Route Planning for Cooperative Air-Ground Robots with Fuel Constraints: An Approach based on CMSA

Divansh Arora
IIIT-Delhi
Delhi, India
divansh15027@iiitd.ac.in

Pedro Pinacho-Davidson
Department of Computer Science,
Universidad de Concepción
Concepción, Chile
ppinacho@udec.cl

Parikshit Maini
IIIT-Delhi
Delhi, India
parikshitm@iiitd.ac.in

Christian Blum
Artificial Intelligence Research Institute (IIIA-CSIC)
Bellaterra, Spain
christian.blum@iiia.csic.es

ABSTRACT

Limited payload capacity on small unmanned aerial vehicles (UAVs) results in restricted flight time. In order to increase the operational range of UAVs, recent research has focused on the use of mobile ground charging stations. The cooperative route planning for both aerial and ground vehicles (GVs) is strongly coupled due to fuel constraints of the UAV, terrain constraints of the GV and the speed differential of the two vehicles. This problem is, in general, an NP-hard combinatorial optimization problem. Existing polynomial-time solution approaches make a trade-off in solution quality for large-scale scenarios and generate solutions with large relative gaps (up to 50 %) from known lower bounds. In this work, we employ a hybrid metaheuristic known as Construct, Merge, Solve & Adapt (CMSA) in order to develop a scalable and computationally efficient solution approach. We discuss results for large scale scenarios and provide a comparative analysis with the current state-of-the-art.

CCS CONCEPTS

• **Theory of computation** → **Discrete optimization**;

KEYWORDS

multi-robot planning; fuel constraint; energy constraint; metaheuristic; cooperative route planning; adaptive model reduction

ACM Reference Format:

Divansh Arora, Parikshit Maini, Pedro Pinacho-Davidson, and Christian Blum. 2019. Route Planning for Cooperative Air-Ground Robots with Fuel Constraints: An Approach based on CMSA. In *Genetic and Evolutionary Computation Conference (GECCO '19)*, July 13–17, 2019, Prague, Czech Republic. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3321707.3321820>

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6111-8/19/07...\$15.00

<https://doi.org/10.1145/3321707.3321820>

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) have gained popularity over the last decade and are used extensively in surveillance [2, 17, 20], for monitoring [21, 23], for mapping [22, 28], in agricultural settings [29] and for remote sensing [3] applications. The popularity of UAVs is due, in part, to the availability of small low-cost drones that do not have special transportation or operational requirements and are easy to deploy. However, a major limitation is payload capacity. UAVs can carry only a limited amount of batteries (fuel) on board. This restricts the flight time. Hence, for extensive missions, both concerning space and time, a UAV needs to recharge, possibly multiple times, to complete the mission. In this work, we address the problem of coupled route planning for aerial surveillance over a large geographical region for a fuel constrained UAV and a ground vehicle (GV), that is used to recharge the UAV. On account of practical considerations we restrict the ground vehicle to traverse on the available road network within the environment. The UAV needs to rendezvous with the GV on the ground to recharge. The two vehicles do not possess any communication capabilities, and passive autonomous landing techniques [5, 6] may be employed to land the UAV on the ground recharging vehicle. The recharging process may be automated using a battery swap system [27], or performed manually by an operator. Hence, the GV must reach the rendezvous location before the UAV runs out of fuel. This requires joint route planning while also accounting for the speed differential of the two vehicles.

Cooperative air-ground vehicle planning is an active area of research. Specifically in the context of fuel constrained planning and mobile recharging stations, there has been a lot of work in recent times. Mathew et al. [19] addressed a simplified version of the problem with fixed routes for the aerial vehicles and focused on route planning for ground vehicles to schedule suitable recharging windows. Maini et al. [14, 15] and Yu et al. [30] have addressed the cooperative routing problem of a UAV and mobile charging stations on the ground. It is worth mentioning that if the fuel limitation is relaxed, the problem reduces to the Euclidean Traveling Salesman Problem (E-TSP) which is known to be NP-hard. As a generalization of the E-TSP, the Cooperative Air-ground Vehicle Route Planning Problem with Fuel Limitations (CRPP-FL) is also NP-hard (see [15, 30]). Yu et al. [30] developed an intelligent discretization

Figure 1: A feasible solution to the cooperative air-ground vehicle route planning problem with fuel limitations. The GV, which is constrained to traverse on the road network, must recharge the UAV along its route, to ensure complete area coverage.

of the UAV battery levels that allows them to reduce the problem to an instance of the Generalized Traveling Salesman Problem (GTSP). By using current state-of-the-art techniques for the GTSP, they compute (near-)optimal solutions to large (discretized) problem instances. They assume equal speeds for both UAV and GV, and do not impose terrain restrictions on the recharging station. Maini et al. [15] have devised Mixed Integer Linear Programming (MILP) formulations to compute optimal solutions to the CRPP-FL. They have also developed construction heuristics [14, 15] to compute feasible solutions to large instances of the problem in a computationally efficient way with a trade-off in solution quality.

