Heterogenous vehicle routing: comparing parameter tuning using genetic algorithm and bayesian optimization

Subramanian Ramasamy[1†‡], Md Safwan Mondal[1†‡], Jean-Paul F. Reddinger[2], James M. Dotterweich[2], James D. Humann [3], Marshal A. Childers[2], Pranav A. Bhounsule [1†]

***Abstract*— Heterogeneous vehicles (e.g., unmanned ground vehicle (UGV) and unmanned aerial vehicle (UAV)) are best suited for surveillance application over large areas. UAVs are fast, but fuel limited, while, UGVs have a larger fuel capacity, but are relatively slow. When UAVs are combined with UGVs they can provide larger coverage at a relatively fast speed. The UAV may also be recharged on the UGV as needed. The resulting route optimization problem is computationally complex, but may be solved relatively fast using heuristics. In this paper, we solve for a mission route using a two-level optimization; (1) the UGV route is assigned using heuristics with free parameters, (2) the UAV route is solved using a vehicle routing problem formulation with capacity constraints, time windows, and dropped visits. However, this open-loop two-level optimization may yield non-optimal solutions or fail completely because of poor choice of UGV parameters. Our primary objective is to explore closed loop optimization where the free parameters of the UGV routes are optimized using Bayesian optimization and Genetic algorithms. Our results show that both methods produce good quality solutions, but bayesian optimization is computationally more efficient than genetic algorithm.**

1. INTRODUCTION

The availability of simple-to-control and low-cost unmanned aerial vehicles (UAVs) has opened the possibility of practical surveillance systems [1]. UAVs can provide automated surveillance for reconnaissance, weather observations, environment and traffic monitoring, search and rescue, and border patrol [6]. Although UAVs are fast, they are severely limited to relatively small area due their limited battery capacity [25].

To increase their range, UAVs may be combined with unmanned ground vehicles (UGVs). UGVs move slow and are terrain limited, but are large enough to enable docking and recharging of aerial vehicles. They may also be used to survey locations that are accessible through the roads [7]. Thus a system consisting of heterogenous vehicles consisting of UAVs and UGVs are ideally suited for automated surveillance applications (e.g., [4], [29]).

[1] Subramanian Ramasamy, Md Safwan Mondal, and Pranav A. Bhounsule are with the Department of Mechanical and Industrial Engineering, University of Illinois Chicago, IL, 60607 USA. sramas21@uic.edu mmonda4@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@army.mil james.m.dotterweich.civ@army.mil marshal.a.childers.civ@army.mil. [3] James D. Humann is with DEVCOM Army Research Laboratory, Los Angeles, CA, 90094 USA. james.d.humann.civ@army.mil † This work was supported by ARO grant W911NF-14-S-003. ‡ These authors contributed equally.

The use of heterogenous vehicles with different capabilities makes the route selection problem more challenging [2]. One has to consider the fuel limitations of the UAV and the speed limitations of the UGV to device successful routing paths. In order to push their limits, optimization of their routes with metrics such as reducing the time and/or the overall fuel consumption are important. In this paper, we provide a framework for optimizing the routes of heterogenous vehicles with resource constraints (e.g., fuel, speed) and terrain constraints (mission spread).

We illustrate the problem using the mission scenario shown in Figure 1 (a). The mission points are shown with black dots. The UAV range for a single charge is shown with a blue circle. The circle indicates that the UAV cannot visit all mission points on a single charge. Figure 1 (b) and (c) shows two possible solutions. In Fig. 1 (b), the UGV and UAV move together through the path $a \to b \to c$. The UAV then takes off from the UGV and travels the path $f \to g$ to return to the UGV. Then the UGV and UAV move along $d \to e$ to return to the starting point. Fig. 1 (c) shows an alternate solution. The UGV and UAV move together along a. Then the UAV takes off and its moves along $e \to f$ to return to the starting point. The UGV moves along $b \to c \to d$ and returning to the start point. If optimization criteria is UAV fuel consumption then the strategy in (b) is better because the UAV flies over a shorter distance. If optimization criteria is the total time taken to complete the missions, then (c) is better here because unlike in (b), the UGV does not have to wait for the UAV to land back on it thus (c) takes less time than (b). From this example, we note that the UGV heuristics, the stop location for the UAV to take-off and the wait time are critical based on the optimization criteria. In this study, these parameters are optimized by the genetic algorithm and bayesian optimization.

There has been considerable work on routing of fuel constrained UAVs. Levy et al. [11] and Sundar et al. [23] considered the routing of multiple fuel-constrained Unmanned Aerial Vehicles (UAVs) with recharging on fixed 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. Ren et al. [18] considered a collaborative two-UAV and one-UGV problem where the purpose of UGV as a carrier is to deploy and retrieve the flying robots on time, and the optimal take-off and landing points was solved using

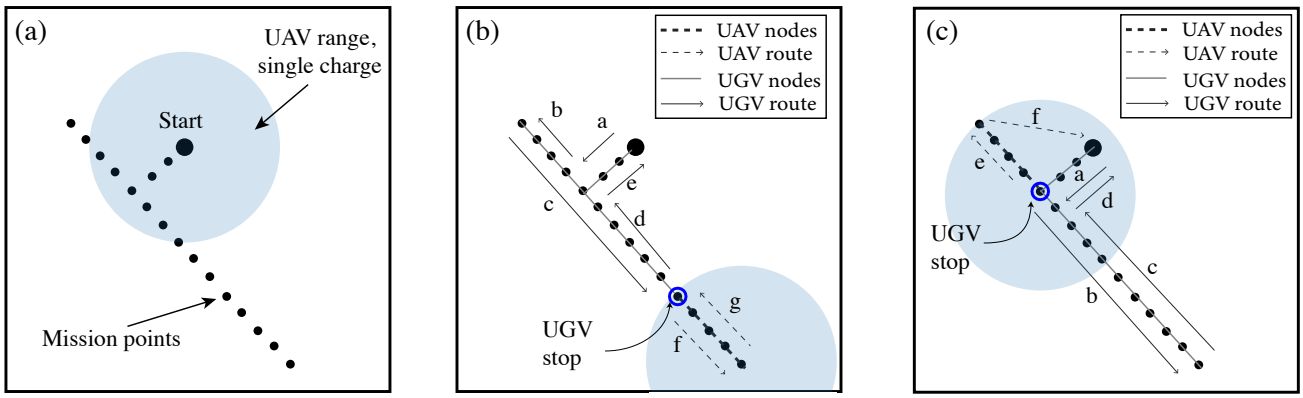

Fig. 1. **An example scenario:** (a) The mission scenario. Both, the UAV and UGV start from the same starting location. The range of the UAV is shown using a blue circle. (b) (c) Two possible solutions for mission coverage.

