European Journal of Operational Research 300 (2022) 992–1004

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

A cumulative unmanned aerial vehicle routing problem approach for humanitarian coverage path planning

Nikolaos A. Kyriakakis, Magdalene Marinaki, Nikolaos Matsatsinis, Yannis Marinakis*

Technical University of Crete, School of Production Engineering and Management, University Campus, Chania 73100, Greece

ARTICLE INFO

Article history:
Received 2 November 2020
Accepted 7 September 2021
Available online 15 September 2021

Keywords:
Humanitarian coverage path planning
Unmanned aerial vehicle routing
Greedy randomized adaptive search procedure

ABSTRACT

This paper presents a Cumulative Unmanned Aerial Vehicle Routing Problem (CUAVRP) approach to optimize Humanitarian Coverage Path Planning (HCPP). Coverage path planning consists of finding the route which covers every point of a certain area of interest. This paper considers a Search & Rescue mission, using a homogeneous fleet of Unmanned Aerial Vehicles (UAVs). In this scenario, the objective is to minimize the sum of arrival times at all points of the area of interest, thus, completing the search with minimum latency. The HCPP problem is transformed into a Vehicle Routing Problem by using an approximate cellular decomposition technique to discretize the area into a grid, where the rectangles represent the UAV sensor's field of view. The center points of the formed rectangles, become the nodes used for a UAV routing problem. This approach uses the objective of minimizing the sum of arrival times at customers, found in the Cumulative Capacitated Vehicle Routing Problem (CCVRP), adjusted for the Search & Rescue Coverage Path Planning using UAVs. The Min-max objective is also implemented and tested. Three versions of a Parallel Weighted Greedy Randomized Adaptive Search Procedure - Variable Neighborhood Decent (GRASP-VND) algorithm is implemented to solve the Cumulative UAV Routing Problem for Humanitarian Coverage Path Planning.

1. Introduction

Unmanned Aerial Vehicles (UAVs) are increasingly used in a wide range of applications, such as surveillance (Babel, 2017; Basilico & Carpin, 2015), agriculture (Lottes, Khanna, Pfeifer, Siegwart, & Stachniss, 2017) and civil protection (Maza, Caballero, Capitan, Martinez-de Dios, & Ollero, 2011; Pham, La H., Feil-Seifer, & Deans, 2020). The Coverage Path Planning (CPP) problem is classified as a motion planning subtopic in robotics, where it is necessary to build a path for a robot to explore every location in a given scenario (Choset, 2001). The utilization of UAVs for CPP has seen various approaches in the literature, investigating different methods of area decomposition and performance metrics, such as energy consumption, path length, area overlapping and mission complete time (Cabreira, Brisolara, & Ferreira, 2019; Galceran & Carreras, 2013). Search & Rescue operations require exploring or covering an area of interest, thus are considered a Coverage Path Planning problem. The use of UAVs to perform Search & Rescue and other similar coverage operations such as area mapping or surveillance, has been studied with emphasis on control models (Goerzen, Kong, & Mettler, 2010; Rosalie et al., 2017), mission planning (Evers, Barros, Monsuur, & Wagelmans, 2014), sensoring systems (Rudol & Doherty, 2008; Santamaria, Segor, Tchouchenkov, & Schnbein, 2013) and coverage time (Nattero, Recchiuto, Sgorbissa, & Wanderlingh, 2014). Research on UAV routing problems and their applications has also been increasing in popularity, a great review of which can be found in Macrina, Di Puglia Pugliese, Guerriero, & Laporte (2020). UAVs have also been combined with ground vehicles in a two-echelon scheme for routing applications (Li, Chen, Wang, & Bai, 2021).

The Coverage Path Planning Problem (CPPP) with multiple UAVs, can be transformed into a Vehicle Routing Problem (VRP), by building a graph that coverage is reached when a set of nodes is visited at least once by a vehicle. The decisions to be made when formulating the CPPP to a VRP pertain to the process of constructing the graph, the way coverage is obtained, the constraints imposed by the vehicles and the objective function to be minimized (Avellar, Pereira, Pimenta, & Iscold, 2015).

In our approach, the humanitarian aspect of Coverage Path Planning is addressed, thus, it focuses on the Search & Rescue objective of locating the victims the soonest possible. A grid approximation method used in the CPP problem is utilized, to formulate the Coverage Path Planning problem as a Vehicle Routing Problem

* Corresponding author.
E-mail addresses: nkyriakakis@isc.tuc.gr (N.A. Kyriakakis), magda@dssl.tuc.gr (M. Marinaki), nikos@ergasya.tuc.gr (N. Matsatsinis), marinakis@ergasya.tuc.gr (Y. Marinakis).

https://doi.org/10.1016/j.ejor.2021.09.008
0377-2217/

and incorporates the objective of minimizing the sum of arrival times at all points in the area of interest. This objective is found in the Cumulative Capacited Vehicle Routing Problem (CCVRP).

The Cumulative Capacited Vehicle Routing Problem (CCVRP) was introduced by Ngueveu, Prins, & Wolfler Calvo (2010) and is NP-hard as it is an extension of the classical Vehicle Routing Problem (VRP). Its objective function is formulated in a way which minimizes the sum of arrival times at the customers instead of the total routing cost in the classical VRP. This variation provides solutions better suited for application in the humanitarian supply chain, such as in the case of a disaster relief, where the priority is to save lives (Campbell, Vandenbussche, & Hermann, 2008). The CCVRP was first introduced and solved by Ngueveu et al. (2010). In this approach UAVs are used instead of traditional ground vehicles, therefore, constrains and assumptions are changed to accommodate the characteristics of the aerial vehicles in a Search & Rescue scenario. We call this adaptation the Cumulative UAV Routing Problem (CUAVRP).

In order to effectively solve the CUAVRP and by extension the Coverage Path Planning problem using the suggested human-centric objective, three versions of a Parallel Weighted Greedy Randomized Adaptive Search Procedure - Variable Neighborhood Decent (GRASP-VND) algorithm are implemented. Each variant incorporates a different communication and information exchange strategy between threads. The performance of the algorithm is tested on well known benchmark instances for the CCVRP. A set of benchmark instances are generated for CUAVRP using VRP instances from the literature and are solved using the developed algorithms.

Section 2 defines the Search & Rescue scenario considered and its transformation to a Routing Problem. In Section 3 the mathematical formulation of the Cumulative UAV Problem is described. Section 4 describes the three Parallel GRASP-VND algorithms implemented to solve the CUAVRP and in Section 5 the computational results are presented. Finally, Section 6 discusses conclusions and suggestions for future research.

2. Search & rescue using UAVs

2.1. Humanitarian coverage path planning problem

For this paper we consider the follow Search & Rescue scenario:

A homogeneous fleet of R rotary wing UAVs, capable of vertical take-off and landing, must cover all points in a polygonal convex area A represented by a set P of vertices in R^2, with the objective to minimize the sum of arrival times at all points and by extension to the victims' potential locations.

The below assumptions are made:

- If the area is not convex, it is assumed that P represents the convex hull of the area.
- All UAVs are deployed from and returned to a predefined point inside the area A, called base.
- Each UAV is equipped with an on-board camera/sensor pointing down and has a square viewing aspect.
- The speed and altitude of the UAVs in flight is constant and is chosen so that the camera/sensor allows the observation of the characteristics of interest on the ground.
- The take-off and landing of the UAVs, are assumed to be vertical, and the time to reach the fixed altitude or land is negligible.
- It is considered that the maximum time of flight of each UAV is finite, known in advance.
- No external forces affecting the UAVs are considered, such as weather conditions (i.e wind).

For this implementation, the convex hull of the area is considered for generating of the cellular grid. Thus, the graph of P vertices is fully connected. Depending of the shape of the area of interest, this assumption might lead to additional area being explored. In most practical cases of humanitarian UAV searching (i.e searching over sea after a shipwreck, over a mountain after an avalanche), a convex area of interest (i.e. a rectangle, circle, square, triangle) is considered beforehand by the search crew, as it is an intuitive way to relay the information amongst the rescuers. The CUAVRP can be easily adapted to a non-convex variant, where a partially connected graph of nodes is used. Nonconvex areas are used not only to limit searching outside the area of interest, but also to accommodate no-fly zones inside the area.

Another assumption which offers additional variants for the CUAVRP, is the static wind field assumption. By not considering external forces, distances between nodes are symmetrical. Asymmetric distances can be easily applied, simulating the difference in time needed to travel between nodes, depending on the travelling direction. Combinations of the assumptions result in a several possible CUAVRP variants, convex or concave, with or without no-fly zones and symmetric or asymmetric distances.

2.2. Cumulative vehicle routing problem transformation

In order to solve the problem presented in the previous section with the humanitarian objective of minimizing latency in reaching all points in area A, the HCPP problem is transformed into a Cumulative Vehicle Routing Problem.

The area of interest is discretized into a set of regular cells of square form using a grid based method of approximate cellular decomposition (Choset, 2001; Galceran & Carreras, 2013). These cells render a grid on the area of interest. Since the UAVs fly at a certain altitude from the ground carrying a camera/ sensor to perform the task, the size $d \times d$ of the cells is defined by the viewing field of the camera on-board the UAV (Cabreira et al., 2019; Valente, Sanz, Cerro, Barrientos, & de Frutos, 2013). Therefore, when the UAV is at the center point of the cell, the camera has vision of the whole square area below it.

To create the grid with square cells of size $d \times d$, the first cell is placed at the base - the UAV launching and retrieval point. Let $x_{\min}, y_{\max}$ be the left-most and top-most coordinates of all points in P and $x_{\max}, y_{\min}$ the right-most and bottom-most coordinates, respectively. Cells are placed next to each other to form a row of cells along the x-axis covering the area of interest from the left-most $x_{\min}$ to the right-most $x_{\max}$ coordinate. Once the two outer x-coordinates are included in a cell's area the row is complete. Rows of the same length (same number of cells) are, then, added along the y-axis until the top-most $y_{\max}$ and bottom-most $y_{\min}$ coordinate is covered by a cell's area. From this procedure the end result is a rectangular grid of cells which completely covers the area of interest.

