problem as a mixed integer programming problem and solved using a custom-written branch-and-bound algorithm. Petitprez et al., [19] considered deployment of sUAS and UGV such that the UGVs pick up sUASs that have land on fixed locations after visiting mission points. The objective was to minimize the operational cost while maximizing the inspection performance, a contradictory set of objectives that are solved using multi-objective optimization using genetic algorithms and capacitated arc routing problem. Their problem did not have a temporal aspect to it. Liu et al., [14] considered the problem of choosing recharge stations and flight routes for sUAS to visit given mission points. They cast the problem as a binary optimization problem, but solved it using heuristics that first cluster the mission points and then use local searches to solve for a path. Bard et al. [3] considered the problem of clustering a set of customers into zones such that a single vehicle can serve all the zones while meeting delivery and pick-up times. Since the pickup and delivery schedule is random, a probabilistic traveling salesman problem is formulated and solved using heuristics that break the problem into two parts, solve them individually, and then connect back to form the complete solution. Dondo et al., [5] used clustering approach to solve a heterogeneous fleet vehicle routing problem with time windows. Initially, the customers are clustered of customers based upon node locations, load, and time windows. Then vehicles are assigned to the cluster followed by their sequencing. Finally, the nodes within the cluster are ordered and arrival times are computed. They used a standard mixed integer linear programming formulation which was solved using branch and bound.

Multiple past researchers have considered the path planning of fuel-constrained sUASs with fixed recharging stations. However, changing to a mobile recharging station, a UGV in this case, adds a significant complexity to the problem. The reasons are two-fold; one, the UGV path has to be planned and two, the sUAS-UGV path is coupled. Our scientific contribution is the development of heuristics to plan the UGV path and then use local search methods to compute sUAS paths that ensures all constraints are met. Our heuristics enables us to decouple the problems to an extent. It also enables us to add more sUASs or mission points without substantial increase in the computation time.

Our main contribution is the solution of a relatively complex sUAS-UGV route planning problem using heuristics.

1. We formulate a novel three-tier optimization: 1) K-means clustering to fix UGV waypoints; 2) UGV route selection using a traveling salesman problem formulation (TSP); and 3) sUAS route selection using a vehicle routing problem formulation (VRP) with capacity constraints (to enforce fuel limits), time windows (to enable rendezvous with the UGV), and dropped visits (to allow sUAS to visit the UGV at some locations and not all).
2. Use of OR-tools provides high quality solutions in relatively short times; we solved upto 25 missions point, 1 UGV and 4 sUASs in about 60 seconds

We solve TSP and VRP using constraint programming within Google's OR-Tools$^{\mathrm{TM}}$framework [7]. The novelty of our work compared to previous work is the formulation of heuristics to solve the coupled route planning problem that includes multiple sUASs, fuel constraints, and travel and service constraints. Our earlier work showed limited results on an example scenario involving 25 mission points [23]. We expanded the current version to include 960 optimizations that cover a range of mission distributions, number of sUASs, and cluster size. Our heuristics enable us to achieve optimal solution under a minute on a standard desktop computer. This opens up the possibility of real-time optimization during practical implementation.

The rest of the paper is organized as follows: Section 2 introduces our coordinated sUAS-UGV routing problem, followed by the formulation for optimizing the ground vehicle route and aerial vehicle routes in a tiered fashion and the solution approach to solve such problem. Section 3 presents the results obtained from solving such multi-tiered problem. Section 4 discusses the limitations present in our model and how those will be addressed in the future research. The paper concludes with Section 5.

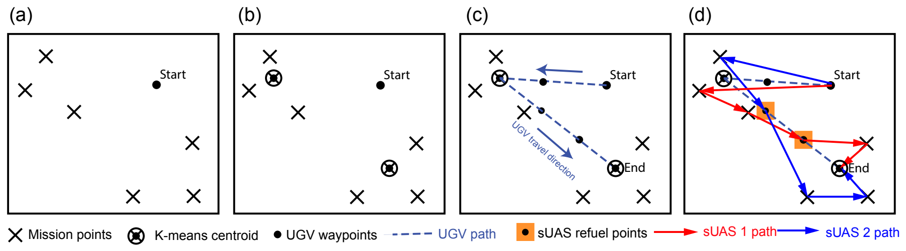

Fig. 1 An overview of the problem and proposed solution: An example scenario is shown in (a). The problem is to get a pre-specified number of sUASs to travel over a set of mission points (shown by crosses). Due to fuel constraints, a single sUAS cannot cover all mission points and hence needs fuel depots which are provided by a moving UGV whose route also needs to be planned. A three-tiered solution is proposed. (b) Tier 1 - Find the centroids of the mission points using K-means clustering, (c) Tier 2- Use a traveling salesman problem formulation to plan the UGV route, and (d) Tier 3 - Use a vehicle routing problem formulation with time windows, capacity constraints, and dropped visits to plan the sUAS route.

2 METHODOLOGY

2.1 Overview of the problem and its solution

The overall objective is to plan the path of K fuel-limited Unmanned Air Vehicles (sUASs) to pre-specified mission points while minimizing the total distance travelled by all sUASs. The sUASs may be recharged by docking on a single Unmanned Ground Vehicle (UGV). The path of the UGV and sUAS-UGV rendezvous locations also needs to be computed. Fig. 1 (a) illustrates a typical scenario. The mission locations (black cross marks) and the starting point ('start') of the UGV and sUAS are pre-specified.

There are several constraints that need to be met. The velocity of the sUAS is fixed and the velocity of the UGV is bounded. The sUAS battery capacity is fixed, while the UGV has unlimited fuel. Since the velocity is constant, the battery capacity of the sUAS is assumed to be directly proportional to the flight time. The sUAS is assumed to dock on a stationary UGV during recharging and recharge to full battery capacity. Both UGV and sUAS are stationary during recharging of the sUAS. The recharging time of the sUAS is constant and is independent of the remaining battery capacity.

We solve the problem in a tiered fashion: first we solve for the UGV route by formulating and solving a traveling salesman problem and then the sUAS route using vehicle routing problem formulation. Fig. 1 (a) shows the mission points (crosses) and the start location for 2 sUASs and a single UGV. Because the sUASs are fuel-constrained, they are unable to travel to all mission points. Thus, the UGV path has to be planned so that it can provide recharging depots. As shown in Fig. 1 (b), we use k-means clustering to find centroids for the mission points (circle with a cross and a dot). Next, as shown in Fig. 1 (c), we use a traveling salesman problem formulation to solve for the UGV route (blue dashed line) using the centroids as nodes. We add more waypoints to the resulting path (black dots). Finally, we formulate and solve a vehicle routing problem with capacity constraint (to ensure we meet fuel constraints), time windows (to enable rendezvous with the UGV), and dropped visits (to enable sUAS to drop visiting some of the UGV

Algorithm 1 K-MEANS ALGORITHM

```
Input: $k$, $[x_1, x_2, ..., x_n]$;
1: Randomly place $k$ centroid points $c_1, c_2, ..., c_k$
2: while not StoppingCondition() do
3:    for each observation $x_i$ do
4:       evaluate nearest centroid point $c_j$ by evaluating
          Euclidean distance min D $(x_i, c_j)$;
5:       Assign observation $x_i$ to the nearest centroid $j$;
6:    end for
7:    for each do cluster $j = 1, 2, ..., k$
8:       recompute $c_j = \frac{1}{n_j} \sum_{x_i \rightarrow c_j} x_i$;
9:    end for
10: end while
```

nodes). Thus, we can obtain the path of the 2 sUASs, including their recharging spots on the UGV (orange square with black dot). Although we chose 2 clusters for the k-means and 2 sUASs, it is unclear if this is the best choice. Hence, we vary the number of clusters and sUASs and re-solve the problem to find their optimum numbers for different spread or density of mission points.