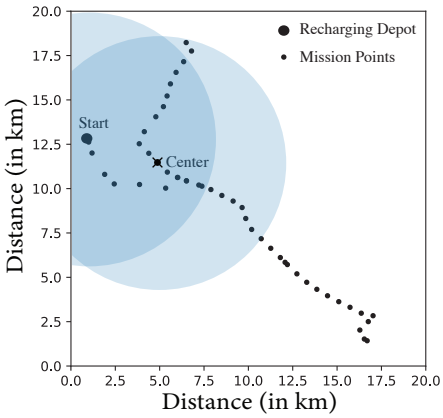

Fig. 2. The problem scenario. The UAV and UGV, both start from the recharging depot. The mission points are shown with black dots. The UAV range is shown with a blue circle.

Particle Swarm Optimization (PSO) algorithm.

Further extensions have considered routing of fuel constrained UAV and recharging on ground vehicles that are free to move on prescribed paths. This is a challenging problem because each of these vehicles have different constraints on speed and fuel capacity. Maini et al. [12] 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. Mathew N. et al. [13] presented a cooperative rendezvous planning of working robots (UAV) and one or more mobile charging robots to recharge UAVs by formulating a rendezvous scheduling problem as a Multi Generalized Traveling Salesman Problem (MGTSP) and then transforming it into a TSP for applying heuristic solvers. The authors then extended this problem for longer planning using receding horizon strategies.

We have considered extension of the problem by considering multiple fuel-constrained UAVs and a single UGV [17]. We solved the problem in a tiered fashion. First, we use K-mean clustering to create nodes for UGVs to visit and solved for the UGV path using a traveling salesman formulation. Second, using the UGV path, we formulated and solved a vehicle routing problem with capacity constraints, time windows, and dropped visits.

One of the issues using heuristics for solving the routing problems is that the heuristics have parameters that can affect the quality of the solution. Thus, some past work has considered tuning of the parameters to improve the solution. Huang et. al. [10] solved a capacitated arc routing problem using hierarchical decomposition to generate feasible solution and then used local search. A bayesian optimization was used to improve the parameters of the hierarchical decomposition. Henrio et. al. [9] considered the problem of reducing the time between consecutive visits of a series of locations to reduce the uncertainty. They used firefly algorithm which is based on flashing of fireflies to attract or mate with other fireflies. A bayesian optimization was used to tune the parameters of the firefly algorithm. Pilat [16] used genetic algorithms to improve the parameter selection in an ant colony optimization algorithm to solve the traveling salesman problem.

In this paper, we extend our past work on UAV-UGV routing using heuristics [17]. We investigate the use of genetic algorithm and bayesian optimization to improve the heuristics. The novelty of this study lies in the application of the global optimization techniques like Genetic Algorithm and Bayesian Optimization for parameterizing different heuristics parameters which can be used for solving combinatorial optimization problems in a shorter period of time.This study shows that with proper tuning of the routing parameters it is possible to obtain a larger route coverage for tiered routing of heterogeneous vehicles. The flow of the paper is as follows. We present details about the optimization method in Sec. 2. The results are in Sec. 3, followed by the Discussion in Sec. 4. Finally, the conclusion and future work is in Sec. 5

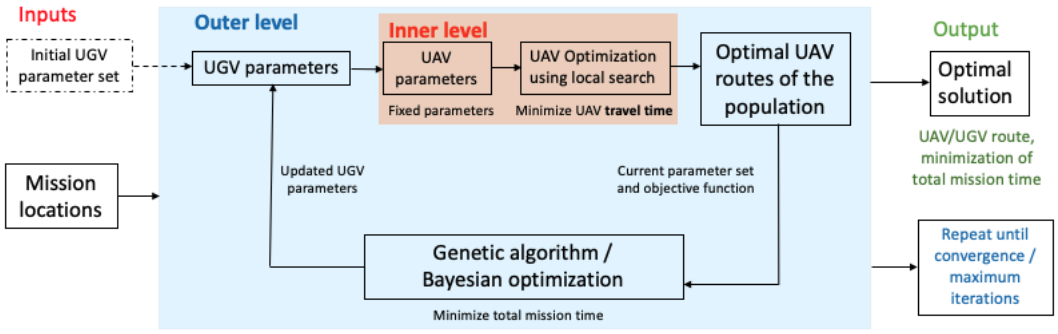

Fig. 3. Overview of the algorithm

2. METHODS

A. Problem statement

Figure 2 shows the problem scenario considered in the paper. The mission points are shown with black dots. There is a recharging depot shown as a blue circle encompassing the solid black dot. The UAV can travel on the UGV or fly by itself. The UAV may be charged on the UGV or the depot. Both, UAV-UGV have to start and end at the depot and have to visit all the mission points.

The blue circles ($radius = fuel\ capacity/2$) of Figure 2 represent the range of the UAV on a full charge; the distance that the UAV can cover is the diameter of the circle. If the UAV starts from the center of the circle on a full charge, it can return back to the center of the circle with an empty charge. We have drawn two circles which are centered approximately at $(1, 12.5)$ and $(5, 12)$. It can be observed that from the start location, the UAV cannot travel to the two mission points approximately at $(7.5, 17.5)$. However, if the UAV starts from the point $(5, 12)$ with a full charge, it can cover the two mission points and return back. To enable this solution, the UAV would need to ride with the UGV till $(5, 12)$, then visit the mission points and get refueled. This illustrates some of the intricacies of choosing an appropriate path for the UGV such that the UAV can successfully move to the mission points at the extreme ends and increase the route coverage. In this problem, we tried to optimally parameterize the UGV parameters like the rendezvous points and time periods of UAV in the UGV path with the help of global optimization techniques.

