electronics

MDPI

Article

UAVs Path Planning under a Bi-Objective Optimization Framework for Smart Cities

Subrata Saha [1], Alex Elkjær Vasegaard [1], Izabela Nielsen [1,*], Aneta Hapka [2], Henryk Budzisz [2]

[1] Department of Materials and Production, Aalborg University, Fibigerstræde 16, DK 9220 Aalborg, Denmark; saha@m-tech.aau.dk (S.S.); aev@mp.aau.dk (A.E.V.)
[2] Faculty of Electronics and Computer Science, Koszalin University of Technology, 75-343 Koszalin, Poland; aneta.hapka@tu.koszalin.pl (A.H.); henryk.budzisz@tu.koszalin.pl (H.B.)
* Correspondence: izabela@mp.aau.dk

Abstract: Unmanned aerial vehicles (UAVs) have been used extensively for search and rescue operations, surveillance, disaster monitoring, attacking terrorists, etc. due to their growing advantages of low-cost, high maneuverability, and easy deployability. This study proposes a mixed-integer programming model under a multi-objective optimization framework to design trajectories that enable a set of UAVs to execute surveillance tasks. The first objective maximizes the cumulative probability of target detection to aim for mission planning success. The second objective ensures minimization of cumulative path length to provide a higher resource utilization goal. A two-step variable neighborhood search (VNS) algorithm is offered, which addresses the combinatorial optimization issue for determining the near-optimal sequence for cell visiting to reach the target. Numerical experiments and simulation results are evaluated in numerous benchmark instances. Results demonstrate that the proposed approach can favorably support practical deployability purposes.

Keywords: unmanned aerial vehicles (UAVs); multi-objective optimization; integer programming; GLPK; variable neighborhood search; search and rescue

Citation: Saha, S.; Vasegaard, A.I.; Nielsen, I.; Hapka, A.; Budzisz, H. UAVs Path Planning under a Bi-Objective Optimization Framework for Smart Cities. *Electronics* **2021**, *10*, 1193. https://doi.org/10.3390/electronics10101193

Academic Editors: Juan M. Corchado, Josep L. Larriba-Pey, Pablo Chamoso and Fernando De la Prieta

Received: 16 March 2021
Accepted: 13 May 2021
Published: 17 May 2021

1. Introduction

The path planning problem for a set of Unmanned Aerial Vehicles (UAVs) has gained unprecedented interest from researchers and practitioners to develop intelligent systems and execute various tasks with minimum human intervention. With upgraded components such as cameras, sensors, or telemetry systems, UAV application is becoming an integral strategic part for emergency management; aerial photography; mountain rescue; smart farming; maritime search and rescue; information collection, post-disaster relief; homeland security, crowd management, etc. [1–3]. UAVs, in practice, has many significant advantages such as human workload reduction, high mobility, saving of valuable resources, etc. In the literature, the path planning problem is categorized in several ways according to problem characteristics. For example, according to the targets' reaction, one can classify the problem into two categories: one-sided vs. two-sided path planning problems. On the other hand, based on targets' motion, one can classify the situation as static vs. moving target search or open vs. closed-loop decision models based on the decision-making context [4–10].

In recent years, the utilization of UAVs has been becoming increasingly attractive in the context of Smart City Management solutions. Several key technologies are continuously integrated into smart cities operations, such as data collection and protection and intrusion detection technologies. In this regard, the application of UAVs to collect data or images is an economical and effective solution. UAVs operations can lead to a new paradigm for developing smart cities with a high-quality life and sustainable economic growth. For example, Felemban et al. [11] noted that UAVs could be used to detect the earlier signs of a stampede, congestion, and other crowd problems. The authors proposed a Priority-Based Routing Framework to increase the delivery speed of images during Hajj in Saudi

Arabia. Researchers found that UAVs can be helpful in policing systems to fight against crime [12]. It was reported that such UAV policing systems work well for extensive crime deterrence [13]. However, there are many challenges, and we highlight one of those where UAVs are deployed in search and rescue problems.