2.2 Problem Formulation

We follow a three-tiered approach. First, we solve for waypoints for the UGV using k-mean clustering (Sec. 2.2.1). Second, we solve for UGV route using a traveling salesman problem formulation (Sec. 2.2.2). Third, we solve for sUAS route using a vehicle routing problem formulation (Sec. 2.2.3).

2.2.1 K-means clustering

We use K-means clustering to find suitable waypoints for the UGV [32]. K-means clustering is a technique to group n observations into k clusters. Each of these k clusters has a central location, which is the centroid of the cluster. Our goal is to find the k centroids. This problem is NP hard, so we resort to the heuristic Algorithm 1. We give a brief description of the algorithm.

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. We repeat these two steps until *StoppingCondition()* that no observation changes cluster membership.

2.2.2 UGV route using traveling salesman formulation

As described in the previous section, we have already found a sparse set of waypoints using k-means clustering. Using these waypoints as vertices, we formulate and solve a Traveling Salesman Problem (TSP) find a route for the UGV. We assume that the starting location of the UGV is pre-specified, but our algorithm chooses the ending location. We assume that there is only a single UGV with unlimited fuel capacity.

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 being the 'start' vertex, 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 as follows,

$$min \sum_{i \in V'} \sum_{j \in V'} c'_{ij} x'_{ij} \tag{1}$$

$$s.t., \sum_{i \in V'} x'_{ij} = 1, \qquad \forall j \in V' \setminus \{0, k\} \tag{2}$$

$$\sum_{j \in V'} x'_{ij} = 1, \qquad \forall i \in V' \setminus \{0, k\} \tag{3}$$

$$\sum_{j \in V'} x'_{0j} = \sum_{i \in V'} x'_{ik} = 1, \quad \{0, k\} \in V' \tag{4}$$

$$\sum_{i \in Q} \sum_{j \in Q} x'_{ij} \leq |Q| - 1, \quad \forall Q \subset \{1, ..., k\}, |Q| \geq 2 \tag{5}$$

$$x'_{ij} \in \{0, 1\}, \quad \forall i, j \in V' \tag{6}$$

The objective in Eq. 1 is to minimize the total distance traveled by the UGV. The constraints shown in Eq. 2 and Eq. 3 ensures that the UGV visits each vertex once, that is, balancing the incoming and outgoing number of vehicles at a particular vertex. Constraint in Eq. 4 ensures that the vehicle must start from a given 'start' vertex and ends at the 'end' vertex, that is, it does not loop back to the start vertex. Although constraint Eq. 4 is satisfied by constraints Eq. 2 and Eq. 3, we present them separately for the sake of completeness. Constraint in Eq. 5 ensures that there are no sub-tours. Finally, constraint Eq. 6 represents the binary decision variables.

2.2.3 sUAS routes using vehicle routing problem formulation

As described in the previous section, we have already found a path for the UGV. Using this path, we assign a sufficient number of vertices on the path as possible recharging depots for the sUAS-UGV rendezvous. We assume that each of the K sUASs starts at the same location as the UGV, formulate a vehicle routing problem with capacity constraints to account for fuel limits, time windows to allow for rendezvous, and dropped visits to allow the sUAS to visit some of the many vertices on the UGV path. We constrain the sUAS to a fixed speed, pre-specify the battery capacity and service time as the sUAS lands and waits on the UGV.

The set of all UGV waypoints is denoted by $D = \{0, 1, 2, ..., m\}$. There are $n-m$ pre-specified mission points which belong to the set $M = \{m+1, ..., n\}$. The set of all vertices is then $V = M \bigcup D = \{0, 1, 2, ..., m, m+1, ..., n\}$. The set of all edges denotes all possible connections between the vertices $E = \{(i, j)|i, j \in V, i \neq j\}$.

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 fuel constraints, time windows, and dropped visits as follows,

$$min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{7}$$

$$s.t., \sum_{i \in V} x_{ij} = 1, \quad \forall j \in M \setminus D \tag{8}$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in M \setminus D \tag{9}$$

$$\sum_{i \in V} x_{ij} <= 1, \quad \forall j \in D \setminus \{0\} \tag{10}$$

$$\sum_{j \in V} x_{ij} <= 1, \quad \forall i \in D \setminus \{0\} \tag{11}$$

$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{im} = K, \{0, m\} \in D \tag{12}$$

$$f_j \leq f_i - (c_{ij} x_{ij}) + L_1(1 - x_{ij}), \quad \forall i \in V, j \in V \setminus D \tag{13}$$

$$f_j = Q, \quad \forall j \in D \tag{14}$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \tag{15}$$

$$t_j \geq t_i + (s_i + (c_{ij} x_{ij})) - L_2(1 - x_{ij}), \quad \forall i \in V, j \in V \tag{16}$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \tag{17}$$

$$x_{ij} = 0, \quad \forall i \in D, \forall j \in D \tag{18}$$

$$x_{ij} = 1 \rightarrow f_i \geq c_{ij}, \quad \forall i \in V \setminus D, \forall j \in D \tag{19}$$

$$x_{ij} = 1 \rightarrow f_i = Q, \quad \forall i \in D, \forall j \in V \setminus D \tag{20}$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \quad \forall j \in D, \forall i \in V \setminus D \tag{21}$$

$$x_{mj} = 0, \quad \forall m \in D, \forall j \in V \tag{22}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \tag{23}$$