B. Solution approach

Figure 3 shows the solution approach that involves an outer- and an inner-level. The outer level block (light blue) involves heuristics to choose a UGV route. The UGV route heuristics has a few free parameters. Once these parameters are set, the inner-level block (light orange) is the UAV route optimization. Thus far, the UAV-UGV route selection is open-loop since the UGV route has not been optimized. Thus, we run an optimization on the outer loop that optimizes the free parameters in the UGV heuristics minimizing an appropriate cost, thus closing the loop. This outer-inner loop optimization proceeds till the maximum iteration limit is reached or the solution has converged. That is, there is no change in the objective value. We now describe the details of the inner-loop, the outer-loop, and the closed-loop optimization.

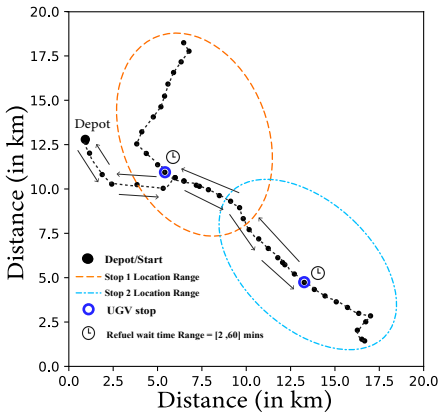

Fig. 4. Heuristics for UGV route. The UGV can make two stops on any mission points such that the first one inside the red ellipse shown with dashed lines and the second is inside the blue ellipse shown with dash-dotted lines. At each stop, the UGV can choose a time to wait (e.g., to recharge the UAV).

C. Heuristics for UGV (Outer-loop)

Our heuristics for UGV route are based on maximum fuel range of the UAV described earlier (range is show as a blue circle in Fig. 2). Figure 4 shows the heuristics for the UGV route. The UGV starts at the depot and travels along the mission points as shown by the arrow. Next, the UGV is allowed to stop anywhere in the ellipse with dashed red lines for a prescribed time. The rationale is that in choosing a stop and wait time is to give the UAV enough time to land and recharge on the UGV. Next, the UGV moves to to the bottom right side and can take another stop anywhere inside the blue ellipse with blue dash-dot lines. We have shown two random stop locations in each ellipse with a blue hollow circle. For the chosen stop locations and wait times for each stop, the UAV routing problem is formulated and solved as described in the next section.

D. Optimizing UAV route (Inner-loop)

We formulate a Vehicle Routing Problem (VRP) with capacity constraints to account for fuel limits, time windows

to allow for rendezvous, and dropped visits to allow the UAV to visit some of the many vertices on the UGV path. We constrain the UAV to a fixed speed, pre-specify the battery capacity and service time as the UAV 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. In this problem, the cost will the time traveled between two nodes 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.

The objective is Eq. 2.2 is to minimize the total time traveled by the all the UAVs. Constraints in Eq. 2.3 and Eq. 2.4 represents the flow conservation where the inflow of a certain UAV should be equal to the outflow of that UAV at any vertex among the mission vertices M. Constraints in Eq. 2.5 and Eq. 2.6 denotes the optional stops the UAV can take on the UGV vertices D, i.e., dropped visits. Next, constraint in Eq. 2.7 also represents the flow conservation but here it is represented for start and end vertices, where the number of UAVs leaving the start vertex must be equal to the number of UAVs 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. 2.8 is the Miller-Tucker-Zemlin (MTZ) formulation [15] for sub-tour elimination. MTZ constraint takes care of the sequential visit of each node by keeping track of values like fuel capacity, travel time of UAV corresponding to each node. It makes sure that if a node is visited twice, then the constraint is violated. This constraint enables that the UAV's energy is not fully drained out while eliminating loops. 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 drains the UAV energy based on time taken from the two vertices. The P_{UAV} in the constraint represents the power profile of the UAV, which basically tells the power consumption when the UAV travels from one node to another. In this problem, such a power profile for the UAV is given by the following equation.

$$P_{UAV} = 0.046v_a^3 - 0.583v_a^2 - 1.876v_a + 229.6 \quad (2.1)$$

where v_a corresponds to the velocity of UAV. In this problem, the velocity of UAV is fixed to $v_a = 10$ m/s.

$$\min \sum_{i \in V} \sum_{j \in V}, c_{ij}x_{ij} \text{ such that} \quad (2.2)$$

$$\sum_{i \in V} x_{ij} = 1, \quad \forall j \in M \setminus D \quad (2.3)$$

$$\sum_{j \in V} x_{ij} = 1, \quad \forall i \in M \setminus D \quad (2.4)$$

$$\sum_{i \in V} x_{ij} <= 1, \quad \forall j \in D \setminus \{0\} \quad (2.5)$$

$$\sum_{j \in V} x_{ij} <= 1, \quad \forall i \in D \setminus \{0\} \quad (2.6)$$

$$\sum_{j \in V} x_{0j} = \sum_{i \in V} x_{im} = 1, \{0, m\} \in D \quad (2.7)$$

$$f_j \leq f_i - (P_{UAV}c_{ij}x_{ij}) + L_1(1 - x_{ij}), \quad \forall i \in V, j \in V \setminus D \quad (2.8)$$

$$f_j = Q, \quad \forall j \in D \quad (2.9)$$

$$0 \leq f_j \leq Q, \quad \forall j \in V \quad (2.10)$$

$$t_j \geq t_i + (s_i + (c_{ij}x_{ij})) - L_2(1 - x_{ij}), \quad \forall i \in V, j \in V \quad (2.11)$$