Due to the sequential decision-making nature, the fundamental search and rescue path planning problem is a non-deterministic polynomial-time problem (NP-hard) [14]. Therefore, researchers employ both exact algorithm and heuristic approaches alternatively to solve such complex decision-making problems. One can argue that the modern search theory originated from the pioneering works by the group of researchers, Stewart [8], Brown [15] and Benkoski et al. [16]. Researchers mainly focus on the allocation decision instead of the optimal sequential path generation. By assuming an exponential detection function, Stewart [8] formulated a network flow model to characterize a moving target detection problem and used the branch and bound method to find a near-optimal solution. Later, Eagle [4] formulated the model in a dynamic programming framework and utilized the Markov process to replicate target motion as state transitions. Washburn [17] made an effort to determine the best upper bound for a generalized path planning problem. After that, researchers progressively shifted their attention toward the evaluation of algorithm performance in more complex enshrinement [18]. However, the travel time in the earlier model was assumed as uniform. Lau et al. [19] relaxed this assumption and formulated a model where travel time among regions are non-uniform. Rogge and Aeyels [20] introduced the concept of a collaborative path planning problem where the search area consists of multiple moving targets with an arbitrary number of obstacles. Li et al. [21] studied energy-efficient rechargeable UAV deployment strategy to provide seamless coverage in urban areas and employed the two-stage particle swarm optimization (PSO) algorithm to solve the problem. Regarding other variants, Berger and Lo [22] introduced a mixed-integer programming model under a directed acyclic graph framework and used CPLEX software to find an optimal path. To overcome computational effort, Perez-Carabaza et al. [23] proposed a modified ant colony optimization (ACO) algorithm to investigate the nature of trajectories for a set of heterogeneous UAVs. Ye et al. [24] used an adaptive genetic algorithm (GA) to find the solution for a collaborative multiple task assignment problem with fixed-wing UAVs. The authors employed a robust encoding strategy to generate feasible chromosomes. Lu et al. [25] use the wolf pack algorithm (WPA) to solve the task assignment problem for UAVs. The authors found that WPA can outperform PSO and GA in terms of convergence speed and solution accuracy. Lou et al. [26] proposed a multi-swarm fruit fly optimization algorithm to find a solution for multi-UAV cooperative mission planning problem. However, Alhaqbani et al. [27] stated that a common problem in most of the metaheuristics is that those can perform poorly in regards to run time. More recently, Xiong et al. [28] introduced Voronoi-based Ant colony optimization algorithm combined with the Dijkstra's algorithm to investigate optimal trajectories. In recent years, various types of machine learning algorithms have been employed to obtain optimal deployment strategy, and we refer to the recent review works by [29] and [30] for detailed discussion in this aspect. In addition, we refer the following works for more discussion on path planning from various perspectives [31–37].

In this study, we use a modified Variable Neighborhood Search (VNS) meta-heuristic [38]. Since its inception, the algorithm has been employed in numerous fields such as network design problems in communication [39], facility location problem [40], data mining [41], timetabling and related manpower organization problems [42], single- and multi-objective job shop scheduling [43,44], vehicle routing problem [45] and bioinformatics [46] due to its user-friendliness, higher precision and robustness. The VNS systematically exploits the idea of neighborhood change iteratively to improve the initial solution inside the shaking and local search procedures [47,48]. Unlike other meta-heuristic approaches, parameter tuning is always an issue; the fundamental VNS algorithm and its extension version require few or, occasionally, no parameters. One significant advantage to the VNS-based approach for path planning is that it accommodates the path maneuverability through the path

constructor (see Algorithm 1) operator. At the same time, the inherent shaking procedure seeks to overcome the possible local optima. The algorithm then attempts to improve the randomly changed path to catch a more rewarded path than the incumbent solution.

The cited literature's main disadvantage is that most authors only studied the problem as a single-objective optimization problem, e.g., maximizing the probability of finding targets, minimizing the path length, equal utilization of resources, etc. However, in a time-constrained decision-making context, only considering one objective may not lead to an acceptable outcome [49,50]. From a practical point of view, it is essential to handle several objectives simultaneously to obtain a pragmatic solution. Explicitly, the two most fundamental goals that need to be considered are maximization of finding the targets and minimizing the path length objective that can ensure minimum utilization of resources and implicitly ensure less operational time and energy consumption. It is challenging to find the ideal solution due to the conflicting nature of objective functions; therefore, researchers have proposed different approaches such as weighted sum [51], global criterion [52], goal programming [53], multi-choice goal programming [54], non-dominated sorting genetic algorithm II [55], fuzzy-two phase approach [56], etc., and the issue of a specific method largely depends on the decision-makers. Note that UAV path planning is itself an NP-hard problem [57]; thus, we use a simple weighted sum approach in this study. This study formulated the model as binary linear programming (BLP) formulation under a bi-objective optimization environment and proposed a modified VNS algorithm to find the solution. Numerical experiments were conducted to validate the overall framework. The key contributions of the study are as follows: First, a bi-objective optimization problem is proposed to obtain paths for multiple UAVs in a time-constrained environment. Second, a modified VNS algorithm is proposed, which is highly parallelizable and straightforward to understand. Moreover, the simulation study reveals that it can provide a solution within a reasonable time when the exact solver fails to provide a solution, and the performance for the algorithm is always higher compared to Dijkstra's algorithm, which is extensively used by several researchers [58,59]. Finally, a sensitivity analysis on the weight-space provide an overview regarding the importance of multi-objective formulation in the practical implementation of UAVs.

The paper is organized as follows. The mathematical model and corresponding assumption and notation are presented in Section 2. In Section 3, an overview is presented for the data generation. The solution procedure for the model is described in Section 4. A detailed overview of the VNS algorithm is also presented in this section. Extensive numerical experiments and validation of the proposed solution framework's effectiveness are presented in Section 5. Finally, Section 6 concludes by highlighting findings, limitations and future research directions.