Each cell is defined by its center point (x_c, y_c) and its side length d. We can extract the (x, y) coordinates of the four corner points and, therefore, the line segments of the cell's sides. The center points of these cells, are the potential nodes of the CCVRP. Only center-points of cells that have overlapping area with the area of interest A are considered as nodes of the CCVRP transformation. In order to make this segregation and keep only the points required, every potential center-point is checked using the Winding Number algorithm published by Alciatore & Miranda (1995) to determine whether the point is inside the polygon (convex hull) represented by P. If the center-point is inside the polygon, then, its cell's area is overlapping with the area of interest and therefore it is considered as a CCVRP node. If the center-point is not inside the polygon, then, the cell's sides are checked whether they collide with any of the polygon's edges. If collision is detected, at least one point of the cell area is inside the polygon and the cell's center point is considered a CCVRP node.

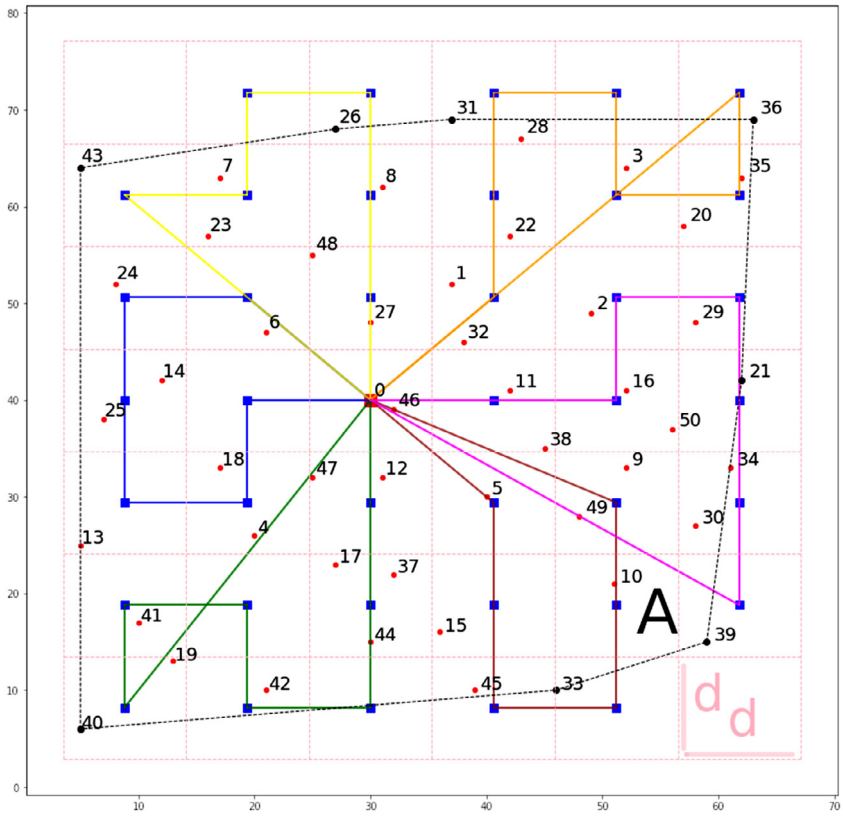

Fig. 1. CUAVRP solution after grid-based approximate cellular decomposition on CMT1 instance.

After completing this procedure, the set of nodes obtained represents the pathway-points the UAVs have to visit in order to completely cover the area of interest, in the Search & Rescue scenario defined in Section 2.1. Fig. 1 illustrates an example of nodes obtained using the suggested grid-based area approximation. The numbered points are the nodes of the original CMT1 VRP instance (Christofides, Mingozzi, & Toth, 1979b), A is the area of interest as the convex hull of the VRP points and the blue squares are the nodes of the CUAVRP after the approximate cellular decomposition of that area. As mentioned in the assumptions, flight velocity is constant at a fixed value for all UAVs in the fleet and, therefore, the travel time between nodes can be calculated as *distance/velocity*. For the sake of simplicity, the euclidean distance is considered as the travel time between nodes in this paper.

2.3. Differences compared to other CPPP approaches and the CCVRP

In the classical motion planning approaches for the CPPP, the two most used trajectories in the literature is the *spiral motion* and the *back-and-forth*. When more than one agents (vehicles) are used, the area is partitioned in sub-regions which are then assigned to the agents. A recent implementation of such a trajectory planning approach for the CPPP using UAVs can be found in Guastella et al. (2019). A survey on both single and multi-robot coverage, as well as different cellular decomposition techniques can be found in Galceran & Carreras (2013). Contrary to these types of approaches in which the motion is predetermined (i.e sweeping trajectory), the Vehicle Routing Problem approach offers the flexibility of exploring more possible trajectories. Another useful advantage of approaching CPPPs as VRPs is that already tested and proven VRP algorithms can be used for their solution. Furthermore, the VRP approach for solving CPP problems makes the incorporation constraints and different objectives while covering the area easier to implement. A disadvantage of the VRP approach is the long computational time required for instances with many cells. This paper utilizes the benefits of the VRP approach and proposes the CUAVRP for solving the humanitarian coverage path planning problem.

The difference between the CCVRP and CUAVRP is the replacement of the capacity constraints related to the ground vehicles and the demand of the customers, with UAV range constraints. The maximum range constraint of the UAVs, requires the vehicles to have enough remaining range to return to the base position at all times. This constraint has to be considered in both the solution construction and solution improvement processes.

An indirect difference occurs from the grid partitioning method, using square cells. All center-point distances between cells with a common side, are equal. Thus, the distance between a node and all of its neighboring (sharing a cell side) nodes, is equal. This characteristic makes constructing greedy solutions based on the nearest node less effective.

3. Mathematical formulation of the cumulative UAV routing problem

Let $V = \{0, 1, \ldots, n, n+1\}$ be the node set (node 0 and $n+1$ are the starting and ending nodes, respectively) and E the set of edges which connect every two nodes. The CUAVRP is defined on an undirected graph $G = (V, E)$. Each edge (i, j) is associated with a travel time u_{ij}. $V' = V \backslash \{0, n+1\}$, is the set of nodes. Each node $i \in V'$ must be visited exactly once. $R(R \geq 2)$ is the number of UAVs, and the maximum flight time of each vehicle is T. Variable t_i^k denotes the arrival time of UAV k at node i. A binary variable x_{ij}^k is equal to 1 if the UAV k traverses edge (i, j) from node i to j, otherwise 0. The CUAVRP aims to minimize the total arrival time at the nodes while each route must start at node 0 and end at node $n+1$, and the total flight time must not exceed T. The mathemat-

ical model of the CUAVRP, similarly to the CCVRP formulation by Ngueveu et al. (2010), can be stated as follows:

$$\min f(X) = \sum_{k=1}^{R} \sum_{i \in V'} t_i^k \tag{1}$$

s.t

$$\sum_{j \in V} x_{ji}^k = \sum_{j \in V} x_{ij}^k, \forall i \in V', \forall k \in \{1, \ldots, R\} \tag{2}$$

$$\sum_{k=1}^{R} \sum_{j \in V} x_{ij}^k = 1, \forall i \in V' \tag{3}$$

$$\sum_{j \in V} x_{0j}^k = 1, \forall k \in \{1, \ldots, R\} \tag{4}$$

$$\sum_{j \in V} x_{j,n+1}^k = 1, \forall k \in \{1, \ldots, R\} \tag{5}$$

$$\sum_{i \in V} \sum_{j \in V} x_{ij}^k u_{ij} \leq T, \forall k \in \{1, \ldots, R\} \tag{6}$$

$$t_i^k + u_{ij} - (1 - x_{ij}^k)G \leq t_j^k, \forall i \in V \backslash [n+1], \forall j \in V, \forall k \in \{1, \ldots, R\} \tag{7}$$

$$t_i^k \geq 0, \forall i \in V, \forall k \in \{1, \ldots, R\} \tag{8}$$

$$x_{ij}^k \in \{0, 1\}, \forall i \in V, \forall j \in V, i \neq j, \forall k \in \{1, \ldots, R\} \tag{9}$$

Formulation (1) states the objective function of the CUAVRP, which minimizes the sum of arrival times. Constraints (2) ensure that a UAV arriving at node i must depart from it. Constraints (3) ensure that each node is visited by exactly one UAV. Constraints (4) and (5) ensure that a route should begin at node 0 and end at node $n+1$, representing the UAV base. Constraints (6) mean the total flight duration of each UAV must not exceed its maximum flight time T. Constraints (7) calculate the arrival time at customers. By using a large positive constant G, it does not allow sub-tours to be created. The constraints' formulation is based on formulations used for VRPs with time windows. Constraints (8) ensure non-negative values for the arrival times and constraints (9) restrict x_{ij}^k variable to binary.

The set of V' vertices in the CUAVRP application on the CPPP is obtained by the cellular approximation method. Nodes 0 and $n+1$, denote the staring and ending position of each UAV route, respectively, and can be determined independently of the grid formation. The travel time u_{ij} between nodes i and j is calculated as the euclidean distance between them in this implementation. As described in Section 2.1, it is possible to create different variants of the CUAVRP by changing the way u_{ij} is calculated.

4. The proposed parallel GRASP - VND algorithm

4.1. Greedy randomized adaptive search procedure

Greedy Randomized Adaptive Search Procedure (GRASP) (Feo & Resende, 1995) is an iterative process, with each iteration consisting of two phases, a construction phase and a local search phase. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. The GRASP iterations terminate when some termination criterion, such as maximum number of iterations have occurred, is satisfied.

In the first phase, a randomized greedy technique provides feasible solutions incorporating both greedy and random characteristics. This phase can be described as a process which stepwise adds one node at a time to the partial (incomplete) solution. The choice of the next node to be added is determined by ordering all elements in a candidate list with respect to a greedy function. The list of best candidates is called the restricted candidate list (RCL). The probabilistic component of a GRASP is characterized by randomly choosing one of the best candidates in the list but not necessarily the top candidate. The heuristic is adaptive because the benefits associated with every element are updated during each iteration of the construction phase to reflect the changes brought on by the selection of the previous element. This choice technique allows for different solutions to be obtained at each GRASP iteration.