$$t_j^l \leq t_j \leq t_j^u, \quad \forall j \in V \quad (2.12)$$

$$x_{ij} = 0, \quad \forall i \in D, \forall j \in D \quad (2.13)$$

$$x_{ij} = 1 \rightarrow f_i \geq P_{UAV}c_{ij}, \quad \forall i \in V \setminus D, \forall j \in D \quad (2.14)$$

$$x_{ij} = 1 \rightarrow f_i = Q, \quad \forall i \in D, \forall j \in V \setminus D \quad (2.15)$$

$$x_{ij} = 1 \rightarrow \sum_{i \in V \setminus D} x_{ji} = 1, \quad \forall j \in D, \forall i \in V \setminus D \quad (2.16)$$

$$x_{mj} = 0, \quad \forall m \in D, \forall j \in V \quad (2.17)$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j \in V \quad (2.18)$$

$$f_i > 0, f_i \in \mathbb{R}_+ \quad \forall i \in V \quad (2.19)$$

$$t_i > 0, t_i \in \mathbb{Z} \quad \forall i \in V \quad (2.20)$$

$$s_i \geq 0, s_i \in \mathbb{Z} \quad \forall i \in V \quad (2.21)$$

$$Q > 0, \quad Q \in \mathbb{R}_+ \quad (2.22)$$

$$L_1, L_2 > 0, \quad L_1, L_2 \in \mathbb{R}_+ \quad (2.23)$$

Constraint Eq. 2.9 states that if the vertex is a recharging UGV stop, then UGV has to refuel the UAV to its full capacity Q. Constraint Eq. 2.10 is the condition that the UAV's fuel at any vertex in V should be between 0 and maximum fuel capacity. Constraint Eq. 2.11 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. 2.8. The s_i in our problem is a decision variable in cases when the UAV travels from a refuel node. That is, the service time is basically the recharging time when it travels from a refuel node. In our problem, the recharging time depends on the existing charge present in the UAV. That is, if the fuel level is low it will take more time

to recharge. Hence, the profile of the power transfer from UGV to UAV should be taken into account based upon the fuel level present in UAV. A first order approximation of the battery recharge rate, P_r is given by:

$$P_r = \begin{cases} 310.8; & \text{E} \leq 270.4 \text{ kJ}, \\ 17.9(287.7 - E); & 270.4 < \text{E} \leq 287.7\text{kJ}. \end{cases} \tag{2.24}$$

where E represents the existing energy level on the UAV when it docks to recharge. The power transfer uses constant current until 94% of the battery capacity, with a 3.5C charge rate. After 94% capacity, it switches to a constant voltage charge. And from the existing fuel level on UAV and computing this battery recharge rate, when the UAV is charged to its maximum capacity, we would get the recharging time of the UAV. Constraint Eq. 2.12 is the time window constraint that tells the vehicle to visit a certain vertex in the specified time window for that node. In this problem, the mission nodes are not constrained by time as the UAVs have the liberty to visit those mission points that benefits them according to the travel of UGV. This means that whenever UAV needs to get refueled, it would be easier for it to go to the UGV to refuel. Constraints Eq. 2.13 restricts that the two consecutive visits made by the UAVs should not be consecutive UGV stops. Constraints Eq. 2.14 - Eq. 2.16 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. 2.14 tells that if there is travel from any mission vertex i to the UGV vertex j, then fuel level at the i^{th} node should be atleast equal to the energy consumed by the UAV when it travels from i to j. Constraint in Eq. 2.15 tells 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 UAV as it is recharging to its full capacity at the UGV stop. The constraint in Eq. 2.16 makes sure that if any UAV 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. 2.17 denotes that there should not exist any flow once the vehicle has reached the end node m. Eq. 2.18 is a binary decision variable that is responsible for flow between the edges. Eq. 2.19 represents the continuous decision variable that monitors the fuel level at any node and has zero as the lower bound value. Eq. 2.20 represents the integer decision variable that computes the cumulative time of UAV's route and has zero as the lower bound. Eq. 2.21 denotes the service time at the respective nodes, which is a positive integer with a lower bound equal to zero. Eq. 2.22 represents the maximum fuel capacity of a UAV Finally, Eq. 2.23 denotes the large numbers used in the constraints Eq. 2.8 and Eq. 2.11. The above Mixed Integer Linear Programming (MILP) formulation for UAV routing can be solved using solvers like Gurobi Optimizer [8]. For each set of UGV routing parameters, it takes a lot of time to obtain the corresponding optimal UAV routing solution using this solver. Hence, the authors resorted with other solving methods that gave quality inner-level solutions in a shorter period of simulation time.

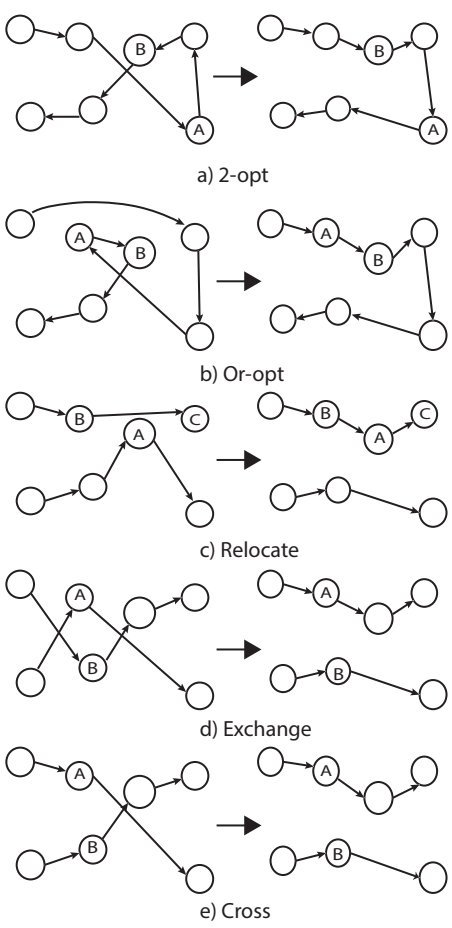

Fig. 5. Move operators using in Constraint Programming [3]

Apart from UAV, the UGV has has a certain limit on its fuel capacity. The power curve for the UGV in this work follows the following equation 2.25, where UGV velocity v_g is in m/s and power is in watts. And irrespective of the nature of power profiles for UGV and UAV, this formulation helps to find the appropriate optimal solution for this co-operative vehicle routing problem. In this study only the kinematics model of the UAV and the UGV was considered.

$$P_{UGV} = 464.8v_g + 356.3 \tag{2.25}$$

E. Solution using Constraint Programming (CP)

We used Google's OR-Tools™for implementation of the heuristics to generate the results in this paper [5] mainly for its speed of solution. Whereas Genetic algorithm and Bayesian optimization were implemented through manual Python programming for optimal tuning of the heuristics parameters in order to improve the solution quality.OR Tools uses Constrained Programming (CP) [19], [20] 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.