In this work, we employ a recent hybrid metaheuristic known as Construct, Merge, Solve & Adapt (CMSA) [1] for solving the CRPP-FL. In particular, we aim to develop an approach that scales well for large-scale problem instances to which exact solvers can not be applied. The main idea of CMSA is to make use of exact solvers in the context of reduced problem instances in an iterative way. CMSA has some resemblance with evolutionary algorithms. At each iteration, a population of feasible solutions to the tackled problem is generated in a probabilistic way. These solutions are then merged into an existing reduced problem instance, which is then solved by an exact solver. This last step can be seen as an *optimal recombination operator* in genetic algorithms [4]. Finally, the last step of each iteration consists in removing seemingly useless components from the current sub-instance. We make use of the MILP formulation and the TSP-based construction heuristic from [15] within different algorithmic components of our CMSA approach.

1.1 Related Problems

Fuel constrained route planning for UAVs in the presence of stationary depots is a closely related problem and has attracted a lot of research effort in recent times. MILP based exact methods for the problem have been developed in [8, 24–26]. An approximation algorithm for an asymmetric cost version of the problem was developed in [24]. Heuristic and metaheuristic algorithms based on local search have also been used to solve the fuel constrained routing problem [11, 24]. Fuel constrained routing for ground vehicles

has also garnered interest. Khuller et al. [9] developed exact and approximation algorithms for a variety of ground vehicle routing applications. Further, cooperative aerial-ground vehicle routing has been studied in the presence of communication constraints [18] and for applications such as data collection [3, 20, 29], search & rescue [7, 10], visual monitoring [13, 16] and mapping [28]. However, the strong coupling of the UAV and GV routes in our problem, which are due to fuel limitations, makes it very challenging to tackle the problem. In general, the application of metaheuristics has not been explored adequately in the context of fuel-constrained routing problems.

2 PROBLEM STATEMENT

The cooperative air-ground vehicle route planning problem with fuel limitations (CRPP-FL) is defined as follows:

Given (1) an environment ϵ with an existing road network, (2) a UAV with a given maximum flight length U , and (3) a GV that is constrained to travel on the road-network and can travel a maximum distance of R between two recharging operations of the UAV, the objective is to plan routes for the UAV and the GV to complete an aerial coverage mission of ϵ such that the UAV does not run out of fuel and the total distance traveled by the UAV is minimized.

The UAV model assumed in this work is a vertical take-off and landing (VTOL) type multirotor UAV with a down-facing constant focal length camera. It travels at a constant altitude and at constant speed. The altitude may be determined based on required image resolution, camera sensor and legal regulations. The amount of fuel consumption for the UAV is proportional to the distance traveled and is independent of the flight maneuver. Assuming the proportionality constant to be unity, we use the terms cost, distance traveled, time taken and fuel consumed interchangeably to refer to the same quantity. Hence U is the maximum fuel capacity, the maximum distance traveled on a single flight and the maximum flight time for a single charge. This is a simplifying assumption for the sake of ease of exposition and does not have a bearing on the solution quality. We impose a maximum speed constraint on the GV allowing it to travel a maximum of R road distance between two recharging operations of the UAV. The UAV may hover at the rendezvous location while it still has fuel if the GV has not yet arrived. We do not consider fuel limitations for the GV, as it typically has a much larger payload capacity and can carry extra batteries (fuel).

We discretize the environment, ϵ , in a grid based on the camera footprint at the UAV flight altitude. The grid centers comprise the set of data points, T , that the UAV must visit to complete the aerial coverage mission. The discretization ensures that if the UAV visits each of the data points and takes photographs using the on-board camera, the environment, ϵ , is covered.

3 PRELIMINARIES

The CMSA algorithm performs a global search by adaptively managing a reduced sub-instance (model) in order to find high-quality solutions. A sub-instance consists of a set of solution components. At each iteration, the incumbent sub-instance is solved, using a

limited amount of computation time, by an exact solution method. The aim of the CMSA algorithm is to assemble a sub-instance that contains high-quality solutions to the original (complete) problem. The sub-instance is updated at each iteration by generating feasible solutions to the problem in a probabilistic way. The solution components that form part of these solutions are added to the incumbent sub-instance. Moreover, after solving the current sub-instance, solution components that apparently do not contribute to good solutions are removed from the sub-instance. In this work we use the heuristic methods and the MILP model developed in [15] in order to deploy a CMSA algorithm. For the sake of completeness, in the following we briefly describe the MILP formulation and the heuristic described in [15].