In the second phase, a local search is initialized from these points which in an iterative manner successively replaces the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. Many interesting extensions and improvements of the second phase have been proposed, such as Path Relinking (Aiex, Resende, Pardalos, & Toraldo, 2005; Alvarez-Valdes, Crespo, Tamarit, & Villa, 2008; Laguna & Marti, 1999), Expanding Neighborhood Search (Marinakis, 2012), Adaptive memory (Armentano & de Frana, 2007) and hybridizations such as GRASP-TABU for scheduling (Laguna & Gonzalez-Velarde, 1991), GRASP and VNS for the Max-Cut (Festa, Pardalos, Resende, & Ribeiro, 2001) and hybrid Genetic-GRASP for the TSP (Marinakis, Migdalas, & Pardalos, 2005). The outline of basic GRASP can be found in Algorithm 1 .

Algorithm 1: Basic GRASP.

```
Input: Instance
Result: BestSolutionFound
while Stop criterion not satisfied do
    ConstructGreedyRandomizedSolution (Solution);
    LocalSearch(Solution);
    if Solution is better than BestSolutionFound then
        BestSolutionFound ← Solution;
return BestSolutionFound;
```

Parallel algorithm implementations have been increasing in metaheuristics. The ever increasing number of cores and threads in new CPUs makes this direction a necessity in order to harvest the computational power available. Most parallel implementations of GRASP found in the literature either partition the solution space or the GRASP iterations and run the GRASP in parallel. These approaches can be considered as multiple-walk independent-thread, with the communication among threads during GRASP iterations being limited to the detection of program termination and gathering the best solution found over all processors (Resende & Ribeiro, 2016). Since the solution construction in each iteration is independent of the previous solution, basic GRASP is straight forward and efficient to parallelize. In other metaheuristics such as Ant Colony Optimization (ACO), which update shared information (i.e. pheromone) in each solution constructed, data-race conditions must be avoided, thus, are more complicated to parallelize (Pedemonte, Nesmachnow, & Cancela, 2011).

In this paper, a memory structure is added in the basic GRASP and the work is shared between a main thread and worker threads. The main thread executes the first phase of the GRASP, while the worker threads execute the second phase. Three different information exchange strategies between the main thread and worker threads are explored. In the following section, a description of the memory structure is given and the implemented parallel GRASP is explained in details.

4.2. Parallel weighted GRASP-VND

4.2.1. Memory structure

In the proposed approach, the basic GRASP is extended by adding a memory structure to assist the solution construction procedure. This weighted GRASP algorithm has each pair of nodes associated with a weight, which is updated during execution by an intensification method and a diversification method. This memory structure is used for the solution construction procedure discussed in the next subsection.

All weights in the memory structure are initialized at a constant number, w_0. At the end of each iteration of the algorithm, the intensification method is applied, updating the weights of the node pairs that are present in the best-so-far solution Eq. (10). The weights are increased by a constant preset rate, r_i. The diversification method is applied immediately after a solution is constructed on the node pairs present in the created solution, the weights of which are decreased by a constant preset rate, r_d Eq. (11).

$$w_{ij}^{new} = (1 + r_i) * w_{ij}^{old} \tag{10}$$

$$w_{ij}^{new} = (1 - r_d) * w_{ij}^{old} \tag{11}$$

4.2.2. Solution construction

In order to construct a solution with the GRASP framework, a restricted candidate list is generated at each step of inserting a node in a route. This restricted candidate list is value-based and consists of all nodes not yet visited, meeting the two following conditions:

i) Insertion of the node must not make the route infeasible.
ii) Distance between the node and the last inserted in the route must be less than $d_{\min} + a * d_{\max}$, where $d_{\min}$ and $d_{\max}$ the minimum and maximum distance to the last inserted node, among all other nodes and $a \in [0, 1]$ is a parameter regulating the greediness during construction.

For each node j in the restricted candidate list, there is an associated weight w_{ij} for being inserted after node i. The probability of choosing node j from the RCL, to be inserted after node i in the route, is given by Eq. (12).

$$p_{ij} = \begin{cases} \frac{w_{i,j}}{\sum_{l \in RCL} w_{i,l}}, & \text{if } j \in RCL \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

Using the above probabilities for the weighted choice, the route construction continues until the RCL is empty. When it's empty, the vehicle returns to the starting node. If all nodes have been added in the solution, the solution construction procedure has finished. Otherwise, a new route is generated with the same process. The generated routes are guaranteed to be feasible, but the solution might not be, as the number of routes generated might exceed the fleet size allowed by the instance. If the solution is infeasible it's discarded and a new solution is constructed.

Although, the graph is undirected, the arc cost is not only depended on the distance between the two nodes, but also on its position in the route. After each route is constructed, the cost of the reverse route is checked. If the reversed cost is less, then the order of the nodes is reversed. For efficiently calculating the cost of the reversed route, the following property given by Ngueveu et al. (2010) is used, where $d_{[i][j]}$ is the distance between the nodes at positions i and j of the route:

$$C_{reversed} = nD - C, \text{where } D = \sum_{j=1}^{n+1} d_{[j-1][j]} \tag{13}$$

The complete solution construction procedure can be found in Algorithm 2

Algorithm 2: Solution construction.

Input: $N = \{1, 2..n\}, \alpha, w, r_d$
`/* N the list of nodes to be added in the solution. */`
Result: S
repeat
  $l \leftarrow 1$ `// the vehicle index`
  $next \leftarrow 0$ `// the next node`
  $R_l \leftarrow \{next\}$ `// the route`
  $S = \{\emptyset\}$ `// the solution`
  while $N \neq \emptyset$ **do**
    $RCL \leftarrow$ generateRCL(N, α);
    if $RCL \neq \emptyset$ **then**
      $next \leftarrow WeightedChoice(RCL, w)$;
      `// probabilities calculated by Eq:(12) as discussed in Section 4.2.2.`
      $R_l \leftarrow R_l \cup \{next\}$;
      $N \leftarrow N - \{next\}$;
    else
      $R_l \leftarrow R_l \cup \{0\}$;
      $S \leftarrow S \cup \{R_l\}$;
      $l \leftarrow l + 1$;
      $R_l \leftarrow R_l \cup \{0\}$;
      $next \leftarrow 0$;
  DiversificationMethod(w, r_d);
until S *is feasible*;
return S;

4.2.3. Parallelization

Parallelization is achieved through a multithreaded implementation of the algorithm. The proposed GRASP consists of a main thread and a set number of worker threads. The main thread is responsible for generating new solutions, updating the memory structure and coordinating the worker threads. The worker threads run the local search procedure.

Three different approaches of Parallel Weighted GRASP-VND are implemented and compared, each utilizing a different communication and information exchange strategy between the threads. All variants share the same solution construction, intensification and diversification processes. Below is a comprehensive description of the three different strategies:

1. At each iteration, all worker threads return the improved solution and take a new solution from the main thread, in order to improve it.
2. At each iteration, the worker thread with the best improved solution, continues the local search procedure on its solution. The rest of the worker threads, are given a new solution from the main thread in order to improve it.
3. At each iteration, a random thread returns the improved solution and is given a new solution from the main thread. The rest of the worker threads continue their local search procedure with their current solutions.

For the rest of the paper these three strategies will be referred as "All Return" (AR), "Best Continue" (BC) and "Random Return" (RR). Algorithms 3–5 illustrate the pseudo code for each strategy, respectively. Algorithm 6 shows the procedure shared among all three versions.

In order to avoid data race conditions between threads on the shared memory, a locking mechanism had to be implemented. Mutexes, have a performance penalty and they may introduce additional issues such as deadlocking. With that in mind, this algorithm is designed with the goal of minimizing communication between threads and the need for locking. The main thread only

Algorithm 3: All return strategy.

Input: $T = \{t_1, t_2, .., t_{T_{\max}}\}, S, w, r_i$
Result: *BestImprovedSol*
Data: *threadSol*
foreach *t in T* **do**
  if *t.complete_flag==true* **then**
    t.pause();
    threadSol ← *t.retrieve_and_setnew*(*S*);
    t.complete_flag = *false*;
    t.start();
    IntensificationMethod(*threadSol*, *w*, r_i);
    BestImprovedSol ← UpdateBestImproved(*threadSol*);
return BestImprovedSol;

Algorithm 4: Best continue strategy.

Input: $T = \{t_1, t_2, .., t_{T_{\max}}\}, S, w, r_i$
Result: *BestImprovedSol*
Data: t_{best}
foreach *t in T* **do**
  if *t.complete_flag==true* **then**
    threadSol ← *t.get_result*(*S*);
    BestImprovedSol ← UpdateBestImproved(*threadSol*);
    t_{best} ← UpdateBestImproved(*t*);
    IntensificationMethod(*threadSol*, *w*, r_i);
foreach *t in T* **do**
  if $t \neq t_{best}$ **then**
    t.pause();
    t.setnew(*S*);
    t.complete_flag = *false*;
    t.start();
t_{best}.*setnew*(*S*);
return BestImprovedSol;

Algorithm 5: Random return strategy.

Input: $T = \{t_1, t_2, .., t_{T_{\max}}\}, S, w, r_i$
Result: *BestImprovedSol*
Data: *threadSol*
t ← ChooseRandomThread(*T*);
if *t.complete_flag==true* **then**
  t.pause();
  threadSol ← *t.retrieve_and_setnew*(*S*);
  t.complete_flag = *false*;
  t.start();
  IntensificationMethod(*threadSol*, *w*, r_i);
  BestImprovedSol ← UpdateBestImproved(*threadSol*);
return BestImprovedSol;

exchanges solutions with the worker threads, a process with insignificant duration compared to the duration of the procedures performed when both threads run. For this reason, a spin lock structure is used as a mutex, to ensure atomic operations on the shared data. In addition to limiting the shared data, each worker thread has a flag variable, which allows the main thread to see whether it has completed at least one run of the VND procedure (for VNDiters), before it pauses the thread and exchange the improved solution with a new one. In order to control the state of the worker threads (start, pause, stop) a conditional variable is used in its loop of execution.