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 [5]. 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 [24]). 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.

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. 5 and is taken from [3].

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 Guided Local Search (GLS) meta-heuristics [26].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 [3]

$$O'(x) = O(x) + \lambda \sum_{i \in F} f_i(x) p_i c_i \tag{2.26}$$

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 [27]), 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.

Algorithm 1 Genetic Algorithm

Input: Population size, n; Maximum generations, MAX
Output: Global best solution

```
Generate initial population of n chromosomes randomly;
Set the current generation g = 0;
while g < MAX do
  if g = 0 then
    Compute the fitness value for each chromosomes;
    Increment the current generation g by 1;
  else
    Select a pair of chromosomes from the initial or
    old population based on fitness;
    Apply crossover operation on selected parents;
    Apply mutation operation on produced offspring
    with a mutation probability;
    Replace initial or old population with newly gen-
    erated population;
    Compute the fitness value for each chromosomes;
    Increment the current generation g by 1;
  end if
end while
```

F. Optimizing the parameters of the UGV heuristics

We have optimized the UAV route for a pre-selected UGV route. It is clear that changing the UGV route will change the UAV solution and consequently the optimum. In this section, we are interested in a closed-loop optimization where we would like to optimize both, the UGV and UAV solutions, thus achieving better solution. We choose 6 parameters in our UGV heuristics; the x- and y-coordinate of each of the two stop locations and the wait times at the stops. We use genetic algorithm and bayesian optimization to optimize these 6 parameters.

1) Genetic Algorithm : Genetic algorithm is a metaheuristic inspired from the process of natural selection in nature. It is an effective method to solve for global optimization problems where the objective function could potential have multiple local optimal solutions [28].

Algorithm 1 describes the workflow of the Genetic algorithm. In our problem, the population in each generation basically consists of the parameter set, in which each individual of the population is a list of parameters that constitute the UGV route and each parameter value in the parameter list are encoded to form genes. The genes, which are the encoded version of the parameters, are concatenated together into a single string to form an individual of the population. In technical terms, those individuals of the population are called chromosomes. We use binary encoding for gene encoding as it helps in improving the diversity of the solutions. We use Latin Hypercube Sampling (LHS) to sample the initial population. LHS is a sampling technique that is not purely based on random sampling, but mimics some structure in randomness. That is, LHS has a memory of previously sampled points and the same sample points or same combination of points are not sampled again unlike random sampling, where repeated sampling of same points can happen. LHS

mimics the distribution of the data and provides an efficient sample [14]. Once the initial population is formed, the fitness function, which is the objective function is computed for each individual. The individual in the population with better fitness are carried forward to the next generation. This is called elitism and it ensures that solutions with better fitness values will be retained. These solutions with then participate in a selection process to further improve the fitness. Thus process is repeated iteratively to improved the fitness till there are no more improvements indicating convergence. Selection process is carried on to select two individuals (parents) from the previous generation to produce offsprings for the next generation. We use a technique called $Tournamentselection$ for selecting the chromosomes as it performs better than other techniques in terms of algorithm workflow, convergence rate and time complexity [21]. In this selection, two individuals to be compared are picked from the population and the individual with the best solution amongst the two will be chosen as a parent for the crossover operation. In order to extend the possibility of obtaining a global optimal solution, a variant in tournament selection called $unbiasedtournamentselection$ is introduced as it eliminates the loss of diversity related to the failure of random sampling for tournament selection [22]. Instead of picking two individuals at random for comparison, the individuals are picked in a particular order viz., permutation and then compared. The two next steps are the crossover and mutation operations, help to improve the solution search space which helps achieve a global optimum. In crossover operation, the selected pair of parents are mated at a randomly chosen crossover points to produce offsprings. For this problem, 2-point crossover operator is used to produce two offsprings from the parents. After performing the crossover operation, the mutation operation is performed in which a random bit of the offspring's chromosome is flipped or mutated if the probability of the random bit exceeds a certain probability value to impart diversity in the solution. This diversity enables the solution to escape the local optima. The probability of the random bit is taken from the uniform distribution. For this problem, the mutation operator value of the GA is set to be 0.01. The GA algorithm is terminated if there is no improvement in the solution, indicating convergence or when the maximum iteration limit is reached.

2) Bayesian Optimization: Bayesian Optimization is a powerful tool for optimizing computationally expensive objective function. Mathematically, bayesian optimization acts as a global optimizer of a blackbox function $f(x)$: $x^* = \arg\min_{x \in \mathcal{X}} f(x)$, where $\mathcal{X}$ is the region of interest for finding the optimal solution. Bayesian optimization uses a probabilistic model (surrogate model) $f(.)$ based on given prior distribution $\mathcal{D}_n = [(x_1, f(x_1)), (x_2, f(x_2)), ...(x_n, f(x_n))]$ for analyzing its posterior belief at the unexplored input regions. The performance of Bayesian Optimization depends significantly on the acquisition function which balances between the exploitation and exploration of the predictive surrogate model $f(.)$ to list out the most promising points x^+ in its domain $\mathcal{X}$ and evaluates the objective at these points, $f(x^+)$. Next $(x^+, f(x^+))$ is added to the prior distribution of the surrogate model $f(.)$ and consequently the posterior distribution is updated. This process is continued iteratively till the solution converges to its optimum value or when maximum iteration limit is reached.