$$f_i > 0, f_i \in \mathbb{R}_+ \quad \forall i \in V \tag{24}$$

$$t_i > 0, t_i \in \mathbb{Z} \quad \forall i \in V \tag{25}$$

$$s_i \geq 0, s_i \in \mathbb{Z} \quad \forall i \in V \tag{26}$$

$$Q > 0, \quad Q \in \mathbb{R}_+ \tag{27}$$

$$L_1, L_2 > 0, \quad L_1, L_2 \in \mathbb{R}_+ \tag{28}$$

The objective is Eq. 7 is to minimize the total distance traveled by the all the sUASs. Constraints in Eq. 8 and Eq. 9 represents the flow conservation where the inflow of a certain sUAS should be equal to the outflow of that sUAS at any vertex among the mission vertices M. Constraints in Eq. 10 and Eq. 11 denotes the optional stops the sUAS can take on the UGV vertices D, i.e., dropped visits. Next, constraint in Eq. 12 also represents the flow conservation but here it is represented for start and end vertices, where the number of sUASs leaving the start vertex must be equal to the number of sUASs reaching the end vertex. The start vertex and end vertex correspond to the first and last vertex of the UGV route. The constraint in Eq. 13 is the Miller-Tucker-Zemlin (MTZ) formulation [18] for sub-tour elimination. This constraint ensures that none of the sUAS batteries are depleted out while eliminating sub-tours. In this constraint, L_1 denotes a large number. This constraint becomes active only when there is a flow between vertices i and j and subtracts from the sUAS fuel based on distance between the two vertices. The fuel consumption of sUAS depends upon the distance traveled by them. It is directly proportional to the distance traveled between two vertices i and j. Constraint Eq. 14 states that if the vertex is a recharging UGV stop, then UGV has to refuel the sUAS to its full capacity Q. Constraint Eq. 15 is the condition that the sUAS's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. 16 denotes that the cumulative arrival time at j^{th} node is equal to the sum of cumulative time at the node i, t_i, the service time at the node i, s_i, and the travel time between nodes i and j, $c_{ij}x_{ij}$. Here L_2 denotes a large number which helps to eliminate sub-tour constraints similar to Eq. 13. Constraint Eq. 17 is the time window constraint that the vehicle visits a certain vertex in the specified time window for that node. In this problem, the mission nodes are not constrained by time as the sUASs have the liberty to visit those mission points that benefits them according to the travel of UGV. This means that whenever sUAS needs to get refueled, it would be easier for it to go to the UGV to refuel. Constraints Eq. 18 restricts that the two consecutive visits made by the sUASs should not be consecutive UGV stops. Constraints Eq. 19 - Eq. 21 represents the indicator constraints where the constraints to the right side of the arrow should hold if the binary decision variable x_{ij} is equal to 1. If x_{ij} is equal to zero, then the constraints to the right side of the arrow may be violated. The constraint in Eq. 19 that if there is travel from any mission vertex i to the UGV vertex j, then the fuel level at the i^{th} node should be atleast equal to the distance traveled between them because the fuel consumption in this problem is assumed to be linearly proportional to the distance traveled. Constraint in Eq. 20 indicates that if there is travel from the UGV vertex i to any mission vertex j, then the fuel level at the i^{th} node should be the maximum fuel capacity of the sUAS as it is recharging to its full capacity at the UGV stop. The constraint in Eq. 21 makes sure that if any sUAS comes to the refuel vertex to recharge, then there must exist an arc between that refuel node and a mission node to maintain the flow conservation. Constraint in Eq. 22 denotes that there should not exist any flow once the vehicle has reached the end node m. Eq. 23 is a binary decision variable that is responsible for flow between the edges. Eq. 24 represents the continuous decision variable that monitors the fuel level at any node and has zero as the lower bound value. Eq. 25 represents the integer decision variable that computes the cumulative time of sUAS's route and has zero as the lower bound. Eq. 26 denotes the service time at the respective nodes, which is a positive integer with a lower bound equal to zero. Eq. 27 represents the maximum fuel capacity of a sUAS Finally, Eq. 28 denotes the large numbers used in the constraints Eq. 13 and Eq. 16.

Table 1 Constraint Quantity analysis

Equation #	Number of constraints	Equation #	Number of constraints
8	$V-D$	16	V^2
9	$V-D$	17	$2\times V^2$
10	$D-2$	18	D^2
11	$D-2$	19	$(V-D)\times D$
12	2	20	$(V-D)\times D$
13	$(V-D)\times V$	21	$((V-D)\times D)-(V-D)$
14	$D\times V$	22	$(V-D)$
15	$(2\times(V-D))\times V$		
$SUM = 6V^2-2D^2+DV+2V-2$			

2.3 Solution using Constraint Programming (CP)

Table 1 gives an itemized lists the number of constraints in the sUAS problem formulation. The equation numbers in the table correspond to the equations from the Sec. 2.2.3. In the table, the notation V denotes the total number of vertices, including all possible UGV stops and the mission points, while D denotes only the UGV stops. The grand sum of all constraints $SUM = 6V^2-2D^2+DV+2V-2$. Thus, the number of constraints scale as the square of the number of mission points.

We used Gurobi Mixed Integer Linear Programming (MILP) to solve for small instances of this problem [9]. For $k = 2$ clusters, $D = 9$ and $V = 34$ would give a total constraint of 7146 and is solved about 40 sec using Gurobi on a standard desktop (3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system). However, if $k = 4$ then $D = 17$ and $V = 42$ would give constraints of 10802 and is solved in 240 sec using Gurobi. Thus, Gurobi takes significantly higher time as the number of constraints are increased. Thus, it does not scale very well for a larger number of constraints.

Instead, we used Google's OR-Tools$^{\mathrm{TM}}$to generate the results in this paper [7] mainly for its speed of solution. OR Tools uses constrained programming (CP) [24,25] to solve TSP and VRP problems. Constraint programming or constraint optimization is a tool for solving hard combinatorial optimization problems by searching for solutions that satisfy a list of constraints. Using Google's OR-tools for $k = 2$ clusters and $k = 4$ clusters as described earlier, required only 60 sec, and the solution was about marginally better, about 4%, compared to Gurobi.

OR-Tools$^{\mathrm{TM}}$uses a search tree, local search, and meta-heuristics to find feasible and, subsequently, the most optimal solutions. At the heart of OR-tools$^{\mathrm{TM}}$is a CP-SAT solver [7]. 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 [28]). Starting with the "start" node, the decision builder connects the node that has the

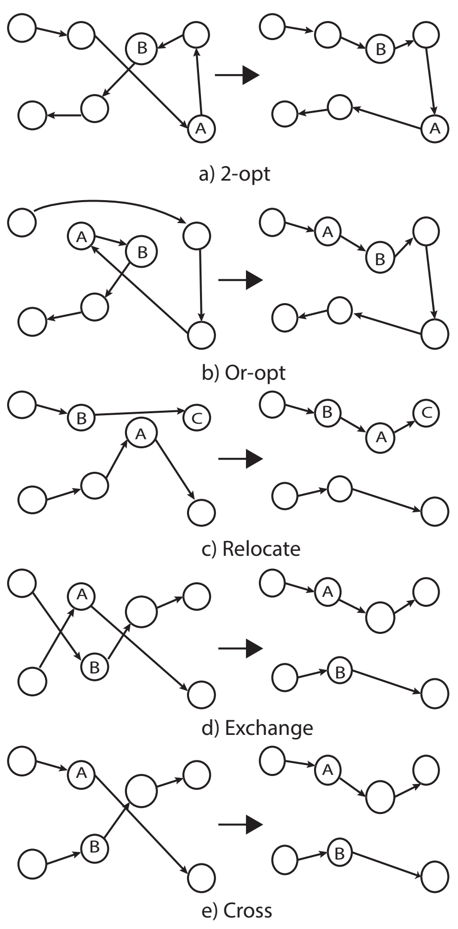

Fig. 2 Move operators using in Constraint Programming [4]

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^{TM}uses a local search to find the best solution in the neighborhood of the current solution.

This local search proceeds by a move operator that rewires the nodes and checks for feasibility and cost. These moves are repeated until a termination criteria, such as no improvement of the objective. There are 5 move operators. These are listed next and shown in Fig. 2 and is taken from [4].

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$^{\text{TM}}$use meta-heuristics. We use the Guided Local Search (GLS) in our problem [30]. In GLS, we add a penalty term to the objective function O leading to an augmented objective O' function. The penalty term is dependent on the neighborhood of the solution x through a set of features F. The augmented objective function is [4]

$$O'(x) = O(x) + \lambda \sum_{i \in F} f_i(x) p_i c_i \tag{29}$$

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 [31]), 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. For more information about the implementation of OR-Tools and the flow of multi-tiered optimization, we have uploaded a repository containing the simulation program in Github [21].

3 RESULTS

We were interested in investigating how the distribution of mission points, the number of clusters, and the number of sUASs affects the solutions of the coordinates planning of sUAS-UGV routes. Figure 3 shows how the mission spread was chosen. We chose 25 mission points over an area of 8×8 kms, but vary the mission spread. We started off with three cluster centroids randomly placed on the map. Then we chose four densities (ρ) for the mission points, $\rho = 1, 2$ being clustered while $\rho = 3, 4$ being uniformly spread out. For each density level $\rho = 1, 2, 3, 4$, we chose 20 different mission spreads. For each mission spread, we optimized for clusters $k = 2, 3, 4$ and sUASs $K = 1, 2, 3, 4$. Thus, we had 4(density) $\times$ 20(mission spread) $\times$ 3(clusters) $\times$ 4(sUASs) = 960 optimization. All the results are in Table 2.