3.1 Site Selection

The solution approach in [15] works in two phases. In the first phase it tests the feasibility of an instance, in terms of the reachability of all data points from the road network. In case of a feasible instance, it returns a set of points on the road network that are used as a superset of the rendezvous locations (also called refueling sites) used in the MILP formulation. This superset, referred to as the set of candidate refueling sites, satisfies two conditions: 1) *Reachability*: All data points in ϵ are reachable from at least one of the candidate refueling sites, i.e. they are at most $U/2$ distance away thus allowing the UAV to visit the data point and return to a refueling site without running out of fuel; 2) *Connectedness*: The graph whose vertices are formed by the set of candidate refueling sites and in which edges exist between vertices that are maximally R road distance away from each other consists of a single connected component. This implies that the UAV can traverse between any pair of candidate refueling sites without running out of fuel. The site selection phase employs a greedy algorithm that discretizes the road network and iteratively adds a new site to the set of candidate refueling sites, S , that (a) lies within R road distance of at least one site in the set S and (b) covers a maximum number of uncovered data points. The set generated in this way satisfies the terrain constraints of the GV. The second phase involves route planning for the two vehicles to cover all data points and ensure that the UAV never runs out of fuel. Set S ensures that there always exists a feasible solution to the routing problem in the second phase. The authors of [15] developed two solution methods to solve the routing problem in the second phase: a branch-and-cut method based on a MILP formulation and a TSP-based construction heuristic.

3.2 MILP formulation

The MILP formulation for the cooperative air-ground vehicle route planning problem with fuel limitations is formulated on the basis of the complete directed graph $G = (V, E)$, with vertex set $V = T \cup S$ and edge set E . UAV and GV start at a common refueling site $S_0 \in S$. We define two weight functions: $f : E \rightarrow \mathbb{R}^+$, where $f(e = (i, j))$ (or simply f_{ij}) denotes the fuel consumed by UAV when it travels along the directed edge $e = (i, j)$, and $r : (S \times S) \rightarrow \mathbb{R}^+$, that represents the road distance between two refueling sites. Let $N : S \rightarrow \wp(S)$, where $\wp(S)$ is the power set of S , denote a neighborhood function defined as $N(s_i) := \{s_j | r_{ij} \leq R, s_j \in S\}$.

The following decision variables are used to formulate the CRPP-FL. Variables x_{ij} , defined for each edge $(i, j) \in E$, are binary variables that indicate if, or not, the UAV traverses edge (i, j) . For every data point $t_i \in T$ there exists a variable u_i that represents the amount of fuel left in the UAV when it reaches t_i . Variables y_{ij} are binary decision variables, defined for every pair $\langle t_i, s_j \rangle$: $t_i \in T$ and $s_j \in S$. Such a variable takes value 1, if s_j is the last visited refueling site when the UAV is at data point t_i . Also, for any subset of vertices $P \subseteq V$, define $\delta^+(P) := \{(i, j) \in E | i \in P, j \notin P\}$. Finally, the objective function minimizes the fuel consumed by the UAV during the mission. The complete MILP is provided in the following.

Objective:

$$\mathcal{F}_1 : \min \sum_{i \in V} \sum_{j \in V} f_{ij} x_{ij}. \quad (1)$$

Degree constraints:

$$\sum_{i \in V \setminus \{j\}} x_{ij} = \sum_{i \in V \setminus \{j\}} x_{ji}, \quad \forall j \in V \text{ and} \quad (2)$$

$$\sum_{i \in V \setminus \{t\}} x_{it} = 1, \quad \forall t \in T. \quad (3)$$

Sub-tour elimination constraints:

$$\sum_{(i,j) \in \delta^+(P)} x_{ij} \geq 1, \quad \forall P \subseteq V \setminus \{s_0\}, P \cap T \neq \emptyset. \quad (4)$$

Fuel Constraints:

$$u_t - u_j + f_{jt} \leq M(1 - x_{jt}), \quad \forall t, j \in T, \quad (5)$$

$$u_t - u_j + f_{jt} \geq -M(1 - x_{jt}), \quad \forall t, j \in T, \quad (6)$$

$$u_t - U + f_{kt} \leq M(1 - x_{kt}), \quad \forall t \in T, \forall k \in S, \quad (7)$$

$$u_t - U + f_{kt} \geq -M(1 - x_{kt}), \quad \forall t \in T, \forall k \in S, \quad (8)$$

$$-u_t + f_{tk} \leq M(1 - x_{tk}), \quad \forall t \in T, \forall k \in S, \quad (9)$$

$$f_{ij} \cdot x_{ij} \leq U \quad \forall i, j \in S, \quad (10)$$

$$\sum_{i \in V} \sum_{j \in V} f_{ij} x_{ij} \leq U \sum_{k \in S} \sum_{i \in V} x_{ki}. \quad (11)$$

Refueling site constraints:

$$y_{ts} - x_{st} \geq 0, \quad \forall t \in T, s \in S, \quad (12)$$

$$y_{t_2s} - y_{t_1s} \leq (1 - x_{t_1t_2}), \quad \forall s \in S, \forall t_1, t_2 \in T, \quad (13)$$

$$y_{t_2s} - y_{t_1s} \geq -(1 - x_{t_1t_2}), \quad \forall s \in S, \forall t_1, t_2 \in T, \quad (14)$$

$$\sum_{k \in S \setminus N(s)} x_{tk} \leq (1 - y_{ts}), \quad \forall t \in T, s \in S, \quad (15)$$

$$\sum_{k \in S \setminus N(s)} x_{sk} = 0, \quad \forall s \in S, \quad (16)$$

$$\sum_{s \in S} y_{ts} = 1, \quad \forall t \in T. \quad (17)$$

Variable restrictions:

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j) \in E, \text{ either } i \text{ or } j \in T \quad (18)$$

$$u_i \in [0, U], \quad \forall i \in T, \quad (19)$$

$$y_{ts} \in \{0, 1\}, \quad \forall t \in T, s \in S. \quad (20)$$

The degree constraints (Eqs. (2)) ensure that each data point is visited exactly once by the UAV, permitting the UAV to make multiple visits to the refueling sites. Eqs. (4) are the sub-tour elimination

constraints. Note that the number of such constraints in the formulation is exponential. Hence, in the branch-and-cut framework from [15] these constraints are not included initially in the formulation. They are rather enforced using a separation routine. The fuel constraints (Eqs. (5)–(11)) ensure that the UAV does not run out of fuel during its trip. $M := U + \max_{(i,j)} f_{ij}$ is a large constant. The road-distance constraints (Eqs. (12)–(17)) limit the road distance between consecutive refueling sites to be at most R .

In feasible solutions to this MILP, consecutive visits to refueling sites on the UAV tour are within a road distance of maximally R units. The refueling site visits considered in the sequence as they occur on the UAV tour imply a valid tour for the GV and ensure feasibility of the UAV tour. Each feasible solution ensures that the GV reaches the refueling site before the UAV runs out of fuel.

3.3 Construction Heuristic

The construction heuristic proposed in [15] generates a feasible solution to the CRPP-FL as follows. The heuristic starts by applying the Lin-Kernighan heuristic (LKH) for TSP tour generation in order to obtain a sequence of all data points, starting with the initial refueling site. In order to fix fuel violations along this route, a *repairing algorithm* is used that inserts UAV visits to refueling sites for recharging the UAV. The recharging visits respect the R road distance constraint of the GV and guarantee a feasible solution. It is important to note that, given any valid TSP tour of the data points, this repairing mechanism generates a feasible solution to the CRPP-FL. We make use of this in the context of our adaptation of the heuristic to generate multiple solutions to the CRPP-FL in a probabilistic way (see Section 4.1).

4 THE CMSA APPROACH

To address the CRPP-FL, a joint planning framework needs to be developed to plan routes for both the UAV and the GV. We develop a hybrid metaheuristic approach following the CMSA [1] framework in order to generate high-quality solutions for the CRPP-FL. CMSA is an iterative metaheuristic for combinatorial optimization problems. At each iteration, first, a population of feasible solutions are generated in a probabilistic way. The components—that is, the variable-value combinations—that are found in these solutions are added to a set of *good components*, also called the sub-instance. Second, a reduction of the original MILP model is obtained by limiting the choice to the components found in the incumbent sub-instance. The algorithm then uses an exact method to compute—if possible within the allowed computation time—the best solution to the reduced model. The confidence level in good components is revised at the end of each iteration, and those that fall below a threshold are dropped from the sub-instance. Over multiple iterations, the algorithm is able to build high confidence values for components that appear repeatedly in the solutions for the reduced models. In summary, the CMSA metaheuristic comprises four stages. Each of those four stages of CMSA—namely, Construct, Merge, Solve and Adapt—is executed at each iteration of CMSA. We describe each stage of our implementation of CMSA to solve the CRPP-FL (Algorithm 1) in the following text.

Algorithm 1 Construct, Merge, Solve & Adapt (CMSA)

Require: problem instance I , parameter values N_a and Age_{max}

```

1:  $Q_{bsf} = \text{NULL}$ ,  $C = \emptyset$  ▷  $C$  is the sub-instance
2:  $\text{age}[c] = 0$ ,  $\forall c \in C$  ▷  $C$  is the complete set of components
3:  $S = \text{SiteSelection}(I)$  ▷ See Sec. 3.1
4: while CPU time limit not reached do
5:   for  $i = 1, \dots, N_a$  do
6:      $Q = \text{ProbabilisticTourGeneration}(T \cup S_0)$  ▷ See Alg. 2
7:      $Q = \text{TourRepairAlgorithm}(Q, S)$ 
8:     for all  $c \in Q$  and  $c \notin C$  do
9:        $\text{age}[c] = 0$ 
10:       $C = C \cup \{c\}$ 
11:       $Q_{opt} = \text{ApplyExactSolver}(C)$ 
12:      if ( $Q_{opt}$  is better than  $Q_{bsf}$ ) then  $Q_{bsf} = Q_{opt}$ 
13:       $\text{Adapt}(C, Q_{opt}, Age_{max})$  ▷ See Alg. 3
return  $Q_{bsf}$ 