Algorithm 2 Bayesian Optimization

Input: sampling points $\mathcal{D}_n$, Maximum iterations k
Output: Global best solution

```
Make g(x) surrogate model on D_n;
for k = 1, 2, ..., do
    obtain x_{n+1} by optimizing acquisition function α
    x_{n+1} = arg max α(g(x)), y_{n+1} = f(x_{n+1})
    update D_{n+1} = [D_n, (x_{n+1}, y_{n+1})] ;
    update g(x) on D_{n+1}
end for
Find x* = arg min g(x), y* = f(x*)
```

Algorithm 2 shows the pseudo code for Bayesian Optimization. Bayesian optimization has two major blocks, first is the surrogate model $g(x)$ which approximates the objective function $f(x)$ based on the sampling points $\mathcal{D}_n$ and the second, the acquisition function α which helps to evaluate the important points (x_{n+1}, y_{n+1}) in the posterior. We have used Gaussian prior as the surrogate model and Expected Improvement (EI) for the acquisition function. Suppose, $\mathcal{GP}(f(\hat{x}), s^2(x))$ is the surrogate model on $f(x)$ and f_{min} is the minimum value of the objective functions among $[f(x_1), f(x_2), ...f(x_n)]$ for n given values of $[x_1, x_2, ...x_n]$. The improvement $I(x)$ of $f(\tilde{x})$ on unexplored points $\tilde{x}$ is defined as

$$I(x) = \max(f_{min} - f(\tilde{x}), 0) \tag{2.27}$$

Thus, Expected Improvement (EI) acquisition function is the expectation (average) of the variable $I(x)$ over $f(x)$ defined as below:

$$\begin{aligned} E(I(x)|f(x)) &= E[\max(f_{min} - f(\tilde{x}), 0)|f(x)] \\ &= (f_{min} - f(\hat{x}))\Phi(\frac{f_{min} - f(\hat{x})}{s(x)}) \\ &+ s(x)\phi(\frac{f_{min} - f(\hat{x})}{s(x)}) \end{aligned} \tag{2.28}$$

The point with maximum EI is chosen as the point of interest to update the prior $\mathcal{D}_{n+1}$ and the surrogate model $g(x)$. The process is continued till the maximum iteration k is reached when we get the optimal point (x^*, y^*) from the Bayesian Optimization.

3. RESULTS

We present results for the scenario shown in Fig 2. The UAV-UGV have to start and end at the Depot. The mission points are shown by black dots. All missions point needs to be covered by either the UAV or the UGV. The UAV can recharge at the Depot or at the UGV. The UAV and UGV velocities when moving are fixed at 10 m/s and 4 m/s

Parameter	Range
UGV stop 1 location (km,km)	(6.90,18.04) to (9.80,9.07)
UGV stop 2 location (km,km)	(8.64, 9.77) to (16.96,1.45)
UGV stop 1,2 wait times (min)	2 to 60

TABLE I
PARAMETER SET AND THEIR RANGE FOR GA/BO OPTIMIZATION

Parameter	GA Values	BO Values
UGV stop 1 (km,km)	(4.99,11.65)	(6.10,10.80)
UGV stop 1 wait time (min)	18	50
UGV stop 2 (km,km)	(16.96,1.45)	(16.96,1.45)
UGV stop 2 wait time (min)	3	2.45

TABLE II
OPTIMAL PARAMETERS AFTER GA/BO OPTIMIZATION

Metrics	Genetic Algorithm (GA)	Bayesian Optimization (BO)
Total time (min)	225	222
Total local-search optimizations	180	60
Computational time (min)	90	15
UGV results		
Travel time (minutes)	225	222
Energy consumed (MJ)	23.13	24.86
Mission visited	37	34
UAV results		
Travel time (minutes)	65	103
Energy consumed (kJ)	575.25	804.905
Recharging stops on UGV	1	2
Recharging stops on Depot	0	0
Missions visited	9	12

TABLE III
COMPARISON BETWEEN GA/BO ON METRICS FROM THE OPTIMAL SOLUTION.

respectively. The UAV and UGV fuel capacity are 287.7 kJ and 25.01 MJ respectively.

The heuristics for the UGV are depicted in Fig. 4. There are 6 free parameters for the UGV optimization. Four for the x- and y-stop locations (stop 1 and stop 2) of the UGV and 2 wait times. Table I shows the ranges for both these parameters. The objective function is to minimize the total time to visit all mission points.

We used Python 3 for the optimizations (GA/BO/OR-Tools) performed the computations on a 3.7 GHz Intel Core i9 processor with 32 GB RAM on a 64-bit operating system.

Table II gives the optimal parameter set computed by GA/BO. The UGV 1 stop locations are slightly different but the wait times are substantially different. The UGV stop 2 locations and times are identical; the stop 2 corresponds to the far right corner of the missions. These results are discussed in detail later in this section.

Table III compares the key metrics between GA and BO optimizations. The objective of the optimization was to reduce the total time. Both optimization gave similar results with BO being marginally better than GA by 3 min. GA needed 3 times more local-search optimizations and took 6 times more computational time than BO. Thus, BO has more computational efficiency than GA. The UGV results show that GA and BO had similar times, energy consumed, but the UGV travel more more mission points in GA rather than BO. The UAV results show substantial difference between the two methods. The travel time, energy consumed in BO are about 1.5 times higher. This is because the BO travels to 3 more mission points than GA.

Figure 6 and 7 shows the optimum route produced by GA and BO respectively. The main difference between the two solutions is in the UAV route. In GA, the UAV visits the missions further away from the intersection of the branches as shown in Fig 6 a), b), and c) then recharges on the UGV, completes the remaining missions as shown in d) and goes to the start. In BO, the UAV visits the missions closer to the intersection of the branches as shown in Fig 7 a), b), then recharges once on the UGV. Then the UAV visits the top most mission points as shown in c) and d) then recharges a second time on the UGV. Finally, the UAV completes the remaining missions as shown in e) and f). Thus, it can be observed that the order of choosing the mission points in BO increases the travel time of UAV. However, since the UGV is the slower of the two vehicles, the travel time of the UGV determines the total time taken to cover all mission points. Since the UGV travel is almost the same except for the slightly more wait time for the UAV to return back in the BO solution, the difference in the objective is only 3 minutes.