Algorithm 6: Parallel weighted GRASP-VND.

Input: *Instance*, *GRASPiters*, *VNDiters*, *a*, r_i, r_d, w_0
Result: *BestSolutionFound*
Initialize T ={t_1(*VNDiters*), .., $t_{T_{\max}}$(*VNDiters*)}, $w(w_0)$;
`/* T the list of worker threads */`
for *iter* ← 1 *to GRASPiters* **do**
  BestConstructed ← ConstructGRASPSolution (*a*, *w*, r_d);
  for *cs* ← 1 *to NumConstructed* **do**
    S ← ConstructGRASPSolution (*a*, r_d);
    `/* DiversificationMethod(w, r_d) is applied inside the construction function, in both feasible and infeasible solutions */`
    if *cost(S)<cost(BestConstructed)* **then**
      BestConstructed ← *S*;
  ImprovedSolution ←ThreadStrategy(*T*, *BestConstructed*, *w*, r_i);
  `/* IntensificationMethod(S, w, r_i) is applied for each improved solution, in all three strategies. */`
  if *cost(ImprovedSolution)* < *cost(BestSolutionFound)* **then**
    BestSolutionFound ← *ImprovedSolution*;
  IntensificationMethod(*BestSolutionFound*, *w*, r_i);
Stop(T);
return *BestSolutionFound*;

In the case where the main thread only needs to retrieve the improved solution without setting a newly constructed one (i.e. BC strategy), the thread is not paused. Instead the main thread tries to acquire the spin-lock, which is very unlikely to be locked by the worker-thread, therefore almost no waiting is expected.

Algorithm 7 , displays in pseudocode the execution loop of the

Algorithm 7: Worker thread.

Input: *VNDiters*
Data: *SharedSolution*, *spinlock*, *condvar*, *complete_flag*
repeat
  while *NOT condvar.pause_thread*() **do**
    spinlock.lock();
    inSol ← *SharedSolution*;
    spinlock.unlock();
    outSol ← VND (*inSol*, *VNDiters*);
    spinlock.lock();
    SharedSolution ← *outSol*;
    complete_flag = *true*;
    spinlock.unlock();
until *condvar.stop_thread*();

worker thread. When the main thread wants to check and exchange an improved solution, it checks the *complete_flag*. If the flag is false, then, it doesn't do anything. If the flag is true, then, it proceeds to pause the worker thread through the conditional variable and tries to lock the spin-lock. Once locked, the main thread retrieves the improved solution (SharedSolution), puts a newly constructed solution in its place and sets the *complete_flag* to false. Then it unlocks the spin-lock and restarts the worker thread loop.

The memory structure is leveraged by the intensification and diversification methods in four ways:

1. The best-so-far solution intensification method increases exploitation of the best solution.

2. Each improved solution, also, applies intensification method and increases exploitation around good solutions.
3. The diversification method applied immediately after a solution is constructed, increases exploration of solution.
4. The initialization of all weights with the same value, allows for higher exploration of the solution space in the start of the algorithm, while increasing exploitation around the best solutions found in the later stages of execution.

4.3. Variable neighborhood decent

The VND is a deterministic variant of the Variable Neighborhood Search framework originally proposed by Mladenovic & Hansen (1997). It has been used as a local search routine for many metaheuristics and has been implemented in various ways (Mjirda, Todosijevic, Hanafi, Hansen, & Mladenovic, 2017). It is based on the idea of systematical changes of neighborhood structures within the search procedure. In this paper, a Pipe VND (P-VND) is used, where the search continues in the same neighborhood if it improves the solution. When no further improvement can be made in the particular neighborhood it proceeds in the next neighborhood. This process is repeated until no further improvement can be made by the last neighborhood.

Let $N = \{N_1, N_2, \ldots, N_{k_{\max}}\}$ be a set of operators that map a given solution to a neighborhood structure $N_k(S)$. Algorithm 8 shows the VND procedure.

Algorithm 8: Variable neighborhood descent.

Data: $S, N = \{N_1, N_2, .N_k\}, VNDiters$
Result: S'
$S' \leftarrow S$;
for $iter \leftarrow 1$ *to* $VNDiters$ **do**
  $R_i, R_j \leftarrow RandomRouteSelection(S')$;
  for $k \leftarrow 1$ *to* $k_{\max}$ **do**
    $improved \leftarrow False$;
    repeat
      $S', improved \leftarrow N_k(S', R_i, R_j)$;
    until $improved = False$;

To complement the exploitation of solutions, the proposed VND procedure consists of the following inter-route and intra-route local search operators (neighborhoods), that are applied to the given solution:

- **1-1 Swap**: two nodes of the same route swap their position.
- **Adjacent-Swap**: two adjacent nodes in a route swap their position.
- **2-opt**: the nodes in a range of positions inside a route are reversed.
- **1-0 Relocation**: One node is removed from a route and is inserted in another route.
- **2-0 Relocation**: Two consecutive nodes are removed from a route and are inserted in another route.
- **1-1 Exchange**: Two nodes from different routes exchange their positions.
- **2-1 Exchange**: Two consecutive nodes from a route are exchanged with one node from another route.
- **2-2 Exchange**: Two consecutive nodes from a route are exchanged with two consecutive nodes from another route.
- **3-3 Exchange**: Three consecutive nodes from a route are exchanged with three consecutive nodes from another route.

5. Computational results

The algorithms were coded in C++ and compiled with GCC 10.0. All tests were run using a 2014 Intel®Core i7-4770 CPU (3.40 GHz) with 7.7 GB RAM running Manjaro Linux 20.1.

In order to assess the performance of the implemented Parallel Weighted GRASP-VND, we run tests for all strategy variants in the closest relative problem with known benchmark instances, the Cumulative Capacited Vehicle Routing Problem (CCVRP). The instances used are 7 instances (CMT) proposed by Christofides, Mingozzi, & Toth (1979a) ranging from 50 to 199 nodes, which were used in Ke & Feng (2013); Lysgaard & Whlk (2014); Ngueveu et al. (2010); Ribeiro & Laporte (2012); Sze, Salhi, & Wassan (2017) and Nucamendi-Guilln, Angel-Bello, Martnez-Salazar, & Cordero-Franco (2018).

The choice to use this similar problem to benchmark the algorithmic approach of the paper, was made based on their shared cost structure. The CCVRP differs from the CUAVRP, as there are demand and capacity constraints instead of a maximum range constraint and instead of UAVs, road vehicles are considered. The changes in the actual code were straightforward to make, as one constraint gets replaced with another while all the logic remains the same.

For the CUAVRP, as there are no known benchmarks instances in the literature, 28 total instances were created. These were split into two sets, one with 18 instances based on the data points of CMT1 and the other containing 7 instances based on CMT11. The positions (coordinates) of the customers in the CMT instances, were used as possible positions of objects or people that a UAV would be looking for in a search & rescue operation over an area. To determine the polygonal convex area of interest, A, for the Coverage Path Planning problem, as defined in Section 2.1, Graham's scan algorithm (Graham, 1972) was used to obtain the corresponding set P of border vertices. As a base for deploying and retrieving the UAVs, the depot coordinates of the CCVRP instance was used.

In real-world UAV searching applications, cell sizes depend on the camera sensor and altitude of the UAV. The number of cells increases as the area of interest increases. In the practical UAV path planning simulation for search and rescue operations of de Alcantara et al. (2019), a 48×48 grid is used to map an area of 23.04 square kilometer. To accomodate the wide range of areas that might be considered for a humanitarian UAV search operations, different scenarios are created, ranging from 4 to 20 UAVs and 40 to 2239 nodes. The instances are created for different values of the side length of the viewing rectangle d, UAV fleet size and maximum range. Fig. 2 shows examples of solution for CMT1 and CMT11 generated instances for the CUAVRP.

5.1. Parameter settings

The parameters needed to be set for the Parallel Weighted GRASP-VND are the following:

- $T_{\max}$, the number of worker threads used.
- *GRASPiter*, the number of iterations.
- *NumConstructed*, the number of constructed solutions in each GRASP iteration.
- *VNDiter*, number of VND iterations the worker thread executes before updating the shared solution.
- a, the parameter controlling indirectly the size of the RCL.
- w_0, the initial weight of edges.
- r_i, the rate of intensification.
- r_d, the rate of diversification.

For all strategies and instances, the number of worker threads $T_{\max}$ was set to 3. Therefore, the total number of threads utilized by the algorithms is 4. The *GRASPiter* parameter is set