```

4.1 Probabilistic Solution Generation

The *construct* stage of CMSA involves generating multiple feasible solutions at each iteration. As already mentioned in Section 3.3 above, the probabilistic construction of solutions is based on the heuristic from [15]. First, a probabilistic TSP tour generation procedure (see Algorithm 2) is employed. In particular, we implemented a probabilistic version of the nearest neighbor heuristic which takes as input the set of data points, T , and the initial refueling site S_0 . The tour construction starts with S_0 . At each step, exactly one data point from set A , the set of data points not visited so far, is added to the current partial tour. At the start of each solution construction, A is initialized with all data points from T . Given a partial tour, the current data point *curr* and the corresponding set A , the next data point to be added to the partial tour, *next*, is determined as follows. First, A' is defined as the subset of A containing all data points from A whose distance to *curr* is at most U units. Then, a random number δ from $[0, 1]$ is chosen uniformly at random. In case $\delta < d_{rate}$, *next* is chosen as the data point from A' that is closest to *curr* with respect to the UAV cost matrix *cost*. Otherwise, *next* is chosen uniformly at random from the set of $\min\{|A'|, l_{size}\}$ data points from A' that are closest to *curr*. In other words, the values of parameters l_{size} and d_{rate} determine the amount of determinism in this procedure. As the value of d_{rate} tends to 1, the procedure shifts towards a deterministic version and follows the nearest neighbor paradigm. If the value of d_{rate} is small, the randomness increases.

Each tour is then passed to the tour repairing algorithm, which is a deterministic algorithm, to generate a feasible solution to the CRPP-FL. Note that—due to the stochasticity of the tour construction process—this solution construction process is stochastic. The level of stochasticity within the procedure is user-determined and must be tuned in order to obtain a high-performing algorithm. We discuss parameter tuning in detail in Section 5.

4.2 Adding Components to the Sub-Instance

The next step of CMSA consists in extracting the solution components that form part of the constructed solutions and in adding them to the sub-instance C . This is shown in lines 8–10 of Algorithm 1. We define solution components in the context of the CRPP-FL in

Algorithm 2 Probabilistic Tour Generation

Require: $A := T \cup S_0$, l_{size} , d_{rate} , U and UAV cost matrix

```

1: Let  $\varrho$  be an empty array
2: Append  $S_0$  to  $\varrho$ 
3:  $A := A \setminus \{S_0\}$ 
4:  $curr := S_0$ 
5: while  $A \neq \emptyset$  do
6:    $A' := \{i \in A \mid cost(curr, i) < U\}$ 
7:    $\delta := \text{rand}(0,1)$ 
8:   if  $(\delta < d_{rate})$  then
9:      $next := \text{argmin}\{cost(curr, i) \mid i \in A'\}$ 
10:  else
11:    Let  $L$  be the min $\{|A'|, l_{size}\}$  data points closest to  $curr$ 
12:    Choose  $next$  uniformly at random from  $L$ 
13:  Append  $next$  to  $\varrho$ 
14:   $A := A \setminus \{next\}$ 
return  $\varrho$ 

```

---

terms of point to point UAV traversals. A component represents whether or not the UAV traverses the path between two vertices (data points and refueling sites). More specifically, solution components are in the form of triplets $\langle i, j, val \rangle$, where $i, j \in T \cup S$ and val corresponds to value 0 or 1. For each path (i, j) we define two components, $\langle i, j, 0 \rangle$ and $\langle i, j, 1 \rangle$. The two components correspond to the absence or existence of the edge in a feasible solution. Only one of them can exist in a solution. In the context of the MILP formulation summarized in Section 3.2, the solution components correspond to the variable-value combinations concerning the x_{ij} variables. The binary decision variables assume value 1 if the UAV traverses edge (i, j) in the solution path and 0, otherwise. It is worth mentioning that any feasible solution ϱ to CRPP-FL may be viewed as a subset of the set of all components, C , subject to feasibility conditions of the MILP.

The *merge* stage of CMSA involves the extraction of those components that occur in the generated feasible solutions. Extracted components that do not already exist in C are then added to C . The rationale is that components that are common to all randomly generated feasible solutions are likely to form part of high-quality feasible solutions to the CRPP-FL.

4.3 Model Reduction and Exact Solution

The reduced model is generated on the basis of the original MILP model and the sub-instance C . More specifically, if both components corresponding to a variable x_{ij} are found in C —that is, both the component corresponding to $val = 0$ and the component corresponding to $val = 1$ —the corresponding variable x_{ij} will be left free for optimization in the reduced model. Otherwise, the variables' value is fixed to the value of the corresponding component. In the latter case, a model reduction—in comparison to the original MILP model—is achieved. The reduced model is then given as input to the *solve* stage (see line 11 of Algorithm 1). In particular, an exact solution method is used to find the solution to the reduced MILP model. In the case of the CRPP-FL, the exact method corresponds to the branch-and-cut algorithm mentioned in Section 3.2, which

Algorithm 3 Adapt

Require: Age_{max} , ϱ_{opt} and C

```

1: for all  $c \in C$  do
2:    $age[c] = age[c] + 1$ 
3:   if  $c \in \varrho_{opt}$  then
4:      $age[c] = 0$ 
5:   if  $age[c] \geq Age_{max}$  then
6:      $C = C \setminus \{c\}$ 
return

```

---

is subject to a computation time limit. The optimal (best) solution, ϱ_{opt} , generated within the given time is returned.