2. Mathematical Model

Path planning and trajectory mapping for UAV is an important topic because of the incredible versatility and flexibility of UAVs that allow them to be employed in different operations. Although path planning goes before trajectory mapping, fundamentally, their characteristics are not entirely distinct. If point-to-point trajectories are measured, the two problem needs to be solved simultaneously if the initial and final positions are specified. One can define the path planning problem as finding a collision-free motion within a specified environment where initial and final locations are pre-defined. In this study, we use the cell decomposition method. In this method, the entire search space is subdivided into several regions (equal/unequal), called cells. The corresponding path will represent a connected graph and describe the adjacent relations between cells. Simultaneously, the trajectory planning problem is based on the input generated by the path planner. To plan a trajectory, commonly, a sequence of waypoints needs to be extracted. A kinematic inversion needs to be performed based on some decision-maker criteria such as minimizing total execution time, energy, distance, jerk, etc. In the present formulation, we ignore the effect of the kinematics of the UAV. We assume that a team of homogeneous UAVs is searching stationary targets in a pre-defined search region [60].

The search area is divided into an $N \times N$ grid describing possible target locations. The time duration for each cell visit, with equal size, is assumed as constant. The cell occupancy probabilities are generated initially, and, as we assume the targets to be stationary and non-moving, we omit the dynamics of a changing probability map. To maneuver its neighboring cells, any UAV can move in eight different directions{E, W, N, S, SE, SW, NE, NW}. However, at the cell where the UAVs start maneuvering is located, the UAVs are also allowed to hover. This mimics the possibility of early landing or later departure for some UAVs. A graph theory-based directed acyclic network representation is employed to streamline the setup. The entire graph is defined as $G_t = (V_t, E_t)$ for all t in a given time horizon T, V_t, the set of vertices, represent all possible locations $n \in N^* = \{1, \ldots, N^2 - 1, N^2\}$ at time $t \in T$. E_t, the set of edges, represents all the possible state transition related to each UAV between episodes t and $t + 1$. An adjacency matrix A defines the connectivity of G, $A_{tn'n} = 1$ if $v_{tn'} \in V_t$ and $v_{t'n} \in V_{t+1}$ are connected, else $A_{tn'n} = 0$. Consequently, a binary decision variable x_{ntr} is introduced to represent the cells n traversed at the respective time period t for the respective rth UAV.

The following notations are used to formulate the mathematical model:

N	the entire search region is divided into $N \times N$ number of cells with equal area in the grid, $n \in N^* = \{1, .., N^2 - 1, N^2\}$
T	set of time intervals with equal length defining the time horizon to explore a grid, $t \in \{0, 1, \ldots, \lvert T\rvert - 1\}$
R	number of UAVs, $r \in \{1, \ldots, R\}$
p_n	probability of actual target occupancy on cell n
x_{ntr}	state transition binary variable; $x_{ntr} = 1$, if the path of rth UAV investigates the nth cell in time period t, while $x_{ntr} = 0$, if that the corresponding cell is not visited
$F_{n'ntr}$	a binary matrix representation of the infeasible maneuvers. That is, $F_{n'ntr} = 1$ whenever $A_{tn'n} = 0$
Z_{ntr}	a binary binary matrix representation of all cells through the time horizon representing the same location
B_{ntr}	a binary matrix representation of the cells that can only be visited once
H_{ntr}	a binary matrix representation of all maneuvers performed in the time period t
S_{ntr}	a binary matrix representation of start and ending positions for rth UAV

Based on the above notation, the following mathematical model is proposed, where the first objective represents the cumulative probability of success for the total number of UAVs to be deployed and the second objective minimizes the total spent time performing the mission:

$$\max \quad f_1 = \sum_{r \in R} \sum_{t \in T} \sum_{n \in N^*} p_n x_{ntr} \tag{1}$$

$$\min \quad f_2 = \sum_{r \in R} \sum_{t \in T} \sum_{n \in N^*} \frac{x_{ntr}}{RT} \tag{2}$$

s.t.

$$\sum_{t \in T} \sum_{n \in N^*} F_{n'ntr} x_{ntr} \leq 1 \quad \forall r \in R \quad \forall n' \in N^* \tag{3}$$

Constraint (3) ensures that infeasible maneuvers cannot be performed between two consecutive time periods. The binary matrix F showcases each pair between consecutive cells n and n' that are infeasible for a given time period t. That is, if $F_{n'ntr} = 1$, then the two cells n and n' in time period t and $t + 1$, respectively, are not feasible in the same path for any r.

$$\sum_{r \in R} Z_{ntr} x_{ntr} \leq 1 \quad \forall n \in N^* \quad t \in T \tag{4}$$

Constraint (4) enforces a safety zone around each path, that is, a single agent r can only traverse a cell in a given time period. Note that the binary matrix Z showcases the decision variable's index that represents the same time period.