In all optimizations, we enforce the following assumptions and constraints. Each sUAS travels at a uniform speed of 10 m/s and has a total flight time of 15 min. Thus, each sUAS can travel 9 km between successive recharges. Hence, for the area of 8×8 kms, depending on the mission spread and number of sUASs, it is possible to travel across all missions without a single recharge. There is a single UGV in all optimization and it can vary its speed from 1.5 m/s to 4.5 m/s. The time to recharge the sUAS once docked on the UGV, known as the service time, is fixed at 5 min. Once the sUAS docks on the UGV, both of them don't move for a time equal to the service time.

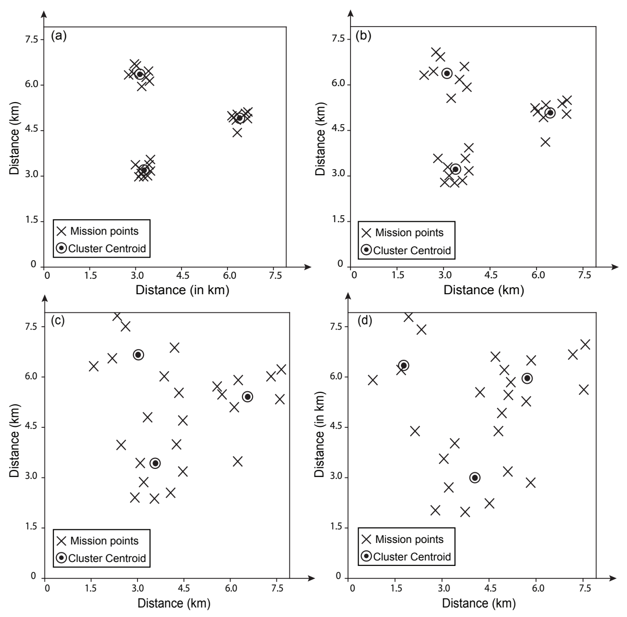

Fig. 3 An example showing how density of mission locations was chosen. Three cluster centroids were chosen randomly. Keeping these centroids the mission points were spread out. Mission density is denoted by ρ (a) $\rho = 1$ (most dense and clustered) (b) $\rho = 2$ (clustered), (c) $\rho = 3$ (spread out), and (d) $\rho = 4$ (lease dense and most spread out).

We used Constraint Programming (CP) solver in Google's OR-Tools$^{\text{TM}}$to solve both, the UGV route and the sUAS route. We used Python 3.9.4 and performed the computations on a 3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system. Each optimization took about 60 seconds, thus we could do all 960 optimizations in about 16 min.

For a density ρ we choose 20 random position for the mission points and run the optimization for a number of sUASs K and given cluster size k. Our metrics for comparing the results for the 20 optimizations are the total distance covered by all sUASs, the feasibility percentage, number of recharges, total time, and mission time. The total distance is the sum of the distances travelled by all sUASs and is the objective of the optimization. The feasibility percentage is the percentage of feasible solutions out of 20 runs. The number of recharges is the total number of recharges taken by all the sUASs. The total time is the sum of the travel and service time for all sUASs. The mission time is the sum of the travel and service time of the sUAS that takes the most time among all sUASs. Table 2 tabulates the results using the above metric. We analyzed these results in more details in Figs. 4 - 9.

Figures 4 compares the different metrics (distance, time, and recharging stops) for sUASs for different mission densities. We assess the results for two sUASs, $K = 2, 3$. The metrics increase as the missions get more spread out with $\rho = 1$ being most clustered and $\rho = 4$ being least clustered. When comparing the different sUASs, it can be seen that the total distance (a) and total time (c) are more for $K = 3$ than $K = 2$ while the total recharging stops (b) and mission time (d) are less for $K = 3$ than $K = 2$. Thus, more sUASs shorten the mission time at the cost of increasing the travel distance.

Table 2 Results for 960 optimization. Here K is the number of sUASs, ρ is the density of mission points with $1, 2$ is clustered and $3, 4$ is uniformly spread out, k is the number of clusters. Each row in the table, that is, for a given K, given density ρ, and given cluster k, corresponds to 20 optimizations. In each optimization there is randomization of mission points.

(a) Total distance (objective), feasibility percentage, and recharging stops

K	ρ	Total distance (in kms.)			Feasibility %			Total recharging stops		
		k=2	k=3	k=4	k=2	k=3	k=4	k=2	k=3	k=4
1	1	-	13.1±1.69	13.8 ± 1.8	0	55	65	0 ± 0	1 ± 0	1.38±0.48
	2	18 ± 0	16.4±2.09	19 ± 2.09	5	20	100	1 ± 0	1.75±0.43	2.75±0.83
	3	-	-	35 ± 4.34	0	0	90	0 ± 0	0 ± 0	4.83±1.17
	4	-	-	38.5±4.73	0	0	65	0 ± 0	0 ± 0	6.23±1.42
2	1	14.5±0.56	14.4±0.59	14.5±0.46	100	100	100	0.4 ± 0.49	0.4 ± 0.49	0.1 ± 0.3
	2	18.5±1.29	18.3±1.21	19.4±1.81	100	100	100	0.85±0.36	0.95±0.22	1.1 ± 0.83
	3	31 ± 4.45	31 ± 4.4	33 ± 8.95	35	85	95	2.57±1.05	2.77±0.94	3.3 ± 1.45
	4	33.8±2.73	36.9±7.49	37.5±5.95	20	80	95	3 ± 0.71	3.93±1.10	4.26±1.37
3	1	17.2±0.58	16.8±0.53	16.9±0.49	100	100	100	0 ± 0	0.2 ± 0.4	0.1 ± 0.3
	2	20.6±1.05	20.4±1.02	21.3±1.62	100	100	100	0.5 ± 0.5	0.75±0.54	0.75±1.04
	3	31.3±3.97	30.5±3.72	31.9±7.97	100	100	90	1.65±0.79	1.85±0.91	2.17±0.83
	4	37.6±5.97	35.8±6.84	38.8±7.93	100	100	90	2.3 ± 0.95	2.65±1.15	3.56±1.30
4	1	20 ± 0.53	19.3 ± 0.6	19.5±0.51	100	100	100	0.2 ± 0.4	0.4 ± 0.49	0.1 ± 0.3
	2	23.3 ± 1.1	23.1±1.16	24.1±1.56	100	100	100	0.45±0.50	0.55±0.50	0.8 ± 1.03
	3	32.9±4.17	33.6±3.51	34.9±4.46	100	100	90	1.65±1.01	1.95±0.86	1.94±0.78
	4	37.8±5.48	36.7±5.61	38.1±6.68	100	95	75	1.65±1.11	1.89±1.12	2.53±1.50