4.4 Model Adaptation

Finally, the *adapt* stage of the CMSA algorithm is the last step of each iteration (see line 13 of Algorithm 1). It takes as input the sub-instance C , the optimal (best) solution, ϱ_{opt} , returned by the exact method, and the maximal age limit Age_{max} . First, the age of all solution components that occur in the solution of the exact solver is reduced to zero. For all other components in set C , the age is increased by one. However, in case the age of a component reaches the predefined threshold Age_{max} , the component is removed from the sub-instance C . This procedure is pseudo-coded in Algorithm 3. Over multiple iterations of CMSA, the sub-instance C is adapted and comprises a set of good components from within the domain of the CRPP-FL. Once a component is in C , its age may only be modified in the *adapt* stage. Thus, components that do not occur in the solution generated by the exact solver over a number of iterations specified by Age_{max} are not considered as good components and are removed from C .

5 SIMULATION RESULTS

The CMSA approach to solve the CRPP-FL was evaluated by means of simulations to verify its performance. The algorithm was implemented in C++ using the STL library, IBM ILOG CPLEX 12.7 and the LEMON graph library. A matlab based simulation platform was also built for visualization. The simulations were carried out on a High Performance Computing Cluster node with 20 cores. We compare our results with 1) the application of the branch-and-cut procedure to the original/complete MILP formulations [15], 2) the TSP-based Heuristic [15] and 3) the Greedy Heuristic from [14]. Each of the algorithms were implemented as described in the respective papers.

For the sake of comparison we use the same instances as used in [15] for evaluation. Two different road networks were used and the scale of the instances was varied from 16 to 100 data points by varying the scale parameter, n , from 4 to 10. The number of data points is computed as $n \times n$. 240 instances were generated for each scale parameter value, while varying U and R from 15 to 25 and 10 to 15, respectively (both in steps of 5). Thus a total of 1680 instances were used in the simulations. A small number (264) of instances in one of the road networks for small values of U were found infeasible. This is consistent with the results reported in [15].

Table 1: Tuning results

	d_{rate}	Age_{max}	N_a	l_{size}
small-scale instances	0.882	1	40	39
large-scale instances	0.836	4	1	32

5.1 Parameter Tuning

In order to find well-working parameter values for our CMSA algorithm, we made use of a scientific tuning procedure. The parameters to be tuned are (1) the size of the population used in the *construct* stage (N_a); (2) the size of the candidate list L for the solution construction process (l_{size}); (3) the determinism rate parameter for the solution construction process (d_{rate}); and (4) the age threshold used to adapt the sub-instance during the *adapt* stage (Age_{max}). In particular, we made use of *irace* [12], which is a well-known tool for finding the most appropriate parameter settings given a set of instances of the problem.

Parameter tuning was conducted for small and large scale instances separately. For this purpose, instances are labelled *small scale* if they are obtained with a scale parameter value between 4 and 6, and *large scale* if they are obtained with a scale parameter value in the range 7 to 10. We used 12 training instances for each scale parameter value. Of the 12 instances, 6 each were picked uniformly at random from different combinations of U and R . The tuning process for each set was given a budget of 1000 CMSA experiments with 3600 seconds or 400 iterations (whatever got exhausted first) for each experiment. Within CMSA, the exact solver was allowed to run for maximally 200 seconds at each iteration. All timing information is in CPU time scale. Further, the range of values permitted by *irace* for each parameter was as follows:

- N_a : integral values from $\{1, \dots, 100\}$
- l_{size} : integral values from $\{1, \dots, 100\}$
- d_{rate} : real values from the range $[0, 1]$
- Age_{max} : integral values from $\{1, \dots, 100\}$

The results of the tuning process are summarized in Table 1. Two interesting observations from this table merit discussion. The value of Age_{max} for small scale instances is 1. This means that, after every iteration, solution components that do not occur in the solution provided by the exact solver are discarded from the sub-instance. This value has to be seen in combination with the large population size ($N_a = 40$), which makes sure that, at each iteration, enough *new material* is added to the sub-instance in order to be able to generate high-quality solutions. The opposite behavior is observed for large scale instances, for which the population size is very small ($N_a = 1$) and the age threshold is greater than 1 ($Age_{max} = 4$). We believe that these settings are chosen by *irace*, because the reduced models for large-scale instances quickly become very large, being due to the exponential growth of the scale and the resulting large number of components. With a constant processing time for the exact solver, computing a feasible solution for the models can also be a challenge in some cases. In this context, we plan to study the effect of tuning the exact solver time budget in the future and also to consider tuning separately for each scale parameter value.