$$\sum_{r \in R} \sum_{t \in T} B_{ntr} x_{ntr} \leq 1 \quad \forall n \in N^* \tag{5}$$

Here, constraint (5) considers gathering images of a cell over multiple different time periods, where the binary B matrix showcase each index that represents the same cell. In

this paper, we neglect the dynamics of changing probability, and we are not interested in obtaining a search path that acquires multiple images of the same cell. Note that we do not have to consider a conditional probability map that is dependent on the chosen paths because of this constraint, as the cumulative probability will be in the range of $[0, 1]$.

$$\sum_{n \in N^2} H_{ntr} x_{ntr} = 1 \quad \forall n \in N^* \quad r \in R \tag{6}$$

In constraint (6), the binary H matrix ensures that the paths only allow a single maneuver to be performed per time period per UAV.

$$\sum_{r \in R} \sum_{t \in T} S_{ntr} x_{ntr} = 1 \quad \forall r \in R \tag{7}$$

Constraint (7) ensures that the complete path starts and ends in the designated time zones in the designated time periods.

$$x_{ntr} \in \{0, 1\} \quad \forall n \in N^*, \forall t \in T, r \in \{1, \ldots, R\} \tag{8}$$

Finally, the above constraint (8) represent the decision and auxiliary variables.

3. Scenario Generation

The UAV-assisted SAR mission generally consists of multiple different phases, with the common goal of deploying as soon as possible when sufficient information about the mission is gathered. The UAV aspect is to either aid or collect information as fast as possible for the rescue team's job. In this research, the UAVs are only gathering information through images. Therefore, when generating the problem scenarios, we have to assume some information that later can be modified to accommodate real-world scenario. In general, the overall map is divided into an $N \times N$ grid where each cell is assumed to have the same area. Then, a probability map is generated where each cell is given a certain probability of containing the missing target. The probability map is generated randomly based on a given number of hotspots and corresponding spread (see Figure 1). To accommodate the problem scenario, the number of deployed UAVs also affects the size of the problem scenario. These are assumed to be taking off and landing in a specific grid cell. There is also denoted a time horizon with a given number of equidistant points in time, and the UAVs are then able to search an entire grid cell for each time period, and then go to one of their neighboring grid cells in the following time period. As mentioned in the Mathematical Modelling Section, the UAVs can move in all directions, but they can only hover (land) in the grid cell containing the UAV station. Note that this cell, therefore, should not have any gain or loss in terms of the objectives, e.g., probability of locating the target. Due to the problem complexity, we assume there to only be two hot spots with a spread of three and the UAV station to be located in grid cell $[0, 0]$. The parameters assumed to affect the size of the problem scenario are the grid size, N, time horizon, T, and number of UAVs, R.

Note that the proposed division of the search area is analogous to the raster model, which is a data storage method used extensively in geographic information systems.

4. Solution Procedure

In this section, we explain the solution procedure and the selection of search parameters for the employed search method. The exact approach is often not applicable in large-scale scenarios, as it can even fail to deliver a feasible solution. In a time-restricted environment such as UAV-assisted search and rescue, this is not applicable. On the other side of the spectrum, a greedy approach does deliver a feasible solution, but it often lacks in performance. This is what we try to investigate with the deployed VNS approach. We evaluate the performance of the algorithm with Dijkstra's algorithm and exact solvers such as GNU Linear Programming Kit (GLPK) to establish its efficiency. However, before doing so, the following definitions should be presented.

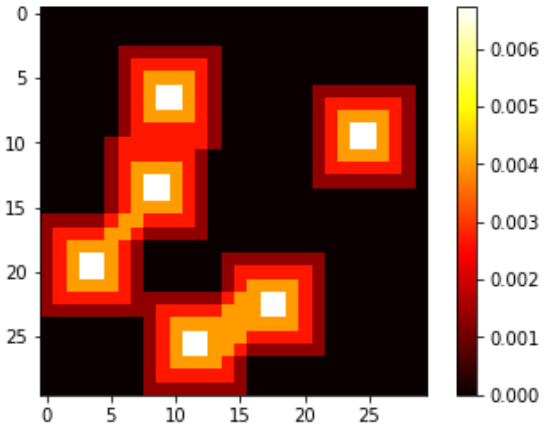

Figure 1. Probability map on a 30 × 30 grid given six hotspots with a spread of 3. The start and end cell (UAV station) is located in cell $[0, 0]$ in the upper left corner.