4. DISCUSSION

In this paper, we presented a framework for optimizing routes of heterogenous vehicles, a UGV and UAV, with fuel constraints. The key idea is to solve the problem in a tiered fashion. First, we use heuristics to decide the UGV route. Second, we optimize the UAV route using a local search. We identify key parameters in the UGV heuristics and optimize them iteratively with UAV route using genetic algorithm and bayesian optimization. Starting from an infeasible solution, both algorithms are able to give solutions with similar cost, but different solutions.

Our objective was the time taken to visit all the mission points. Both GA/BO gave almost the same time. However,

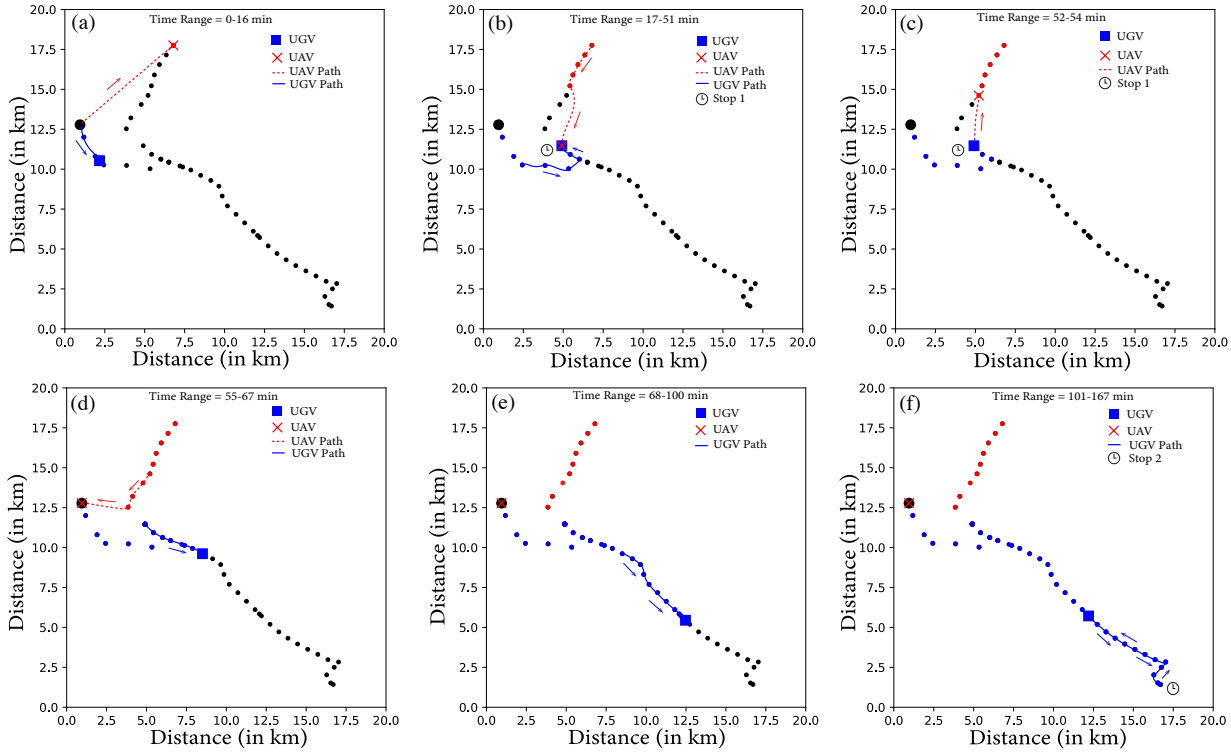

Fig. 6. Solution produced by Genetic Algorithm. The plots show the UAV and UGV route at different time spans.