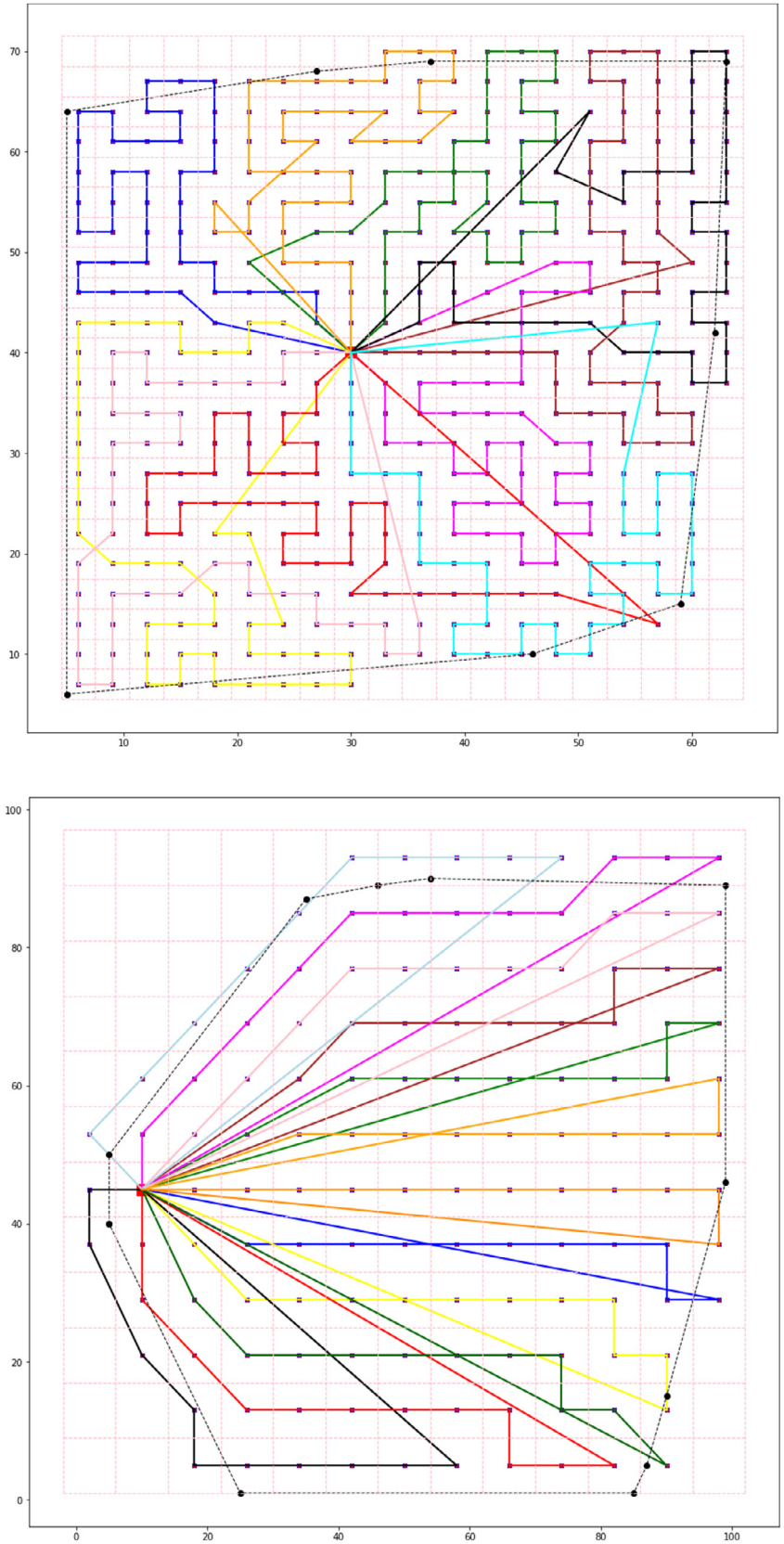

Fig. 2. CUAVRP solutions on CTM1 based instance for $d = 3$ (left) and CMT11 based instance for $d = 8$ (right).

Table 1
Results for the CCVRP on CMT instances (Christofides et al., 1979a) (BKS obtained in bold).

				Parallel weighted GRASP-VND AR				Parallel weighted GRASP-VND BC				Parallel weighted GRASP-VND RR			
Instance	n	k	BKS	Best	$T_{Avg}(s)$	Avg	Gap %	Best	$T_{Avg}(s)$	Avg	Gap %	Best	$T_{Avg}(s)$	Avg	Gap %
*CMT*1	50	5	2230.35*	**2230.35**	0.59	2274.14	0.00	**2230.35**	0.56	2269.89	0.00	**2230.35**	0.54	2270.00	0.00
*CMT*2	75	10	2391.63*	**2391.63**	2.44	2400.33	0.00	**2391.63**	3.01	2415.64	0.00	**2391.63**	2.39	2439.79	0.00
*CMT*3	100	8	4045.42*	**4045.42**	2.89	4089.08	0.00	**4045.42**	2.93	4085.8	0.00	**4045.42**	3.1	4090.67	0.00
*CMT*4	150	12	4987.52*	**4987.52**	9.62	5003.98	0.00	**4987.52**	9.88	4995.73	0.00	**4987.52**	9.6	4993.83	0.00
*CMT*5	199	17	5806.02	**5806.02**	20.4	5842.51	0.00	5809.16	20.86	5848.20	0.06	5818.21	20.55	5839.94	0.21
*CMT*11	120	7	7314.55	**7314.55**	10.97	7372.43	0.00	**7314.55**	13.53	7379.31	0.00	**7314.55**	12.52	7386.12	0.00
*CMT*12	100	10	3558.92*	**3558.92**	6.46	3561.81	0.00	**3558.92**	6.45	3558.92	0.00	**3558.92**	6.37	3558.96	0.00
Total (s)					53.37				57.22				55.07		

Table 2
Average elapsed CPU time (s) of other approaches for CMT instances (Christofides et al., 1979a) (BKS obtained in bold).

	BCP		MA1		MA2		ALNS		2-Phase		AVNS		IG-PRB		IG-CE	
Instance	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$	Gap_{best}	$T_{Avg}(s)$
*CMT*1	**0.00**	27.00	**0.00**	10.63	**0.00**	3.70	**0.00**	30.29	**0.00**	14.70	**0.00**	11.89	**0.00**	10.48	**0.00**	7.36
*CMT*2	**0.00**	22511.00	1.27	27.78	1.57	2.04	**0.00**	60.77	**0.00**	18.73	**0.00**	15.04	**0.00**	38.81	**0.00**	21.05
*CMT*3	**0.00**	25033.00	0.68	97.91	**0.00**	40.46	**0.00**	172.45	**0.00**	50.80	**0.00**	41.03	**0.00**	77.92	**0.00**	64.79
*CMT*4	**0.00**	7801.00	**0.00**	449.44	**0.00**	188.41	**0.00**	235.12	**0.00**	77.23	**0.00**	60.54	**0.00**	48.77	**0.00**	48.25
*CMT*5	(LB 5775.28)	28800.00	0.07	1035.45	0.05	629.27	0.56	277.37	0.06	112.80	**0.00**	89.75	0.35	72.88	0.06	62.29
*CMT*11	(LB 7197.69)	28800.00	0.05	160.64	0.45	68.83	0.02	202.07	**0.00**	59.70	**0.00**	48.32	0.08	40.90	**0.00**	22.17
*CMT*12	**0.00**	841.00	**0.00**	38.20	0.01	23.66	**0.00**	152.74	**0.00**	44.77	**0.00**	34.21	**0.00**	80.90	**0.00**	61.61
CPU freq.	2.53 Ghz		2.4 Ghz		2.4 Ghz		2.0 Ghz		2.4 Ghz		3.4 Ghz		2.4 Ghz		2.4 Ghz	

BCP by Lysgaard & Whlk (2014), MA1 and MA2 by Ngueveu et al. (2010).
ALNS by Ribeiro & Laporte (2012), 2-Phase by Ke & Feng (2013), AVNS by Sze et al. (2017).
IG-PRB and IG-CE by Nucamendi-Guilln et al. (2018).

equal to $[Numberofnodes] \times 100$, $NumConstructed = 10$, $VNDiter = GRASPiter/NumConstructed$. The initial weights of node pairs were set to 1. The rate of intensification $r_i = 0.01$ and the rate of diversification $r_d = r_i/NumConstructed$. These parameter values were set after experimentation with different combinations. Multiple combination are able to produce quality results.

5.2. Experimental results on CCVRP instances

In order to test the performance of the metaheuristic algorithm, the Cumulative Capacitated Vehicle Routing Problem was used, which has been solved using both exact methods and metaheuristics. Out of the seven CMT instances, five instances (denoted with '*') have their best-known solution (BKS) proven to be optimal, in the brand-and-cut-and-price approach of Lysgaard & Whlk (2014). The gap to best known solution (BKS) is calculated by the below formula:

$$Gap(Value) = 100 * (Value - BKS)/BKS \quad (14)$$

Table 1 shows the results on CMT instances of the three different Parallel Weighted GRASP-VND strategies, for the CCVRP. Column 1 denotes the instance names, columns 2 and 3 indicate the number of customers and number of vehicles respectively, column 4 shows the BKS. The values in columns 5, 9, 13 are the best solutions found in the 15 runs of each strategy, columns 6, 10, 14 are the average computational time in seconds, columns 7, 11, 15 are the average elapsed time of runs, columns 8, 12, 16 are the average cost of the solutions and columns 9, 13, 17 show the gap % to the BKS.

The AR version of the algorithm was able to reach the BKS in all CMT instances. Both BC and RR versions were able to reach the BKS in 6 out of 7 instances for the chosen a, and with only a 0.06% and 0.21% gap, respectively.

In Table 2 columns 2, 4, 6, 8, 10, 12, 14 present the best results on CMT instances in literature for the CCVRP by metaheuristic approaches. The Parallel Weighted GRASP-VND AR is the only other algorithm to reach the BKS in CMT5 found by the AVNS implementation of Sze et al. (2017). AR strategy is also the fastest among the three while BC is the slowest. Although BC and RR reached 6 out of 7 BKS, BC's gap was only 0.06% in comparison to the 0.21% of the RR strategy.

The AR strategy allows for more exploration compared to the other two strategies, as it returns all the improved solutions and continues the local search on newly generated. The other strategies do not replace solutions in all threads and, thus, favor exploitation. This is the reason that AR strategy was able to overcome local minimums and reach the BKS in CMT5.

Table 2 also shows the average computational time for the CMT instances of the single threaded metaheuristic approaches in the literature as well as the exact algorithm of Lysgaard & Whlk (2014). Although the execution time cannot be accurately compared, since different hardware is used, a rough estimate can be obtained taking in consideration the CPU's frequency. Since, our approach utilizes multiple threads, the elapsed time columns are included as a reference for scale.

5.3. Experimental results on CUAVRP instances

Tables 3 and 4 display the results on the CUAVRP instances of the three different Parallel Weighted GRASP-VND strategies using the min-sum objective. For this objective the best solution values are computed as the sum of arrival times. Table 3 contains the instances based on CMT1 and Table 4 contains the instances based on CMT11. Column 1 denotes the instance names, columns 2 and 3 indicate the number of nodes and number of UAVs respectively, column 4 shows the BKS. The values in columns 5, 9, 13 are the best solutions found in the 15 runs of each strategy, columns 6, 10, 14 are the average computational time in seconds, columns 7, 11, 15 are the average elapsed time of runs, columns 8, 12, 16 are the average cost of the solutions and columns 9, 13, 17 show the gap % to the BKS. In the instance name, the number next to d denotes the side length of the sensor's viewing square, the number next to

Table 3
Results for the CUAVRP with Min-sum objective on CMT1 based instances. (BKS obtained in bold).