(b) Total time and mission time

K	ρ	Total time (in min.)			Mission time (in min.)		
		k=2	k=3	k=4	k=2	k=3	k=4
1	1	-	163.7 ± 6.02	169.18 ± 9.1	–	163.7 ± 6.02	169.18 ± 9.1
	2	165.72 ± 0	171.03 ± 6.98	192.27 ± 8.58	165.72 ± 0	171.03 ± 6.98	192.27 ± 8.58
	3	-	-	230.37±12.39	-	-	230.37±12.39
	4	-	-	242.82±14.39	-	-	242.82±14.39
2	1	161.43 ± 3.67	161.23 ± 3.64	160.21 ± 4.09	93.36 ± 14.96	98.63 ± 22.40	89.48±416.08
	2	171.29 ± 5.91	173.78 ± 4.84	178.66 ± 9.81	107.92±12.07	121.24±18.48	116.82±22.11
	3	205.68±15.35	208.52±14.56	208.08±50.77	117.95 ± 7.07	141.06±10.64	157.62±45.54
	4	214.05 ± 8.34	224.92±21.56	230.19±16.56	119.05 ± 3.98	136.44±12.95	190.32±25.26
3	1	168.28 ± 0.97	170.27 ± 3.84	67.99 ± 5.18	168.28 ± 0.97	88.36 ± 19.85	76.68 ± 16.31
	2	177.94 ± 4.26	179.77 ± 3.84	182.27±10.61	91.19 ± 19.37	111.13±28.31	95.39 ± 31.21
	3	202.60±12.40	205.53±13.86	212.77±13.94	90.04 ± 8.43	118.12±28.31	135.35±27.61
	4	219.23±14.99	219.57±19.04	233.91±19.77	98.91 ± 13.64	124.52±18.10	166.09±32.66
4	1	168.20±33.11	180.21 ± 3.58	178.64 ± 4.17	78.65 ± 35.78	89.69 ± 21.99	78.65 ± 33.18
	2	186.21±4.315	187.69 ± 5.54	192.61±11.16	84.58 ± 20.18	101.61±29.47	94.27 ± 31.91
	3	211.20±13.77	215.19±11.29	219.42±13.10	90.37 ± 10.10	113.84±19.98	116.56±30.68
	4	219.30±15.84	219.52±16.02	229.52±20.02	94.31 ± 10.51	112.15±21.02	137.69±41.12

Fig. 5 depicts four distributions of mission points, with decreasing density from (a) to (d). We use cluster size $k = 3$ for computing UGV route and $K = 2$ sUASs to serve the mission points. Since the cluster centroids are at the same location for all cases, the UGV route, including its distance, remains the same. We show the metrics in the side box in each figure. We can see that for (a) and (b) with $\rho = 1, 2$ (most dense) have lower values for the 4 metrics when compared with (c) and (d) with $\rho = 3, 4$. Clearly, as the missions get spread out, the sUASs have to travel a larger distance, which increases the number of recharging stops, the total time, and the mission time.

Figures 6 compares the different metrics (distance, time, and recharging stops) for different cluster size in the K-mean clustering. We present results for two different densities $\rho = 2$ (clustered) and $\rho = 3$ (uniformly spread) for 3 sUASs. The

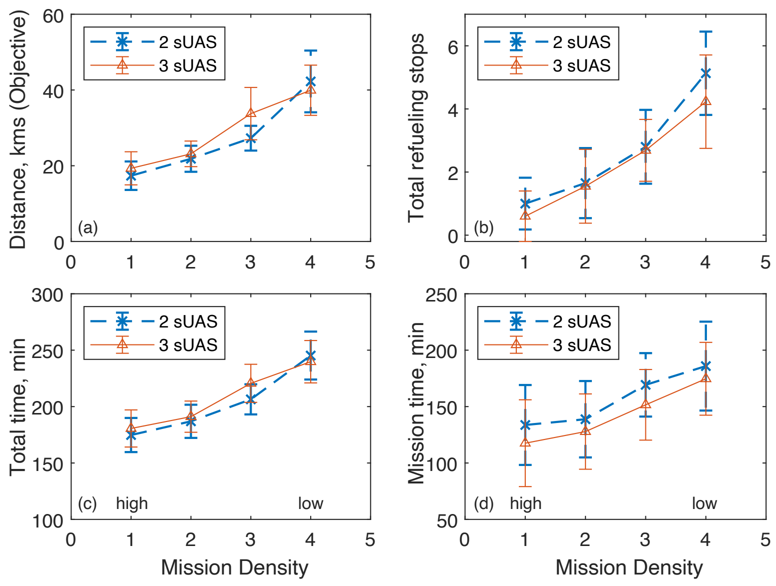

Fig. 4 Metrics as function of mission density for 2 and 3 sUASs: (a) Total Distance (objective), (b) Total recharging stops, (c) Total time, (d) Mission time, all as a function of density ρ from $\rho = 1, 2$ (missions are clustered) to $\rho = 3, 4$ (missions are uniformly spread out).

metrics increase as the density changes from clustered $\rho = 2$ to uniformly spread out $\rho = 3$ as expected. The clusters size k does not seem to show a correlation for highly clustered missions. For instance, for density level $\rho = 2$, the total distance and total time are almost the same for different cluster sizes, while the recharging stops and mission time show an increase for $k = 3$. The clusters size k seems to show some correlation for spread out missions. For instance, for density level $\rho = 3$, the total distance decreases, but the total time, mission time, and recharging stops all increase with an increase in the cluster size.

Fig. 7 (a) shows the mission points for one scenario with density of $\rho = 3$. We chose different cluster sizes for the k-mean clustering. The plots (a), (b), and (c) correspond to $k = 2, 3, 4$ respectively. We can observe that the UGV route increases as k increases. We then solve each scenario with the same number $k = 3$ sUASs. We note that the total distance (objective) is almost the same for all three optimizations. The recharging stops and the total time for $k = 2$ (b) and for $k = 3$ (c) are almost the same, while those for $k = 4$ are slightly higher. The extra total time for $k = 4$ is probably because of the added recharging stop, which adds a service time to the total time. Finally, the mission time increases across from (b) to (d) because of the unequal sharing of missions as the cluster size increases.

Figures 8 compares the different metrics (distance, time, and recharging stops) for different number of sUASs. We use two density levels, $\rho = 1, 2$ (both clustered) and chose a cluster size of $k = 3$. The metrics increase as the density decreases from $\rho = 1$ to $\rho = 2$ as expected. As the number of sUASs increase, the total distance (objective) (a) and total time (c) increases while the recharging stops (b) and the mission time (d) decreases. We may attribute the reduction in these latter metrics to better sharing of the missions among the sUASs.

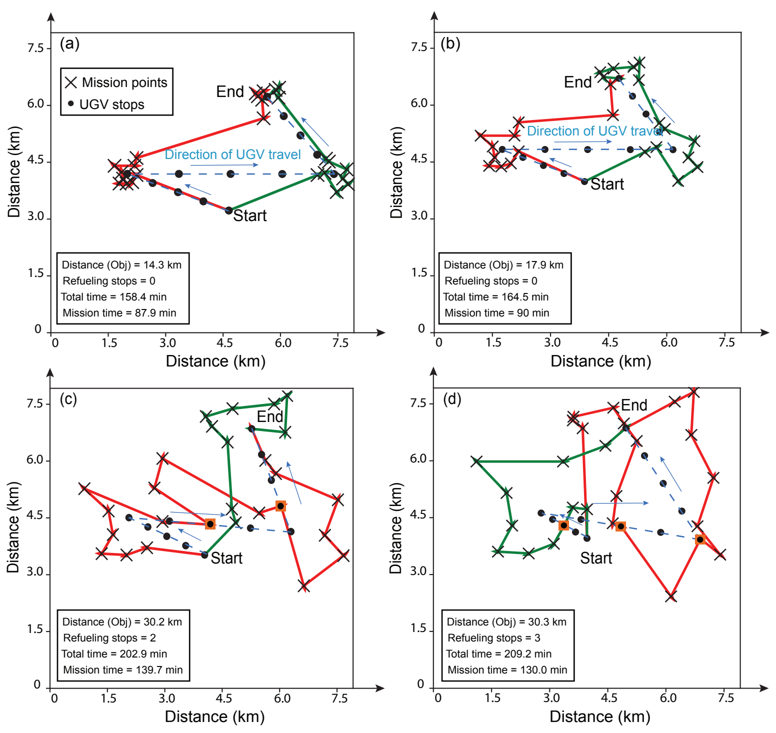

Fig. 5 Optimizations illustrating the solution for a mission density changes. Here cluster $k = 3$ and sUAS $K = 2$. The mission densities are (a) $\rho = 1$ (dense and clustered), (b) $\rho = 2$ (less dense than $\rho = 1$, but clustered), (c) $\rho = 3$ (uniformly spread out), $\rho = 4$ (uniformly spread out with spread greater than $\rho = 3$)

Fig. 9 shows the solution obtained by changing the number of sUASs for the same distribution of mission with density $\rho = 2$ and cluster size $k = 3$. Since the density and cluster size are the same, the UGV route is the same across all scenarios. The overall distance and total time increases, and the refueling stops and mission times decreases with the increase in the number of sUASs. The increase in total distance with increase in sUASs may be attributed to inefficient sharing of missions, which also increases the total time. However, the benefit of inefficient sharing is that recharging stops decrease, which also reduces the mission time.