Definition 1. *Multiple objective optimization problems can be represented as follows:*

$$\begin{cases} max & (f_1(x), f_2(x), \ldots, f_k(x)) \\ min & (g_1(x), g_2(x), \ldots, g_r(x)) \\ s.t. & x \in X = \{x \mid h_t(x) \leq 0, t = 1, \ldots, m\} \end{cases}$$

where $x = (x_1, x_2, \ldots, x_n)$ *are the decision variables;* $f_i(x), (i = 1, \ldots, k)$ *are maximization type objective functions;* $g_j(x), (j = 1, \ldots, r)$ *are minimization type objective function;* $h_t(x)$, $(t = 1, \ldots, m)$ *are set of constraints [60].*

Definition 2. *A decision plan* $x^0 \in X$ *is said to be a Pareto optimal solution to the multiple objective optimization problems if there does not exist another* $y \in X$*, such that* $f_k(y) \leq f_k(x^0)$ *for all k and* $f_s(y) < f_s(x^0)$ *for at least one s Wu et al. [61].*

From the perspective of the search and rescue problem, it is difficult to define the strict upper or lower bounds for the multi-objective setting problem. This is first because of the fuzzy nature of the multi-objective setting but also because of the complexity of obtaining a solution. Therefore, we incorporate both exact and inexact solution approachs to illustrate these issues.

4.1. Transforming Multi-Objective Framework into a Single-Objective One

When dealing with a multi-objective framework, several types of solution approaches can be applied, such as transforming the problem into a single-objective one, incorporating them through a lexicographic method, identifying the entire Pareto front to determine the trade-off among objective weightings, etc. Therefore, it generally comes down to whether the decision maker's preference is incorporated before, under or after exploring the solution space.

In a time-restricted environment such as search and rescue mission planning, it is of absolute necessity that a solution can be obtained in real-time. Therefore, we utilize the approach to transform the multi-objective framework into a single objective. For the bi-objective framework, the objectives do not have a fitting cost transform due to the respective units of the objectives. However, there is a range similarity in terms of the sum of them being between 0 and 1; a simple weighted average is, therefore, fitting to do this. Here, α represents the trade-off between the objectives [62].

$$f_{combined} = \alpha f_1 + (1 - \alpha) f_2 \tag{9}$$

Note that the naive weighted average can be controversial, and we therefore elaborate the use of this in Section 5 (for more information see, Wang [29]).

4.2. GLPK

We utilized the freely available GNU Linear Programming Kit (GLPK) package for the exact solution procedure. The GLPK package is used for large-scale mixed-integer linear programming problems [63]. It utilizes the branch-and-cut method for integer restriction of the decision variables, extending to the branch-and-bound and cutting plane method. The package is implemented in Python, where a maximum solution time is set to 12 min. In a general real-world setting, the ultimately allowed solution time in practice is likely to be lower, and this limit is therefore only set for illustrative purposes.

4.3. Dijkstra's Algorithm

A useful path can be established by implementing graph searching algorithms. In this direction, we utilize Dijkstra's Algorithm, which is extensively used in single-source shortest path problems with non-negative weights for each edge. In implementing the Dijkstra's algorithm for the path finding problem, it is imperative to introduce the constraint on revisiting nodes that represent the same location in different time periods. A way to incorporate this is when visiting the node (i.e., that node being the lowest distance in the queue), then not allowing it to go back after a defined safety period has passed. The set of nodes is then removed in the same way as the visiting node is removed from the queue. Here, the distance that is sought to be minimized is the cumulative score, while the graph traversed is the directed graph G, not allowing it to go backward in time. We refer to the works of Yuan et al. [58] and Sathyara et al. [59] for the detail overview of the algorithm.

4.4. Variable Neighborhood Search

The inexact solution procedure developed in this research is a two-step VNS method that incorporates the general approaches of the VNS but couples that with the known information of directed acyclic graph of feasible paths through a path construction algorithm. The general VNS is proposed by Mladenovic and Hansen [38] in 1997, and it represents a flexible framework for building heuristics to approximately solve combinatorial and non-linear optimization problems. The VNS search heuristic systematically changes its neighborhood structures to obtain a solution. It does so based on the following key observations [64]:

- A local optimum relative to one neighborhood structure is not necessarily a local optimum for another neighborhood structure.
- A global optimum is a local optimum concerning all neighborhood structures.
- Empirical evidence shows that all or a large majority of the local optima are relatively close to each other for many problems.

The ingredients of a variable neighborhood search heuristic include an improvement phase used to improve a given solution and a so-called shaking phase used to resolve local minima traps. The improvement phase, the shaking procedure and the neighborhood change step are executed alternately until a predefined stopping criterion. This research combined it with a path construct algorithm to obtain feasible solutions more quickly and ensure that it follows the stated constraints. The path construct algorithm can be found in the pseudo-code of Algorithm 1. This approach linearly goes through the available time horizon and selects the next maneuver through a weighted probability based on each alternative's respective score. It accompanies the constraint by removing feasible maneuvers and steers it back to the end position by narrowing the feasible maneuvers based on the Chebyshev and Manhattan distances to the end position. Note that this feature of steering the path back to the selected end position is necessary as the two-step VNS randomly selects new neighborhoods to investigate. The grid representation is, therefore, not enough to steer it back. The integrated VNS approach selects a random neighborhood to improve upon the path. It stops selecting new neighborhoods when a designated number of iteration have been investigated. The pseudocode of the algorithm is presented in Algorithms 1 and 2.