				Parallel weighted GRASP-VND AR				Parallel weighted GRASP-VND BC				Parallel weighted GRASP-VND RR			
Instance	n	k	BKS	Best	Avg	$T_{Avg}(s)$	Gap %	Best	Avg	$T_{Avg}(s)$	Gap %	Best	Avg	$T_{Avg}(s)$	Gap %
d11_k6_r200	40	6	1676.23	**1676.23**	1676.23	0.40	0.00	**1676.23**	1676.23	0.39	0.00	**1676.23**	1676.23	0.41	0.00
d11_k4_r200	40	4	2310.00	**2310.00**	2311.21	0.43	0.00	**2310.00**	2311.21	0.43	0.00	**2310.00**	2310.60	0.39	0.00
d10_k3_r300	40	3	3010.00	**3010.00**	3010.00	0.54	0.00	**3010.00**	3010.00	0.50	0.00	**3010.00**	3010.00	0.53	0.00
d10_k5_r200	40	5	1923.14	**1923.14**	1923.14	0.49	0.00	**1923.14**	1923.14	0.43	0.00	**1923.14**	1923.14	0.46	0.00
d9_k8_r200	57	8	2120.38	**2120.38**	2120.73	1.20	0.00	**2120.38**	2120.38	1.09	0.00	**2120.38**	2120.73	1.05	0.00
d9_k5_r200	57	5	3122.74	**3122.74**	3122.74	1.10	0.00	**3122.74**	3122.74	0.99	0.00	**3122.74**	3122.74	1.00	0.00
d9_k4_r300	57	4	3780.00	**3780.00**	3783.83	1.12	0.00	**3780.00**	3784.77	1.07	0.00	**3780.00**	3781.49	1.02	0.00
d8_k4_r300	68	4	4763.31	**4763.31**	4767.61	1.86	0.00	**4763.31**	4766.58	1.84	0.00	**4763.31**	4766.23	1.97	0.00
d7_k4_r300	90	4	7247.90	**7247.90**	7256.42	3.64	0.00	**7247.90**	7258.26	3.52	0.00	**7247.90**	7254.57	3.58	0.00
d6_k4_r300	109	4	9076.97	9080.49	9098.65	6.78	0.04	9080.49	9089.93	6.27	0.04	**9076.97**	9093.37	6.22	0.00
d6_k12_r250	109	12	3598.60	3604.56	3619.12	6.87	0.17	3598.60	3605.46	6.10	0.00	**3601.04**	3617.57	6.89	0.07
d5_k8_r300	159	8	8386.90	8401.87	8459.90	23.68	0.18	8389.76	8438.18	21.30	0.03	**8386.90**	8454.84	21.20	0.00
d5_k4_r400	159	4	16014.14	16015.35	16051.41	20.37	0.01	**16014.14**	16054.96	20.12	0.00	16020.71	16049.69	21.18	0.04
d5_k20_r250	159	20	4464.93	4477.49	4489.32	22.31	0.28	**4464.93**	4474.22	19.84	0.00	4483.58	4491.16	18.99	0.42
d4_k5_r650	231	5	21786.60	21858.80	22030.05	70.27	0.33	21847.90	21954.87	64.70	0.28	**21786.60**	22002.59	64.53	0.00
d4_k7_r300	231	7	15789.80	15821.40	15875.53	59.65	0.20	**15789.80**	15858.43	55.56	0.00	15794.20	15879.96	54.84	0.03
d4_k15_r250	231	15	8266.94	**8266.94**	8307.97	62.77	0.00	8276.92	8307.94	58.14	0.12	8272.60	8316.49	58.87	0.07
d3_k10_r300	405	10	25836.18	**25836.18**	26024.54	332.18	0.00	25838.75	25973.97	315.35	0.01	25982.28	26049.53	328.71	0.57
d2_k15_r1200	1816	15	240082.59	**240082.59**	241791.45	1151.31	0.0	242358.16	243511.54	1090.38	0.94	240526.22	242231.28	1103.43	0.18
d1.9_k14_r1500	2032	14	300986.79	**300986.79**	304176.02	1704.20	0.0	304897.55	307151.98	1329.27	1.29	302165.42	304530.65	1510.27	0.39
d1.8_k15_r1500	2239	15	326498.67	**326498.67**	328793.22	2065.60	0.0	330468.01	334546.78	1769.82	1.21	329116.98	330202.70	1835.13	0.80
Average							0.06				0.18				0.12
Total						5536.77				4767.11				5040.67	

Table 4
Results for the CUAVRP with Min-sum objective on CMT11 based instances. (BKS obtained in bold).

				Parallel weighted GRASP-VND AR				Parallel weighted GRASP-VND BC				Parallel weighted GRASP-VND RR			
Instance	n	k	BKS	Best	Avg	$T_{Avg}(s)$	Gap %	Best	Avg	$T_{Avg}(s)$	Gap %	Best	Avg	$T_{Avg}(s)$	Gap %
d9_k6_r800	112	6	10092.02	10157.33	10214.66	7.41	0.65	**10092.02**	10141.88	7.59	0.00	10099.23	10165.16	9.07	0.07
d8_k12_r450	132	12	7795.45	7800.54	7814.51	12.38	0.07	**7795.45**	7801.32	11.84	0.00	7800.54	7815.40	11.92	0.07
d8_k17_r450	132	17	7266.57	7269.00	7276.23	11.98	0.03	**7266.57**	7270.76	11.78	0.00	7271.90	7278.44	11.19	0.07
d8_k5_r850	132	5	14481.97	14540.10	14684.85	12.69	0.40	**14481.97**	14593.27	11.62	0.00	14564.97	14695.45	12.21	0.57
d7_k4_r1100	166	4	24630.44	24690.84	24904.13	29.98	0.25	**24630.44**	24842.99	30.35	0.00	24688.60	24942.80	26.90	0.24
d6_k8_r850	217	8	19201.21	19282.96	19421.67	57.60	0.43	**19201.21**	19286.51	57.93	0.00	19212.75	19403.58	55.75	0.06
d5_k10_r850	324	10	28543.34	28838.11	29063.07	188.33	1.03	**28543.34**	28853.61	171.90	0.00	28782.89	29036.46	162.34	0.84
Average.							0.41				0.00				0.27
Total						320.37				303.01	0.00			289.38	

k is the UAV fleet size, and the number next to r represents the maximum distance the UAVs can travel.

The Parallel Weighted GRASP-VND with the BR strategy yields the best results for the CUAVRP in the small and medium size instances, having an average gap of 0.06% for the 18 first CMT1 based instances and reaching the BKS in all 7 CMT11 based instances. AR strategy performs better for the very large instances with nodes ranging from 1816 to 2239. RR strategy outperforms AR in CMT11 based instances with an average gap of 0.27% compared to 0.41% of the latter.

In terms of computation time, BC strategy is, also, the faster among the three for the first set and second best for the second set. AR is the slowest in both sets. For total computational time in both sets combined, BC is 15.5% faster than the AR and 5.1% faster than RR. As expected the difference is mostly noticed in the very large instances.

These two sets of instances have different topology. The CMT1 based has the starting point in the center area with the nodes spread around it. Instances based on CMT11 have the starting point on the left border with almost all nodes being on the right of it. As observed by the results, the AR strategy favours first type of topology in the grid and thus is able to obtain the best results overall. For the second type of grid, AR is the worst performer among the three strategies. The BC strategy allows for more *VNDiter* iterations to be performed on the best-so-far-solutions and, thus, the algorithm can further exploit good solution.

5.4. Min-max objective

In order to compare the difference in the results between the two objectives, the algorithms have been adapted as necessary.

Tables 5 and 6 display the results for the CUAVRP with the Min-max objective. Similarly to the Min-sum objective, AR and RR strategies perform better for the CMT1 based instances. For this objective the RR strategy is able to surpass the AR, even in the largest instances. The performance of the BC strategy is lacklustre and can be attributed to the increased exploitation of the best-so-far solution which might have limited the exploration of other solutions. For the CMT11 based instances, RR performance is also the best overall. Contrary to the Min-sum objective, the BC is the worst performer for using the Min-max objective.

The Min-max objective provides useful insights on how the grid affects the MA time and the coverage. Instances d9_k4_r300, d8_k4_r300, d7_k4_r300, d6_k4_r300 d5_k4_r400 have the same number of UAVs and differ in the cell sizes. Their respective number of nodes are 57, 68, 90, 109, 159. As the results suggest, the

Table 5
Results for the CUAVRP with Min-max objective on CMT1 based instances. (BKS obtained in bold).