5.2 Results

The parameter values as given in Table 1 (obtained by *irace*) were used for the simulations. For each instance CMSA was given a budget of 3600 seconds and 400 iterations. The execution stopped when the first one of the two budgets was exhausted. In addition, within each iteration of CMSA, the exact solver was allowed to run for 200 seconds. Thus a constant time of 3600 seconds (maximum) was allocated to each instance. The branch-and-cut framework for the MILP formulation proposed in [15] was able to solve instances of only up to 36 data points—that is, a scale parameter value of 6—within a time budget of 7200 seconds. It also generated lower bounds for these instances. We computed relative gaps for solutions generated by CMSA using these lower bounds. Figure 2 provides a comparison of the four different algorithmic approaches to solve the CRPP-FL. It shows results only for the small scale set of instances because the MILP does not scale well to large instances and was not able to generate any feasible solution or lower bounds for large-scale instances. The figure shows that CMSA is able to compute solutions (nearly) as good as the ones of the branch-and-cut framework over the spectrum of small-scale instances.

For the comparison with the two heuristics we show results for instances of both small and large scale. In the absence of any lower bounds, we show the results in terms of the objective function values. As may be seen in Figure 3, CMSA outperforms both heuristic algorithms consistently and with a small standard deviation. Even though the TSP-based heuristic and the greedy heuristic both complete execution for any instance within 60 seconds and are considerable faster than CMSA (3600 seconds), the results are significant because the CRPP-FL is an offline planning problem and can afford computation times of this order. CMSA provides a mechanism to compute good solutions within a constant time of 3600 seconds. Further, it is also worthwhile to note that CMSA does not require the full 3600 seconds to generate the best solution for all considered instances. In fact, for the small scale instances the best solutions of a run are generally obtained within a few seconds. However, we would like to do some more experiments and analysis to provide further bounds on the computation time for CMSA.

6 CONCLUSION AND FUTURE WORK

We have successfully shown the deployment of a hybrid meta-heuristic known as CMSA to solve a complex problem from the field of UAV cooperative path planning (CRPP-FL). CMSA is based on model reduction, which is achieved by making the exact solver focus on solution components that arise frequently in feasible solutions generated by a probabilistic approach. Our CMSA algorithm is shown to scale well for large instances of the problem. Moreover, it outperforms previous efforts on scalable solution techniques for the CRPP-FL. As noted previously, in future work we plan to explore the effects of including the time budget for the exact solver into the tuning process. Another aspect that we would like to look at involves removing the binary decision criteria on the inclusion of a component and instead consider components that occur in a large fraction of the generated solutions, rather than in all solutions. This work has also opened the possibility to explore the performance of CMSA and other global-search metaheuristics on complex and challenging problems like the CRPP-FL.

Figure 2: The graphics show comparative results concerning four algorithmic approaches: the MILP formulation [15], CMSA, the TSP-heuristic [15] and the Greedy heuristic [14]. The results are reported in terms of the relative gap of the objective function value with the lower bounds provided by the MILP solver. It is observed that CMSA consistently outperforms both the TSP-heuristic and the greedy approach over all sets of instances and produces results very close to those computed by the MILP solver. In (b) and (d), due to the existence of some infeasible instances, the number of data points was very small.

ACKNOWLEDGMENTS

Pedro Pinacho-Davidson acknowledges the support of the CONICYT PCI Program under grant REDES170160.

REFERENCES

- [1] Christian Blum, Pedro Pinacho, Manuel López-Ibáñez, and José A Lozano. 2016. Construct, merge, solve & adapt a new general algorithm for combinatorial optimization. *Computers & Operations Research* 68 (2016), 75–88.
- [2] David W Casbeer, Derek B Kingston, Randal W Beard, and Timothy W McLain. 2006. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Science* 37, 6 (2006), 351–360.
- [3] Doo-Hyun Cho, Dae-Sung Jang, and Han-Lim Choi. 2018. Heterogeneous, Multiple Depot Multi-UAV Path Planning for Remote Sensing Tasks. In *2018 AIAA Information Systems-AIAA Infotech@Aerospace*. <https://doi.org/10.2514/6.2018-0894>
- [4] Anton V. Eremeev. 2016. Optimal recombination in genetic algorithms for combinatorial optimization problems part II. *Yugoslav Journal of Operations Research* 24, 1 (2016), 165–186.
- [5] Alvika Gautam, Ashwini Ratnoo, and PB Sujit. 2018. Log Polynomial Velocity Profile for Vertical Landing. *Journal of Guidance, Control, and Dynamics* 41, 7 (2018), 1617–1623.
- [6] Alvika Gautam, PB Sujit, and Srikanth Saripalli. 2014. A survey of autonomous landing techniques for UAVs. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 1210–1218.
- [7] Samira Hayat, Evşen Yanmaz, Timothy X Brown, and Christian Bettstetter. 2017. Multi-objective UAV path planning for search and rescue. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5569–5574.
- [8] Tanya E Kannon, Sarah G Nurre, Brian J Lunday, and Raymond R Hill. 2015. The aircraft routing problem with refueling. *Optimization Letters* 9, 8 (2015), 1609–1624.
- [9] Samir Khuller, Azarakhsh Malekian, and Julián Mestre. 2011. To fill or not to fill: The gas station problem. *ACM Transactions on Algorithms* 7, 3 (2011), 36.
- [10] Heba Kurdi, Jonathon How, and Guillermo Bautista. 2016. Bio-inspired algorithm for task allocation in multi-UAV search and rescue missions. In *AIAA Guidance, Navigation, and Control Conference*. <https://doi.org/10.2514/6.2016-1377>
- [11] David Levy, Kaarthik Sundar, and Sivakumar Rathinam. 2014. Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. *Mathematical Problems in Engineering* Article ID 131450 (2014).
- [12] Manuel López-Ibáñez, Jérémie Dubois-Lacoste, Leslie Pérez Cáceres, Thomas Stützle, and Mauro Birattari. 2016. The irace package: Iterated Racing for Automatic Algorithm Configuration. *Operations Research Perspectives* 3 (2016), 43–58.
- [13] Parikshit Maini, Gautam Gupta, Pratap Tokekar, and PB Sujit. 2018. Visibility-based monitoring of a path using a heterogeneous robot team. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3765–3770.
- [14] Parikshit Maini and PB Sujit. 2015. On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 1370–1377.
- [15] Parikshit Maini, Kaarthik Sundar, Sivakumar Rathinam, and P. B. Sujit. 2018. Cooperative Planning for Fuel-constrained Aerial Vehicles and Ground-based