Algorithm 1: The path constructing algorithm: **path_constructor(x_original, t1, t2, rs, score)**

```
Result: feasible path in x
x_original := the path that should be updated
N, R, T := the dimensions of the problem (grid size, number of UAVs, size of time horizon)
t1, t2 := the start and end time where a new path between should be located
rs := the set of UAVs
x_new := x_original
x_new[:,:,[t1:t2]] := 0
for r in rs do
    start = starting position
    end = end position
    if start, end is empty then
        start := global start
        end := global end
    end
    t := t1 + 1
    infeasible_neighbor := []
    infeasible_neighbor_T := []
    while t < t2 do
        prior := cell of t-1 maneuver
        feasible_maneuver := all maneuvers inside grid
        /* remove infeasible maneuvers                                          */
        feasible_maneuvers := remove infeasible manuevers as indicated by
          infeasible_neighbor if t is in infeasible_neighbor_T
        if Manhatten distance from prior to end >= t2 - (t-1) then
            maneuver_distance = manhatten distance from possible neighbors to end + 1
            if maneuver_distance <= t2-t then
                remove maneuver from feasible_maneuvers
            end
        end
        if chebyshev distance from prior to end >= t2 - (t-1) then
            maneuver_distance = chebyshev distance from possible neighbors to end + 1
            if maneuver_distance <= t2-t then
                remove maneuver from feasible_maneuvers
            end
        end
        if maneuver in feasible_maneuvers has already been visited by other UAVs then
            remove maneuver from feasible_maneuvers
        end
        if feasible_maneuvers is empty then
            if t-1 is equal to t1 then
                RETURN(x_original)
            end
            else
                Add prior manuever to infeasible_neighbor
                Add t-1 to infeasible_neighbor_T
                remove prior from x_original
                t := t-1
                break
            end
        end
        /* select feasible maneuver based on weighted probability              */
        ranked_maneuvers := ARGSORT(score[feasible_manuevers])
        weighted_maneuvers := ranked_maneuvers / SUM(ranked_maneuvers)
        chosen := CHOOSE(feasible_manuevers, weighted_maneuvers, 1)
        x_new[chosen] = 1
    end
end
RETURN(x_new)
```

Algorithm 2: Pseudocode representing **VNS(score, N, R, T, neighborhood_size, nmax, kmax, tmax)**

```
h! Result: best path P
score := score for each cell
N, R, T := the dimensions of the problem (grid size, number of UAVs, size of time horizon)
neighborhood_size := size of searched neighborhood
nmax := maximum number of neighborhood changes
kmax := maximum searches per neighborhood
tmax := total maximum runtime in seconds
n := 0
x_best := path_constructor(ZEROS(N,R,T), t1=0, t2=end, rs=[0,1], score)
score_best := SUM(x_best * score)
while n<nmax do
    k := 0
    while k<kmax do
        n1 := RANDOM(0,T)
        n2 := min(T,n1+neighborhood_size)
        nr := RANDOM(0,R)
        x_temp := path_constructor(x_best, t1=n1, t2=n2, rs=nr, score)
        score_temp := SUM(x_best * score)
        if score_temp > score_best then
            x_best := x_temp
            score_best := score_temp
            k := 0
        end
        k := k+1
    end
    n := n+1
end
RETURN(x_best)
```

5. Experiments

All numerical experiments were executed with Intel Core i5-8250 CPU with 1.60 GHz processors and 8.00 GB RAM for performance evaluation. For numerical verification, we model the probability map through two hotspots with a spread of two cells.

5.1. Sensitivity of VNS Parameters

The VNS algorithm has three different parameters indicating the search depth, i.e., neighborhood, $nmax$ and $kmax$, defining the size of the neighborhood each search considers; the maximum number of searched neighborhoods; and the number of searches per neighborhood. The results are shown in Table 1.

Table 1. Average performance, standard deviation and average runtime for 100 different runs with different neighborhood parameter.

Neighborhood Parameter	Relative Performance ($f_{combined}$)	Standard Deviation ($f_{combined}$)	Runtime (s)
0.250	0.630	0.086	13.239
0.333	0.820	0.033	20.094
0.417	0.837	0.036	27.870
0.500	0.860	0.025	35.917
0.583	0.908	0.027	39.621
0.667	0.921	0.027	42.898
0.750	0.893	0.043	47.320
0.833	0.862	0.035	49.739

The performance in Table 2 illustrates the change in deviation and runtime when modifying the *nmax* and *kmax* parameter, but it should be noted that the computation of these could easily be parallelized. In the parameter indicating the neighborhood's size, we can see that there is not a unified result showing which size of a neighborhood to chose. Therefore, we choose to further extend the algorithm by randomly selecting a length within the range of 0.3 to 0.9 for each neighborhood change. This furthers the shake and improvement steps of the VNS, as both local and global solutions will be investigated.

Table 2. Average relative performance of the Variable Neighborhood Search (VNS) method compared to the exact GNU Linear Programming Kit (GLPK) approach for 100 different runs with different nmax and kmax settings.