				Parallel weighted GRASP-VND AR				Parallel weighted GRASP-VND BC				Parallel weighted GRASP-VND RR			
Instance	n	k	BKS	Best min-max	Avg	$T_{Avg}(s)$	Gap %	Best min-max	Avg	$T_{Avg}(s)$	Gap %	Best min-max	Avg	$T_{Avg}(s)$	Gap %
d11_k6_r200	40	6	77.00	**77.00**	80.64	0.38	0.00	**77.00**	79.73	0.44	0.00	**77.00**	1676.23	0.40	0.00
d11_k4_r200	40	4	110.00	**110.00**	110.91	0.44	0.00	**110.00**	113.64	0.41	0.00	**110.00**	112.73	0.38	0.00
d10_k3_r300	40	3	140.0.0	**140.00**	140.0	0.52	0.00	**140.00**	140.0	0.50	0.00	**140.0**	140.00	0.53	0.00
d10_k5_r200	40	5	90.00	**90.00**	90	0.47	0.00	**90.00**	90	0.42	0.00	**90.00**	90	0.48	0.00
d9_k8_r200	57	8	66.72	**66.72**	66.72	1.25	0.00	**66.72**	66.72	1.14	0.00	**66.72**	66.72	1.04	0.00
d9_k5_r200	57	5	108.00	**108.00**	108.00	1.10	0.00	**108.00**	108.00	1.01	0.00	**108.00**	108.00	0.97	0.00
d9_k4_r300	57	4	126.00	**126.00**	126.00	1.14	0.00	**126.00**	129.72	1.11	0.00	**126.00**	126.00	1.05	0.00
d8_k4_r300	68	4	139.31	**139.31**	139.78	1.81	0.00	**139.31**	143.49	1.93	0.00	**139.31**	140.11	1.88	0.00
d7_k4_r300	90	4	161.00	**161.00**	161.00	3.49	0.00	**161.00**	161.00	3.48	0.00	**161.00**	161.00	3.58	0.00
d6_k4_r300	109	4	164.48	166.97	167.69	6.45	1.51	**164.48**	166.54	6.50	0.00	**164.48**	166.43	5.96	0.00
d6_k12_r250	109	12	61.41	**61.41**	62.55	6.89	0.00	**61.41**	63.90	5.85	0.00	**61.41**	62.91	7.19	0.00
d5_k8_r300	159	8	106.21	**106.21**	108.75	22.92	0.00	109.14	112.38	22.31	2.76	107.07	108.72	21.56	0.81
d5_k4_r400	159	4	202.07	204.14	205.60	21.28	1.02	207.07	214.89	20.87	2.47	**202.07**	205.18	20.46	0.00
d5_k20_r250	159	20	50.81	**50.81**	51.44	22.25	0.00	51.21	53.72	19.66	0.79	**50.81**	51.76	19.40	0.00
d4_k5_r650	231	5	192.00	194.60	197.72	71.56	1.35	**192.00**	197.65	66.20	0.00	194.60	199.94	62.32	1.35
d4_k7_r300	231	7	139.31	**139.31**	142.19	57.95	0.00	143.31	146.09	57.47	2.87	139.54	142.69	54.05	0.17
d4_k15_r250	231	15	68.94	**68.94**	70.55	63.21	0.00	72.25	107.50	56.27	4.80	70.60	71.07	57.17	2.41
d3_k10_r300	405	10	131.48	132.90	134.05	346.59	1.08	131.83	133.95	318.41	0.27	**131.48**	133.98	319.81	0.00
d2_k15_r1200	1816	15	296.52	**296.52**	299.88	1,168.59	0.00	298.84	300.12	1,131.62	0.78	297.23	299.85	1,057.44	0.24
d1.9_k14_r1500	2032	14	334.20	334.87	338.72	1,731.81	0.20	334.22	338.52	1,272.90	0.01	**334.20**	337.36	1,481.74	0.00
d1.8_k15_r1500	2239	15	330.90	334.60	337.27	2,048.79	1.12	334.01	338.68	1,745.69	0.94	**330.90**	335.73	1,802.37	0.00
Average							0.29				0.74				0.23
Total						5578.90				4734.19				4919.80	

Table 6
Results for the CUAVRP with Min-Max objective on CMT11 based instances. (BKS obtained in bold).

				Parallel weighted GRASP-VND AR				Parallel weighted GRASP-VND BC				Parallel weighted GRASP-VND RR			
Instance	n	k	BKS	Best min-max	Avg	$T_{Avg}(s)$	Gap %	Best min-max	Avg	$T_{Avg}(s)$	Gap %	Best min-max	Avg	$T_{Avg}(s)$	Gap %
d9_k6_r800	112	6	178.45	180.00	183.32	7.26	0.65	180.00	183.62	7.55	0.00	**178.45**	183.11	8.99	0.07
d8_k12_r450	132	12	107.88	**107.88**	108.66	11.88	0.07	**107.88**	110.98	11.43	0.00	**107.88**	109.41	12.14	0.07
d8_k17_r450	132	17	101.82	**101.82**	103.14	12.21	0.03	103.09	105.09	11.33	0.00	102.18	103.26	11.68	0.07
d8_k5_r850	132	5	227.31	229.20	232.85	13.07	0.40	228.61	238.89	11.55	0.00	**227.31**	233.68	12.66	0.57
d7_k4_r1100	166	4	309.69	311.35	321.00	30.17	0.25	**309.69**	325.77	30.91	0.00	312.55	320.59	27.97	0.24
d6_k8_r850	217	8	180.00	181.45	187.12	59.97	0.43	180.38	188.51	59.61	0.00	**180.00**	181.45	53.15	0.06
d5_k10_r850	324	10	178.57	179.46	182.27	195.43	1.03	180.35	187.81	172.61	0.00	**178.57**	183.55	167.24	0.84
Average.							0.50				0.55				0.18
Total						329.98				311.68	0.00			300.47	

MA times increases linearly as the number of nodes in the grid increases. On the other hand, the computational time needed is growing at an exponential rate. In practice, the grid size is determined by multiple factors, such as the sensor on board the UAV and the capabilities of the image recognition algorithm used.

Table 7 displays the best results obtained using the Min-sum and Min-max objective on the CUAVRP. $MA(Best_{\text{Min-max}})$ and $Cost(Best_{\text{Min-max}})$ denote the arrival time at the last customer (Maximum Arrival time) of the best result obtained and the corresponding CUAVRP cost of the solution, using the Min-max objective. Likewise, $Cost(Best_{\text{Min-sum}})$ is the CUAVRP cost of the solution obtained using the Min-sum objective and $MA(Best_{\text{Min-sum}})$ presents the arrival time at the last customer of the solution. Columns are ordered this way to better indicate the objective used for the particular pair of result.

For instances with up to 90 nodes, both objectives yield results with the same MA time. For most of those instances the CUAVRP cost of the solution is also the same. A 0.81% and 0.13% differences are observed in instances d9_k5_r200 and d8_k4_r300, respectively, with the Min-sum objective having the better results.

An interesting observation is that the Min-sum objective is able to obtain better MA times for many of the instances, while the Min-max objective cannot reach the CUAVRP cost values found with the Min-sum. This can be attributed to a combination of two factors. First, the cost structure of the CUAVRP penalizes routes with many customers, thus, it promotes shorter routes which is a goal closely related to the Min-max objective. Secondly, all nodes belonging to neighbouring cells are equally distant. Therefore, the MA time of a route largely depends on how many nodes it has. As a result, small routes obtained with the Min-sum objective are highly expected to have MA times, similar to those obtained with the Min-max objective.

Overall, the Min-sum objective was able to obtain 1.32% better results for the MA time metric and 1.50% better CUAVRP cost. For both objectives, minimizing the arrival time at the last grid cell and minimizing the sum of arrival times, the Min-sum objective is recommended for the CUAVRP.

6. Conclusions

In this paper, a Cumulative Unmanned Aerial Vehicle Routing Problem (CUAVRP) approach to optimize Humanitarian Coverage Path Planning was presented. The cost formulation of the CUAVRP is based on the Cumulative Capacitated Vehicle Routing Problem(CCVRP). Using a grid-based method of cellular decomposition, the Coverage Path Planning problem was transformed into a Vehicle Routing Problem. Three versions of a Parallel Weighted Greedy Randomized Adaptive Search Procedure (GRASP) - Variable Neighborhood Decent (VND) algorithm were implemented to effec-

Table 7
Comparison of results using Min-max and Min-sum objective for the CUAVRP.

Instance	n	k	$MA(Best_{\text{Min-max}})$	$Cost(Best_{\text{Min-max}})$	$Cost(Best_{\text{Min-sum}})$	$MA(Best_{\text{Min-sum}})$
d11_k6_r200	40	6	77.00	1676.23	1676.23	77.00
d11_k4_r200	40	4	110.00	2310.00	2310.00	110.00
d10_k3_r300	40	3	140.00	3010.00	3010.00	140.00
d10_k5_r200	40	5	90.00	1923.14	1923.14	90.00
d9_k8_r200	57	8	66.72	2120.38	2120.38	66.72
d9_k5_r200	57	5	108.00	3148.19	3122.74	108.00
d9_k4_r300	57	4	126.00	3780.00	3780.00	126.00
d8_k4_r300	68	4	139.31	4769.94	4763.31	139.31
d7_k4_r300	90	4	161.00	7247.90	7247.90	161.00
d6_k4_r300	109	4	164.48	9085.45	9076.97	168.00
d6_k12_r250	109	12	61.41	3608.07	3598.60	62.48
d5_k8_r300	159	8	106.21	8734.67	8386.90	97.07
d5_k4_r400	159	4	202.07	16132.84	16014.14	180.00
d5_k20_r250	159	20	50.81	4500.96	4464.93	53.28
d4_k5_r650	231	5	192.00	22064.11	21786.60	180.64
d4_k7_r300	231	7	139.31	15901.27	15789.80	128.97
d4_k15_r250	231	15	68.94	8333.96	8266.94	71.31
d3_k10_r300	405	10	131.48	26324.99	25836.18	130.41
d2_k15_r1200	1816	15	296.52	244297.97	240082.59	310.47
d1.9_k14_r1500	2032	14	334.20	306780.00	300986.79	330.10
d1.8_k15_r1500	2239	15	330.90	330404.37	326498.67	334.02
d9_k6_r800	112	6	178.45	10192.16	10092.02	166.61
d8_k12_r450	132	12	107.88	7859.87	7795.45	104.56
d8_k17_r450	132	17	101.82	7269.24	7266.57	106.45
d8_k5_r850	132	5	227.31	14779.72	14481.97	211.31
d7_k4_r1100	166	4	309.69	25332.77	24630.44	309.65
d6_k8_r850	217	8	180.00	19239.22	19201.21	183.94
d5_k10_r850	324	10	178.57	28844.53	28543.34	175.32
Total			4380.08	1139671.95	1122753.81	4322.61

tively solve it, each utilizing a different strategy of communication and information exchange between threads.

In “All Return” (AR) strategy, at each iteration, all worker threads return the improved solution and take a new solution from the main thread, in order to improve it. In “Best Continue” (BC) strategy, at each iteration, the worker thread with the best improved solution, continues the local search procedure on its solution. The rest of the worker threads are given a new solution from the main thread, in order to improve it. In the “Random Return” (RR) strategy, at each iteration, a random thread returns the improved solution and is given a new solution from the main thread. The rest of the worker threads continue their local search procedure with their solutions.