4 DISCUSSION

In this paper, we have presented heuristics for planning the path of a multiple fuel-constrained small Unmanned Aerial Systems (sUASs) and a single Unmanned Ground Vehicle (UGV) such that the sUASs can visit a set of mission points while minimizing the total distance covered and without running out of fuel by docking on the UGV to recharge. We solved the problem in a tiered fashion; first, we use the mission points to create waypoints for the UGV using K-means clustering; second, we solved for the UGV route using the Traveling Salesman Problem formulation

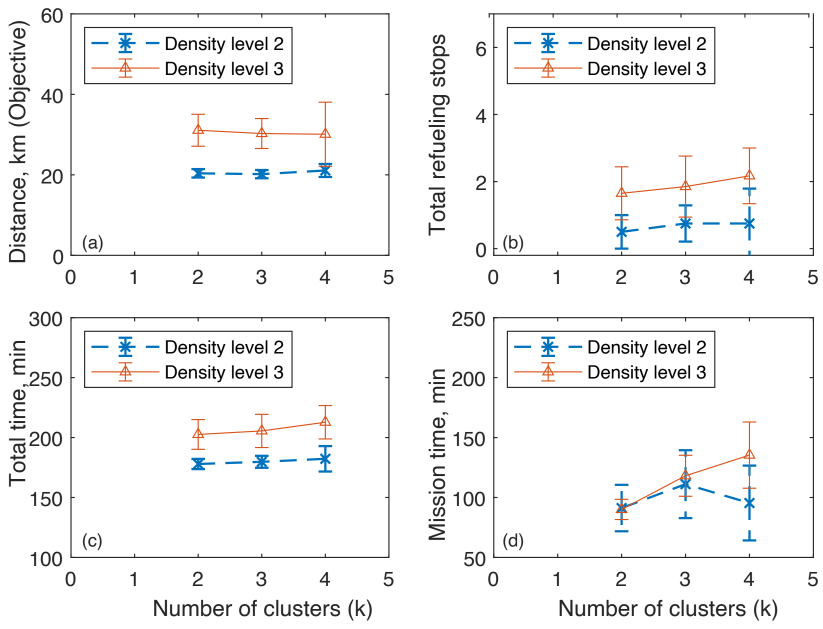

Fig. 6 Metrics as function of number of clusters for 3 sUASs for two density levels $\rho = 2$ is clustered and $\rho = 3$ is uniformly spread out. (a) Total Distance (objective), (b) Total recharging stops, (c) Total time, (d) Mission time, all as a function of number of clusters.

using the waypoints as vertices; third, we solved for the sUAS route by using mission points and using waypoints on the path of the UGV as vertices and using a Vehicle Routing Problem formulation with capacity constraints (fuel limits), time windows (to match UGV rendezvous points for recharging), and dropped visits (to allow sUAS to drop some of the rendezvous points).

We solved the mixed integer programming problem (MILP) formulated in Sec. 2.2.3 using constraint programming which is based on heuristics. The same problem may be solved using MILP solvers such as Gurobi [9]. We compared solutions obtained by both solvers in a limited number of cases and the results are shown in Tab. 3. Each row in the table corresponds to a certain setting given by the number of sUAS's (K), the mission density (ρ), and the cluster size k. For each setting, we choose 10 scenarios and run the optimization. The table gives the objective value and the optimization time for each solver. The optimality gap is the percent difference in the objective value between OR-Tools and Gurobi. It can be seen that Gurobi is between $4-15\%$ more optimal than OR-tools. However, while OR-Tools takes a maximum of 10 seconds (which is the termination criteria for the optimization), Gurobi takes between $71-300$ seconds, indicating the computational superiority of using OR-Tools. Thus, we conclude that OR-Tools is $7-30$ times faster than Gurobi, but only $4-15\%$ sub-optimal. Hence, we used OR-Tools in our calculations.

We make some general observations based on our limited study of 25 mission points with at least one mission point beyond the coverage area of a single sUAS. The spread of the mission points is an important criteria that affects the solution

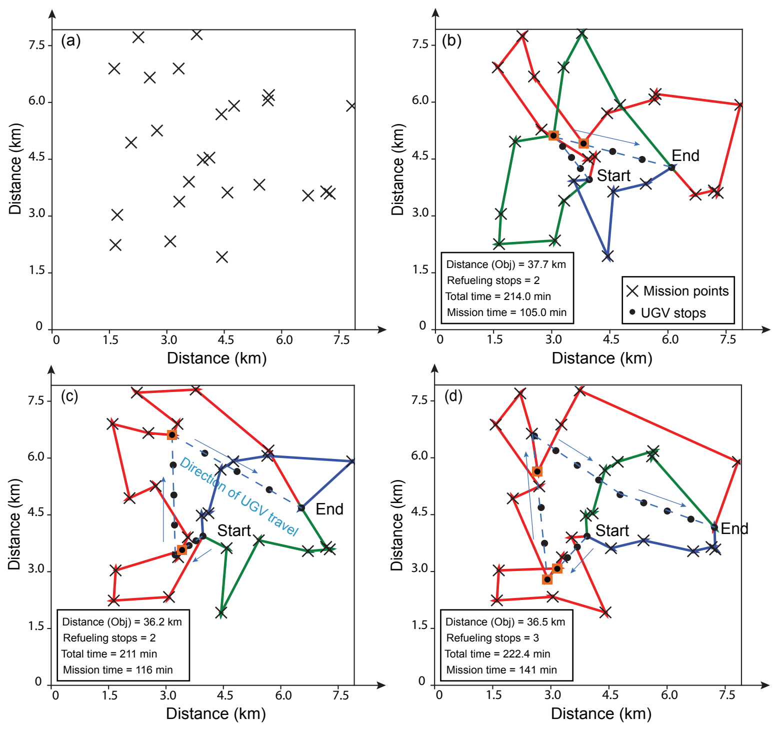

Fig. 7 Optimizations illustrating the solution different cluster sizes. Here mission density $\rho = 3$ and sUAS $K = 3$. The spread of missions is shown in (a). The cluster sizes are (b) $k = 2$, (c) $k = 3$, and (d) $k = 4$

feasibility and the nature of optimal solution. When the missions are well clustered ($\rho = 1, 2$ here), we find feasible solutions with $K \geq 2$ sUASs, but as the missions spread out ($\rho = 3, 4$), we need $K \geq 3$ sUASs to serve all mission points. The results suggest that there is a minimum number of sUASs to generate feasible solutions which depend on the mission spread and the fuel level.

The number of sUASs has a strong correlation with the optimization outcomes. The total distance travelled by all sUASs increases as the number of sUASs increases. This is because multiple sUASs are now sharing mission points closer to each other, resulting in a larger travel distance. The increased total distance leads to larger total travel time. However, the maximum distance travelled by any given sUAS is smaller, which correlates to a shorter mission time. The number of recharging stops also decreases as more sUASs have a bigger collective range.