	Avg. Performance (Relative to GLPK)						**Avg. Runtime (In Seconds)**					
nmax\kmax	**5**	**15**	**25**	**35**	**45**	**55**	**5**	**15**	**25**	**35**	**45**	**55**
50	0.778	0.844	0.847	0.865	0.907	0.889	1.279	3.634	6.006	8.895	10.885	13.773
100	0.777	0.865	0.889	0.885	0.885	0.847	2.335	7.114	11.527	16.675	22.014	26.318
150	0.931	0.890	0.870	0.931	0.950	0.933	3.662	11.177	16.975	24.241	32.656	37.795
200	0.926	0.931	0.843	0.823	0.864	0.912	5.356	14.136	23.159	32.099	42.245	54.182
250	0.867	0.779	0.911	0.933	0.891	0.865	6.185	16.359	28.851	39.850	55.093	62.888
500	0.932	0.867	0.913	0.911	0.869	0.975	12.262	33.660	59.775	87.177	102.260	126.429
1000	0.934	0.912	0.910	0.868	0.871	0.932	23.976	70.143	111.092	156.020	211.473	269.177
1500	0.886	0.846	0.928	0.928	0.867	0.913	32.441	103.338	171.001	248.853	323.948	360.003
2000	0.927	0.849	0.911	0.912	0.956	0.912	45.564	133.198	229.218	338.928	360.003	360.007
2500	0.869	0.976	0.911	0.912	0.974	0.912	61.237	175.410	298.774	360.004	360.002	360.002

5.2. Performance and Runtime for VNS, Dijkstra, and GLPK

GLPK is an exact approach and is therefore significantly slower, but it also yields the optimal solution. However, the GLPK is not able to solve any of the larger problem scenarios. The performance and runtime for the three approaches on different scenario sizes relative to grid size N, time horizon T and the number of UAVs R can be seen in Figure 2. Note that, when a solution approach reaches the time limit, the time is noted, while its performance is not.

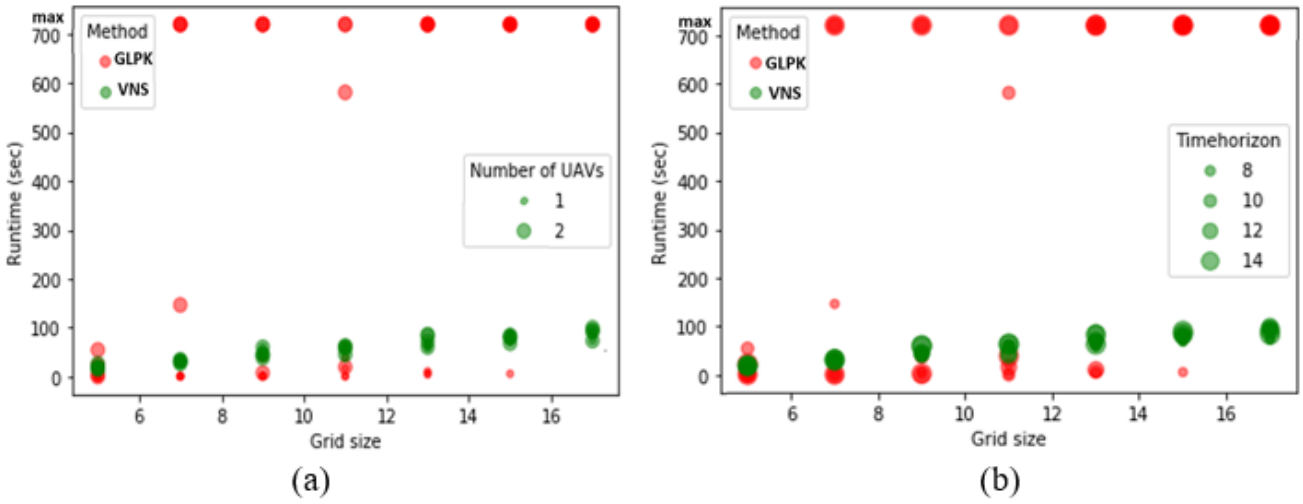

Figure 2. (**a**) The relative performance of the Variable Neighborhood Search (VNS) and Dijkstra algorithm compared to the optimal solution found by the GNU Linear Programming Kit (GLPK) approach is shown. Note that many experiments do not yield a relative performance as GLPK could not obtain a solution. (**b**) The runtime for the three approaches is presented, demonstrating relation to different grid sizes and time horizons. The exact value of performance measures is presented in Table 3.

Table 3. The performance of the respective solution approaches on different scenarios. Note that GLPK could not obtain a solution on some of the scenarios. This is illustrated by (-), while its runtime reached the limit of 720 s.