The performance of the algorithm was tested on the CCVRP, in instances solved by multiple algorithmic approaches in the literature. The performance, both in terms of solution quality and computational time, can be considered satisfactory. The algorithm with the AR strategy was able to reach the BKS in all CMT instances while the BC and RR strategies reached 6 out of 7 BKS. The gap in CMT5 instance, where the BKS was not reached, was minimal and comparable to other approaches in the literature.

For the CUAVRP, 28 instances were created based on CMT1 and CMT11. Using the coordinates of the customer nodes in the CMT instances, their convex hull was extracted. This convex polygonal area was the area of interest considered in the Humanitarian Coverage Path Planning problem. For different values of the UAV’s sensor viewing square dimension, the transformation of the HCPP problem into a CCVRP results in different grids, and thus, different graphs. With those graphs, for varying numbers of UAVs and maximum UAV ranges, 21 instances besed on CMT1 were created and solved and 7 instances based on CMT11 were created and solved. Instances range from 40 to 2239 nodes and UAVs from 3 to 20.

Both the Min-sum and Min-max objectives were tested and their results were compared in terms of the maximum arrival time at the last node and the CUAVRP cost function. The Min-sum objective was found to obtain overall better results for both metrics.

The topology of nodes in the CUAVRP instances play an important role to determine the best performing strategy of the implemented GRASP-VND. For the instance set with the majority of the nodes spread around the starting point, the AR is the best performer, especially in the instances with more than 230 nodes. For instances with the starting point placed near the border of the area of interest, the BC strategy with its increased exploitation is able to obtain the best results.

In terms of computational time, the first place with RR, which was classified second concerning the solution quality for both CUAVRP instance sets. As a conclusion, the Parallel Weighted GRASP-VND BC is suggested for solving the CUAVRP.

Modern CPUs have reached a threshold where single thread frequency is not expected to increase further. Instead manufacturers focus on increasing the number of cores/threads and on minimizing core-to-core latency. High-end professional CPUs (as of 2020) come with up to 64 cores (128 threads). Designing multithreaded algorithms will allow us to leverage on these new processors, therefore an interesting future work involves efficiently and effectively extending metaheuristic algorithms for mass parallel execution.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2021.09.008.

References

Aiex, R., Resende, M., Pardalos, P., & Toraldo, G. (2005). Grasp with path relinking for three-index assignment. *INFORMS Journal on Computing, 17*, 224–247.

Alciatore, D., & Miranda, R. (1995). A winding number and point-in-polygon algorithm. In *Glaxo virtual anatomy project research report, department of mechanical engineering, Colorado state university.*.

Alvarez-Valdes, R., Crespo, E., Tamarit, J., & Villa, F. (2008). Grasp and path relinking for project scheduling under partially renewable resources. *European Journal of Operational Research, 189*, 1153–1170.

Armentano, V. A., & de Frana, F. M. F. (2007). Minimizing total tardiness in parallel machine scheduling with setup times: An adaptive memory-based grasp approach. *European Journal of Operational Research, 183*, 100–114.

Avellar, G., Pereira, G., Pimenta, L., & Iscold, P. (2015). Multi-UAV routing for area coverage and remote sensing with minimum time. *Sensors, 15*, 27783–27803.
Babel, L. (2017). Curvature-constrained traveling salesman tours for aerial surveillance in scenarios with obstacles. *European Journal of Operational Research, 262*, 335–346.
Basilico, N., & Carpin, S. (2015). Deploying teams of heterogeneous UAVs in cooperative two-level surveillance missions. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 610–615).
Cabreira, T., Brisolara, L., & Ferreira, P. R., Jr. (2019). Survey on coverage path planning with unmanned aerial vehicles. *Drones, 3*, 4.
Campbell, A. M., Vandenbussche, D., & Hermann, W. (2008). Routing for relief efforts. *Transportation Science, 42*, 127–145.
Choset, H. (2001). Coverage for robotics—A survey of recent results. *Annals of Mathematics and Artificial Intelligence, 31*, 113–126.
Christofides, N., Mingozzi, A., & Toth, P. (1979a). *The vehicle routing problem*. London: John Wiley and Sons.
Christofides, N., Mingozzi, A., & Toth, P. (1979b). The vehicle routing problem. In *Proceedings of the combinatorial optimization* (pp. 315–338).
de Alcantara, A., Reinier Hovenburg, A., de Lima, N., Dahlin Rodin, L. C., Johansen, T. A., Storvold, R., et al. (2019). Autonomous unmanned aerial vehicles in search and rescue missions using real-time cooperative model predictive control. *Sensors, 19*, 4067.
Evers, L., Barros, A. I., Monsuur, H., & Wagelmans, A. (2014). Online stochastic UAV mission planning with time windows and time-sensitive targets. *European Journal of Operational Research, 238*, 348–362.
Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization, 6*, 109–133.
Festa, P., Pardalos, P., Resende, M., & Ribeiro, C. (2001). Grasp and VNS for max–cut. In *Extended abstracts of the fourth metaheuristics international conference* (pp. 371–376).
Galceran, E., & Carreras, M. (2013). A survey on coverage path planning for robotics. *Robotics and Autonomous Systems, 61*, 1258–1276.
Goerzen, C., Kong, Z., & Mettler, B. (2010). A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems, 57*, 65–100.
Graham, R. (1972). An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters, 1*, 132–133.
Guastella, D. C., Cantelli, L., Giammello, G., Melita, C. D., Spatino, G., & Muscato, G. (2019). Complete coverage path planning for aerial vehicle flocks deployed in outdoor environments. *Computers and Electrical Engineering, 75*, 189–201.
Ke, L., & Feng, Z. (2013). A two-phase metaheuristic for the cumulative capacitated vehicle routing problem. *Computers and Operations Research, 40*, 633–638.
Laguna, M., & Gonzalez-Velarde, J. (1991). A search heuristic for just-in-time scheduling in parallel machines. *Journal of Intelligent Manufacturing, 2*, 253–260.
Laguna, M., & Marti, R. (1999). Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing, 11*, 44–52.
Li, H., Chen, J., Wang, F., & Bai, M. (2021). Ground-vehicle and unmanned-aerial-vehicle routing problems from two-echelon scheme perspective: A review. *European Journal of Operational Research, 294*, 1078–1095.
Lottes, P., Khanna, R., Pfeifer, J., Siegwart, R., & Stachniss, C. (2017). UAV-based crop and weed classification for smart farming. In *2017 IEEE international conference on robotics and automation (ICRA)* (pp. 3024–3031).
Lysgaard, J., & Whlk, S. (2014). A branch-and-cut-and-price algorithm for the cumulative capacitated vehicle routing problem. *European Journal of Operational Research, 236*, 800–810.
Macrina, G., Di Puglia Pugliese, L., Guerriero, F., & Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies, 120*, 102762.
Marinakis, Y. (2012). Multiple phase neighborhood search-grasp for the capacitated vehicle routing problem. *Expert Systems with Applications, 39*, 6807–6815.
Marinakis, Y., Migdalas, A., & Pardalos, P. (2005). A hybrid genetic-grasp algorithm using Lagrangean relaxation for the traveling salesman problem. *Journal of Combinatorial Optimization, 10*, 311–326.
Maza, I., Caballero, F., Capitan, J., Martinez-de Dios, J. R., & Ollero, A. (2011). Experimental results in multi-UAV coordination for disaster management and civil security applications. *Journal of Intelligent and Robotic Systems, 61*, 563–585.
Mjirda, A., Todosijevic, R., Hanafi, S., Hansen, P., & Mladenovic, N. (2017). Sequential variable neighborhood descent variants: An empirical study on the traveling salesman problem. *International Transactions in Operational Research, 24*, 615–633.
Mladenovic, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24*, 1097–1100.
Nattero, C., Recchiuto, C., Sgorbissa, A., & Wanderlingh, F. (2014). Coverage algorithms for search and rescue with UAVdrones. In *Workshop of the XIII AIIA symposium on artificial intelligence*.
Ngueveu, S. U., Prins, C., & Wolfler Calvo, R. (2010). An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers and Operations Research, 37*, 1877–1885.
Nucamendi-Guilln, S., Angel-Bello, F., Martnez-Salazar, I., & Cordero-Franco, A. E. (2018). The cumulative capacitated vehicle routing problem: new formulations and iterated greedy algorithms. *Expert Systems with Applications, 113*, 315–327.
Pedemonte, M., Nesmachnow, S., & Cancela, H. (2011). A survey on parallel ant colony optimization. *Applied Soft Computing, 11*, 5181–5197.
Pham, H. X., La, M., Seifer, H. D. Feil-, & Deans, M. C. (2020). A distributed control framework of multiple unmanned aerial vehicles for dynamic wildfire tracking. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 50*, 1537–1548.
Resende, M. G. C., & Ribeiro, C. C. (2016). Parallel grasp heuristics. In *In optimization by GRASP: Greedy randomized adaptive search procedures* (pp. 205–227). New York: Springer.
Ribeiro, G., & Laporte, G. (2012). An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers and Operations Research, 39*, 728–735.
Rosalie, M., Dentler, J. E., Danoy, G., Bouvry, P., Kannan, S., Olivares-Mendez, M. A., et al. (2017). Area exploration with a swarm of UAVs combining deterministic chaotic ant colony mobility with position MPC. In *2017 international conference on unmanned aircraft systems (ICUAS), Miami, United States* (pp. 1392–1397).
Rudol, P., & Doherty, P. (2008). Human body detection and geolocalization for UAVsearch and rescue missions using color and thermal imagery. In *2008 IEEE aerospace conference (1–8). 2008 IEEE aerospace conference*.
Santamaria, E., Segor, F., Tchouchenkov, I., & Schnbein, R. (2013). Rapid aerial mapping with multiple heterogeneous unmanned vehicles. *International Journal on Advances in Systems and Measurements, 6*, 384–393.
Sze, J. F., Salhi, S., & Wassan, N. (2017). The cumulative capacitated vehicle routing problem with min-sum and min-max objectives: An effective hybridisation of adaptive variable neighbourhood search and large neighbourhood search. *Transportation Research Part B: Methodological, 101*, 162–184.
Valente, J., Sanz, D., Cerro, J., Barrientos, A., & de Frutos, M. (2013). Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precision Agriculture, 14*, 115–132.