The cluster size chosen shows a weak correlation with the outcomes. For a given density and number of sUASs, the total distance covered, recharging stops, total time, and mission time remained almost constant.

We have not shown solutions for a single cluster $k = 1$ and multiple sUASs $K = 1, 2, 3, 4$. This is because we end up with very few feasible solutions and hence should be avoided. The $k = 1$ cluster size leads to a rather restrictive UGV route that prevents the VRP formulation from finding feasible solutions.

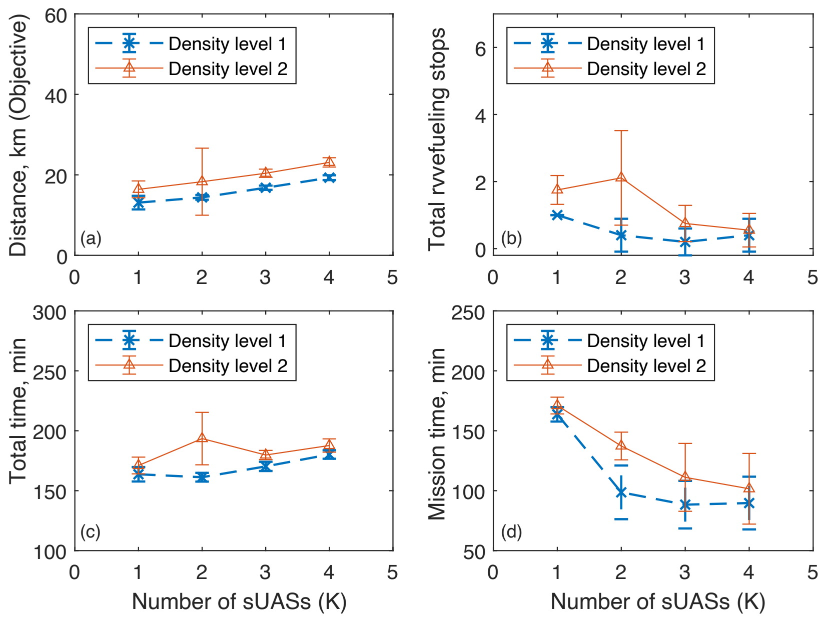

Fig. 8 Metrics as function of number of sUASs for cluster size of 3 for two density levels, both of which correspond to clustered distributions. (a) Total Distance (objective), (b) Total recharging stops, (c) Total time, (d) Mission time, all as a function of number of sUASs.

Table 3 Comparison between constraint programming solver (OR-tools) and mixed-integer linear programming solver (Gurobi). Each row corresponds to particular combination of K, ρ, and k where K represents the number of sUAS, ρ represents the mission density, and k represents the number of clusters. These are for 10 randomly chosen scenarios for each row. The optimality gap is the percent error between Gurobi and OR-tools

Mission distribution pattern	Gurobi		OR-Tools		Optimality gap (%)
	Objective value (in km.)	Optimization time (in s)	Objective value (in km.)	Optimization time (in s)	
K=1, ρ=2,k=4	17.15±0.85	300 ± 0	20.41±1.23	10 ± 0	15.5±5.80
K=2, ρ=3,k=3	27.99±2.89	155.8 ± 135.52	30.04±4.19	10 ± 0	8.37±5.42
K=3, ρ=3,k=3	29 ± 2.77	71.3 ± 117.96	30.59±3.76	10 ± 0	4.9 ± 4.14
K=4, ρ=4,k=2	35.95±3.15	117.2 ± 76.26	37.79±4.11	10 ± 0	4.3 ± 4.17

The key advantage of our framework is the decomposition of a complex problems into three-stages. This simplifies the problem formulation. The use of constraint programming as a solver gives high quality solution in fraction of seconds for the scenarios considered here (25 missions points, 1-4 sUAS an 1 UGV).

The prime disadvantage of our tiered heuristics is the cascading effect of parameter choices. In our case, the quality of the k-means clustering determines the UGV route, which then determines the sUAS route. Thus, a poor choice of cluster size can affect the final solution. We can see this for $K = 1$ and $k = 2, 3$ where we get no feasible solution. The K-mean clustering and UGV route selection do

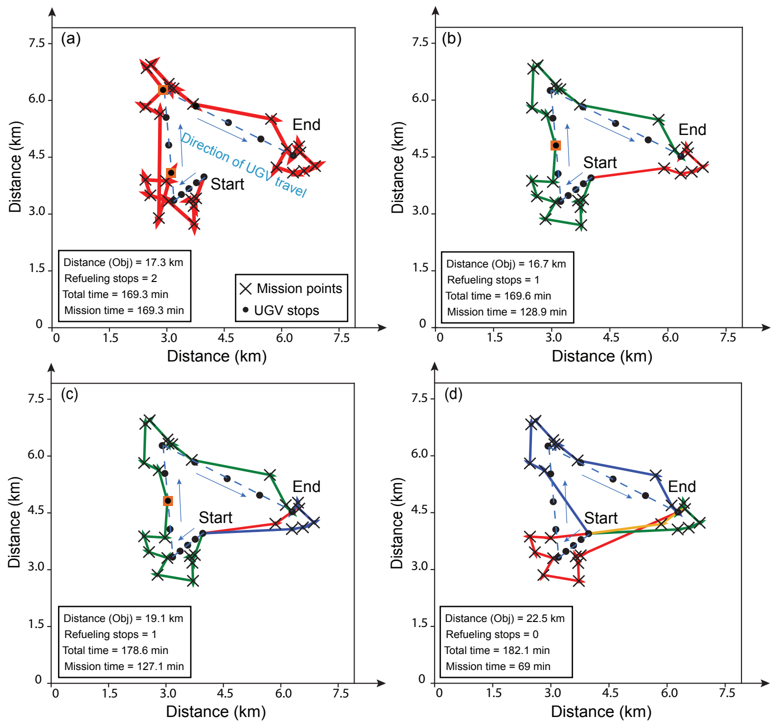

Fig. 9 Optimizations illustrating the solution for different number of sUASs. Here mission density $\rho = 2$ and cluster size is $k = 3$. The number of sUASs are (a) $K = 1$, (b) $K = 2$, (c) $K = 3$, and (d) $K = 4$.

not consider the number of mission points in a cluster. If there are more mission points in a cluster than we should probably give that cluster a higher preference to be visited earlier than later. As the number of sUASs increases, we notice that there is unequal sharing of mission points (e.g., Fig. 9 (d)). 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 sUASs finish their missions simultaneously, thus allowing some sUASs to travel a shorter distance. We have overcome this issue to an extent by using genetic algorithms and bayesian optimization to tune the parameters [22].

We finish by listing directions for future work. We assumed fixed recharging time for sUAS irrespective of its existing fuel level before recharging. This is not optimal. Thus, future research address this issue in such a way that the refueling amount, and thereby the refueling time, of the sUAS on UGV depends upon the existing fuel level before any sUAS reaches the UGV from a mission to get recharged. We have assumed that UGV has indefinite fuel capacity which is not the case in practical scenarios. Future studies will consider fuel capacity of the UGV. Furthermore, the terrain chosen affects the UGV fuel usage and it will also need to be taken into account in solving the routing problem.