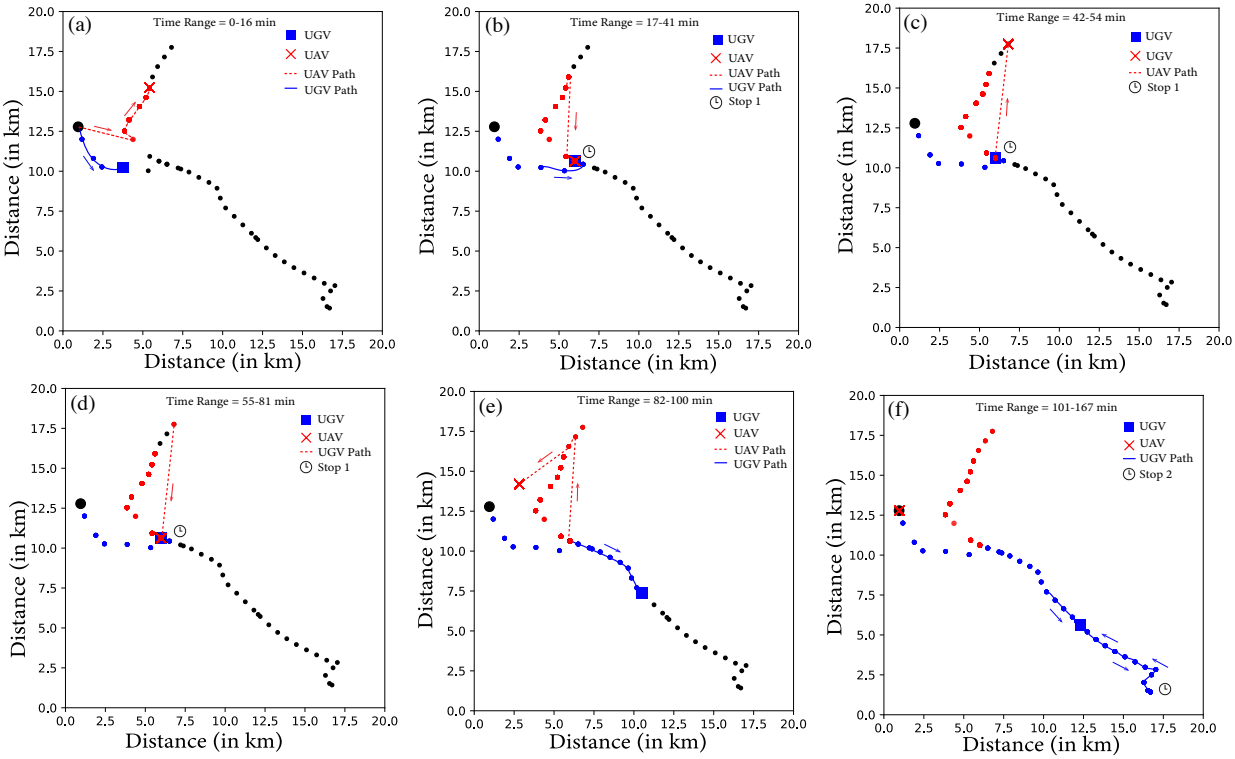

Fig. 7. Solution produced by Bayesian Optimization. The plots show the UAV and UGV route at different time spans.

the individual UAV solutions were quite different. The UAV for the BO used more energy and travelled for more time. However, since the total time was dependent on the UGV travel time (as it was the slower vehicle) and the UGV travel route were similar, the overall cost was the same. If the UAV time, energy, and other metrics are important then they can

be added to the cost function using a weighted sum.

Bayesian optimization was more efficient than Genetic algorithm in the number of function evaluations and computational time. BO is designed for computationally expensive function evaluations hence it is ideal in such cases where it takes substantial time to compute a solution by doing a local search for the UAV route.

One of the main limitations of the work is that UGV heuristics are limited to only 6 parameters which restricts the solution space. Adding more parameters will potentially make the search space too large and computationally restrictive. Another limitation is that the parameters and their range for the UGV heuristic optimization had to be chosen by trial and error.

5. CONCLUSIONS AND FUTURE WORK

We conclude that a tiered optimization is a feasible method to solve heterogenous vehicles optimization where the solution of one vehicle needs to be performed before the other one. In our case, the UGV route needs to be determined first as the UAV route involves refueling on the UGV. Moreover, by suitably parameterizing the heuristics and optimizing the parameters with global optimization methods enables exploration of the space and provide good quality solutions.

Our future work would explore both these optimization methods with different routes, different costs, and different heuristics in order to make broader claims about validity of the proposed approach. And moreover, since this research work addresses a problem-solving based approach where the authors focus on solving the problem that is given in hand, the algorithm that was developed focuses on solving problems with scenario maps that are similar to the scenario used in this work rather than focusing on comparing this algorithm to benchmark instances in the literature. Future works will deal with testing this algorithm on various scenarios in order to ensure the robustness of the algorithm.

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] Roberto Baldacci, Maria Battarra, and Daniele Vigo. Routing a heterogeneous fleet of vehicles. In *The vehicle routing problem: latest advances and new challenges*, pages 3–27. Springer, 2008.

[3] 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.

[4] Khaled A Ghamry, Mohamed A Kamel, and Youmin Zhang. Cooperative forest monitoring and fire detection using a team of uavs-ugvs. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1206–1211. IEEE, 2016.

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

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

[7] Ben Grocholsky, James Keller, Vijay Kumar, and George Pappas. Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13(3):16–25, 2006.

[8] Gurobi. Gurobi Optimization LLC. `https://www.gurobi.com/`, 2021. Online; accessed May 5, 2022.

[9] Jordan Henrio, Theo Deligne, Tomoharu Nakashima, and Tatsuhisa Watanabe. Route planning for multiple surveillance autonomous drones using a discrete firefly algorithm and a bayesian optimization method. *Artificial Life and Robotics*, 24(1):100–105, 2019.

[10] Changwu Huang, Bo Yuan, Yuanxiang Li, and Xin Yao. Automatic parameter tuning using bayesian optimization method. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2090–2097. IEEE, 2019.

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

[12] 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.

[13] Neil Mathew, Stephen L Smith, and Steven L Waslander. Multirobot rendezvous planning for recharging in persistent tasks. *IEEE Transactions on Robotics*, 31(1):128–142, 2015.

[14] Michael D McKay, Richard J Beckman, and William J Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[15] C. E. Miller, A. Tucker, and R. A. Zemlin. Integer programming formulation of traveling salesman problems. *J. ACM*, 7:326–329, 1960.

[16] Marcin L Pilat and Tony White. Using genetic algorithms to optimize acs-tsp. In *International workshop on ant algorithms*, pages 282–287. Springer, 2002.

[17] Subramanian Ramasamy, Jean-Paul F Reddinger, James M Dotterweich, Marshal A Childers, and Pranav A Bhounsule. 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)*, pages 155–164. IEEE, 2021.

[18] Shiwen Ren, Yang Chen, Ling Xiong, Zhihuan Chen, and Mengqing Chen. Path planning for the marsupial double-uavs system in air-ground collaborative application. In *2018 37th Chinese Control Conference (CCC)*, pages 5420–5425. IEEE, 2018.

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

[20] 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.

[21] Anupriya Shukla, Hari Mohan Pandey, and Deepti Mehrotra. Comparative review of selection techniques in genetic algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, pages 515–519. IEEE, 2015.

[22] Artem Sokolov and Darrell Whitley. Unbiased tournament selection. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 1131–1138, 2005.

[23] 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.

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

[25] 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.

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

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

[28] QJ Wang. Using genetic algorithms to optimise model parameters. *Environmental Modelling & Software*, 12(1):27–34, 1997.

[29] Yu Wu, Shaobo Wu, and Xinting Hu. Cooperative path planning of uavs & ugvs for a persistent surveillance task in urban environments. *IEEE Internet of Things Journal*, 8(6):4906–4919, 2020.