Figure 3: The graphics show comparative results concerning three algorithmic approaches: CMSA, the TSP-heuristic [15] and the Greedy heuristic [14]. In the absence of lower bounds for large-scale instances, the objective function values are reported. It is observed that CMSA consistently outperforms both the TSP-heuristic and the greedy approach over all sets of instances. In (b) and (d), due to the existence of some infeasible instances, the number of data points was very small.

- Refueling Vehicles for Large-Scale Coverage. *CoRR* abs/1805.04417 (2018). arXiv:1805.04417 <http://arxiv.org/abs/1805.04417>
- [16] Parikshit Maini, Kevin Yu, PB Sujit, and Pratap Tokekar. 2018. Persistent monitoring with refueling on a terrain using a team of aerial and ground robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 8493–8498.
- [17] Satyanarayana G Manyam, Steven Rasmussen, David W Casbeer, Krishnamoorthy Kalyanam, and Suresh Manickam. 2017. Multi-UAV routing for persistent intelligence surveillance & reconnaissance missions. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 573–580.
- [18] Satyanarayana G Manyam, Sivakumar Rathinam, Swaroop Darbha, David Casbeer, and Phil Chandler. 2014. Routing of two unmanned aerial vehicles with communication constraints. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 140–148.
- [19] Neil Mathew, Stephen L Smith, and Steven L Waslander. 2015. Multirobot rendezvous planning for recharging in persistent tasks. *IEEE Transactions on Robotics* 31, 1 (2015), 128–142.
- [20] ABHA Moitra, Robert M Mattheyses, VA DiDomizio, LJ Hoebel, RJ Szczerba, and B Yamrom. 2003. Multivehicle reconnaissance route and sensor planning. *IEEE Trans. Aerospace Electron. Systems* 39, 3 (2003), 799–812.
- [21] Sivakumar Rathinam, Zu Whan Kim, and Raja Sengupta. 2008. Vision-Based Monitoring of Locally Linear Structures Using an Unmanned Aerial Vehicle. *Journal of Infrastructure Systems* 14, 1 (2008), 52–63.
- [22] PB Sujit, BP Hudzietz, and S Saripalli. 2013. Route Planning for Angle Constrained Terrain Mapping Using an Unmanned Aerial Vehicle. *Journal of Intelligent & Robotic Systems* 69, 1 (01 Jan 2013), 273–283.
- [23] PB Sujit, Derek Kingston, and Randy Beard. 2007. Cooperative forest fire monitoring using multiple UAVs. In *2007 46th IEEE Conference on Decision and Control*. IEEE, 4875–4880.
- [24] Kaarthik Sundar and Sivakumar Rathinam. 2014. Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. *IEEE Transactions on Automation Science and Engineering* 11, 1 (2014), 287–294.
- [25] Kaarthik Sundar and Sivakumar Rathinam. 2017. Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems. *Journal of Intelligent & Robotic Systems* 88, 2-4 (2017), 513–526.
- [26] Kaarthik Sundar, Saravanan Venkatachalam, and Sivakumar Rathinam. 2017. Analysis of mixed-integer linear programming formulations for a fuel-constrained multiple vehicle routing problem. *Unmanned Systems* 5, 04 (2017), 197–207.
- [27] Koji AO Suzuki, Paulo Kemper Filho, and James R Morrison. 2012. Automatic battery replacement system for UAVs: Analysis and design. *Journal of Intelligent & Robotic Systems* 65, 1-4 (2012), 563–586.
- [28] Pratap Tokekar and Vijay Kumar. 2015. Visibility-based persistent monitoring with robot teams. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 3387–3394.
- [29] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. 2016. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Transactions on Robotics* 32, 6 (2016), 1498–1511.
- [30] Kevin Yu, Ashish Kumar Budhiraja, and Pratap Tokekar. 2018. Algorithms for routing of unmanned aerial vehicles with mobile recharging stations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 1–5.