5 CONCLUSION

We conclude that the problem of routing multiple fuel-constrained small Unmanned Aerial Systems (sUAS) with recharging on a single Unmanned Ground Vehicle (UGV) can be solved quickly and efficiently using a tiered heuristics where the UGV route is solved first followed by the sUAS route within a constraint programming approach. We can solve routes for about 25 mission points with 1 to 4 sUAS and a single UGV in less than a minute on standard desktop computer. This opens up the possibility of real-time optimization during practical implementation. Our main observations are: (1) there is a minimum number of sUASs needed based on the fuel constraints and velocities of the sUAS and UGV and mission spread, and (2) the number of clusters need to be $k > 1$, but there is no clear correlation between cluster size and the solution. The overall distance and overall time taken increases as the missions spread out and as the number of sUASs increase. However, the mission time and the recharging stops decrease as the number of sUASs increases. Finally, we found that constraint programming solvers are $7 - 30$ times faster, but $4 - 15\%$ sub-optimal compared to mixed-integer solvers, which provide exact solutions.

Declarations

- Funding: The work by SR and PAB was funded by Army Research Laboratory grant W911NF-14-S-003.
- Conflict of interest/Competing interests: The authors declare no conflict of interest.
- Code or data availability: `https://github.com/Subram0212/JINT_paper_codes`
- Authors' contributions: Conceptualization, SR, JPR, JD, CM, PAB; methodology, SR and PAB; software and validation, SR; writing—original draft preparation, SR and PAB; writing—review and editing, JPR, JD, CM, PAB; supervision, JPR, JD, CM, PAB; project administration, PAB. All authors have read and agreed to the published version of the manuscript.'
- Ethics approval: Not applicable.
- Consent to participate: Not applicable.
- Consent for publication: All authors consent to publication.

References

1. Albert, A., Imsland, L.: Combined optimal control and combinatorial optimization for searching and tracking using an unmanned aerial vehicle. Journal of Intelligent & Robotic Systems **95**(2), 691–706 (2019)
2. Altshuler, Y., Pentland, A., Bruckstein, A.M.: Optimal dynamic coverage infrastructure for large-scale fleets of reconnaissance UAVs. In: Swarms and Network Intelligence in Search, pp. 207–238. Springer (2018)
3. Bard, J.F., Jarrah, A.I., Zan, J.: Validating vehicle routing zone construction using monte carlo simulation. European Journal of Operational Research **206**(1), 73–85 (2010). DOI https://doi.org/10.1016/j.ejor.2010.01.045. URL `https://www.sciencedirect.com/science/article/pii/S0377221710001049`
4. De Backer, B., Furnon, V., Shaw, P., Kilby, P., Prosser, P.: Solving vehicle routing problems using constraint programming and metaheuristics. Journal of Heuristics **6**(4), 501–523 (2000)

5. Dondo, R., Cerdá, J.: A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. European journal of operational research **176**(3), 1478–1507 (2007)
6. Freed, M., Fitzgerald, W., Harris, R.: 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)
7. Google: Google OR-tools. `https://developers.google.com/optimization` (2021). Online; accessed Feb 2, 2021
8. Griffiths, S.R.: Remote terrain navigation for unmanned air vehicles (2006)
9. Gurobi: Gurobi Optimization LLC. `https://www.gurobi.com/` (2021). Online; accessed Sep 19, 2021
10. Kannon, T.E., Nurre, S.G., Lunday, B.J., Hill, R.R.: The aircraft routing problem with refueling. Optimization Letters **9**(8), 1609–1624 (2015)
11. Khuller, S., Malekian, A., Mestre, J.: To fill or not to fill: The gas station problem. ACM Transactions on Algorithms (TALG) **7**(3), 1–16 (2011)
12. Lee, A.C., Dahan, M., Weinert, A.J., Amin, S.: Leveraging suas for infrastructure network exploration and failure isolation. Journal of Intelligent & Robotic Systems **93**(1-2), 385–413 (2019)
13. Levy, D., Sundar, K., Rathinam, S.: Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. Mathematical Problems in Engineering **2014** (2014)
14. Liu, Y., Liu, Z., Shi, J., Wu, G., Chen, C.: Optimization of base location and patrol routes for unmanned aerial vehicles in border intelligence, surveillance, and reconnaissance. Journal of Advanced Transportation **2019** (2019)
15. Maini, P., Sundar, K., Rathinam, S., Sujit, P.: Cooperative planning for fuel-constrained aerial vehicles and ground-based refueling vehicles for large-scale coverage. arXiv preprint arXiv:1805.04417 (2018)
16. Manyam, S.G., Casbeer, D.W., Sundar, K.: Path planning for cooperative routing of air-ground vehicles. In: 2016 American Control Conference (ACC), pp. 4630–4635 (2016). DOI 10.1109/ACC.2016.7526082
17. Mersheeva, V.: UAV routing problem for area monitoring in a disaster situation. Ph.D. thesis, PhD thesis (2015)
18. Miller, C.E., Tucker, A., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. J. ACM **7**, 326–329 (1960)
19. Petitprez, E., Georges, F., Raballand, N., Bertrand, S.: Deployment optimization of a fleet of drones for routine inspection of networks of linear infrastructures. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 303–310 (2021). DOI 10.1109/ICUAS51884.2021.9476674
20. Radzki, G., Golinska-Dawson, P., Bocewicz, G., Banaszak, Z.: Modelling robust delivery scenarios for a fleet of unmanned aerial vehicles in disaster relief missions. Journal of Intelligent & Robotic Systems **103**(4), 1–18 (2021)
21. Ramasamy, S.: Uav-ugv routing code. `https://github.com/Subram0212/JINT_paper_codes` (2022). Online; accessed Feb 13, 2022
22. Ramasamy, S., Md, M., Reddinger, J.P.F., Dotterweich, J.M., D., H.J., Childers, M.A., Bhounsule, P.A.: Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization. In: 2022 International Conference on Unmanned Aircraft Systems (ICUAS) (2022)
23. Ramasamy, S., Reddinger, J.P.F., Dotterweich, J.M., Childers, M.A., Bhounsule, P.A.: Cooperative route planning of multiple fuel-constrained unmanned aerial vehicles with recharging on an unmanned ground vehicle. In: 2021 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 155–164 (2021). DOI 10.1109/ICUAS51884.2021.9476848
24. Rossi, F., Van Beek, P., Walsh, T.: Handbook of constraint programming. Elsevier (2006)
25. Shaw, P., Furnon, V., De Backer, B.: A constraint programming toolkit for local search. In: Optimization software class libraries, pp. 219–261. Springer (2003)
26. Song, B.D., Kim, J., Morrison, J.R.: Rolling horizon path planning of an autonomous system of uavs for persistent cooperative service: Milp formulation and efficient heuristics. Journal of Intelligent & Robotic Systems **84**(1-4), 241–258 (2016)
27. Sundar, K., Venkatachalam, S., Rathinam, S.: Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem. In: 2016 American Control Conference (ACC), pp. 6489–6494. IEEE (2016)

28. Tatsch, C.A.A.: Route Planning for Long-Term Robotics Missions. West Virginia University (2020)
29. Theile, M., Bayerlein, H., Nai, R., Gesbert, D., Caccamo, M.: UAV coverage path planning under varying power constraints using deep reinforcement learning. arXiv preprint arXiv:2003.02609 (2020)
30. Voudouris, C., Tsang, E.P.: Guided local search. In: Handbook of metaheuristics, pp. 185–218. Springer (2003)
31. Voudouris, C., Tsang, E.P., Alsheddy, A.: Guided local search. In: Handbook of metaheuristics, pp. 321–361. Springer (2010)
32. Wilkin, G.A., Huang, X.: K-means clustering algorithms: implementation and comparison. In: Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007), pp. 133–136. IEEE (2007)
33. Younghoon, C., Youngjun, C., Briceno, S., Mavris, D.N.: Energy-constrained multi-uav coverage path planning for an aerial imagery mission using column generation. Journal of Intelligent & Robotic Systems **97**(1), 125–139 (2020)