Grid Size	Time Horizon	No. of UAVs	Performance ($f_{combined}$)			Relative Performance	
N	T	R	VNS	Dijkstra	GLPK	$\frac{f_{\|VNS}}{f_{\|GLPK}}$	$\frac{f_{\|Dijkstra}}{f_{\|GLPK}}$
5	10	1	0.225	0.018	0.242	0.928	0.074
5	10	2	0.137	0.324	0.416	0.329	0.778
5	14	1	0.310	0.206	0.357	0.868	0.576
5	14	2	0.454	0.124	0.454	1.000	0.273
5	18	1	0.332	0.199	0.484	0.686	0.412
5	18	2	0.371	0.172	0.428	0.865	0.406
5	22	1	0.460	0.128	0.460	1.000	0.280
5	22	2	0.400	0.185	0.481	0.830	0.384
7	10	1	0.075	0.046	0.102	0.730	0.450
7	10	2	0.082	0.019	0.135	0.612	0.143
7	14	1	0.163	0.037	0.178	0.912	0.208
7	14	2	0.119	0.045	0.143	0.831	0.316
7	18	1	0.366	0.140	0.385	0.951	0.364
7	18	2	0.214	0.061	-	-	-
7	22	1	0.554	0.005	0.554	1.000	0.009
7	22	2	0.344	0.016	-	-	-
9	10	1	0.184	0.057	0.222	0.832	0.258
9	10	2	0.319	0.188	0.344	0.927	0.546
9	14	1	0.096	0.035	0.101	0.947	0.351
9	14	2	0.026	0.011	-	-	-
9	18	1	0.346	0.244	0.371	0.932	0.656
9	18	2	0.431	0.003	-	-	-
9	22	1	0.446	0.237	0.496	0.899	0.479
9	22	2	0.458	0.300	-	-	-
11	10	1	0.185	0.007	0.185	1.000	0.042
11	10	2	0.283	0.099	0.296	0.958	0.336
11	14	1	0.273	0.042	0.297	0.916	0.141
11	14	2	0.408	0.162	0.427	0.955	0.380
11	18	1	0.242	0.029	0.297	0.812	0.098
11	18	2	0.272	0.003	-	-	-
11	22	1	0.447	0.191	0.447	0.999	0.428
11	22	2	0.545	0.002	-	-	-
13	10	1	0.097	0.056	0.111	0.876	0.504
13	10	2	0.110	0.027	-	-	-
13	14	1	0.223	0.068	0.248	0.899	0.273
13	14	2	0.347	0.274	-	-	-
13	18	1	0.348	0.002	0.348	1.000	0.008
13	18	2	0.522	0.005	-	-	-
13	22	1	0.373	0.089	-	-	-
13	22	2	0.547	0.047	-	-	-
15	10	1	0.111	0.042	0.111	0.999	0.384
15	10	2	0.085	0.028	-	-	-
15	14	1	0.095	0.009			
15	14	2	0.039	0.028	-	-	-
15	18	1	0.072	0.043	-	-	-
15	18	2	0.112	0.039	-	-	-
15	22	1	0.092	0.004	-	-	-
15	22	2	0.107	0.031	-	-	-
17	10	1	0.010	0.008			
17	10	2	0.024	0.081	-	-	-
17	14	1	0.492	0.068	-	-	-
17	14	2	0.573	0.355	-	-	-
17	18	1	0.372	0.163	-	-	- -
17	18	2	0.448	0.003	-	-	-
17	22	1	0.249	0.001	-	-	-
17	22	2	0.141	0.033	-	-	-

The performance clearly indicates that the GLPK is generally faster for small problem scenarios with a single UAV. However, it cannot even obtain a solution whenever there are two UAVs to consider or the grid size or time horizon is larger. The relative performance of VNS indicates that, for larger problem scenarios, it will perform within 20% of the optimal

solution, while, for smaller problem scenarios, it performs within 50% of the optimal. The latter is perhaps because VNS searches with a neighborhood size that is too small relative to the grid size, so it will never get out of the local optima. However, it does not seem to be an issue for the larger problem scenarios. Similarly, the Dijkstra approach seems to decrease in performance relative to the exact approach when the scenario size increases. This is probably due to the greedy nature of the method, as it does not want to investigate areas that require it to cross a section of cells without any probability of success. The results also showcase the complexity of the large-scale problem scenarios in UAV-assisted search and rescue missions. Overall, Figure 2 demonstrates that the VNS outperforms GLPK and Dijkstra's algorithm in the perspective of relative performance measures for most of the instances.

5.3. Sensitivity of Objective Weighting for the GLPK

Figure 3 illustrates the sensitivity to changes in the trade-off between objectives represented by modifying α. The sensitivity analysis sheds light on the change in the optimal path for different trade-offs. Figure 3 shows that the UAVs for alpha equal to 0 and 0.1 clearly stay in take-off and landing zone for the entire time horizon for both UAVs or just for one UAV. This is because the score for each grid cell outside the take-off zone is too high to consider. Finally, Figure 4 shows that the optimal path changes for almost all different alpha settings. However, the pattern of each path seems to follow the same structure because the path is sensitive to the parameter α, which also justifies the multi-objective formulation of the problem.

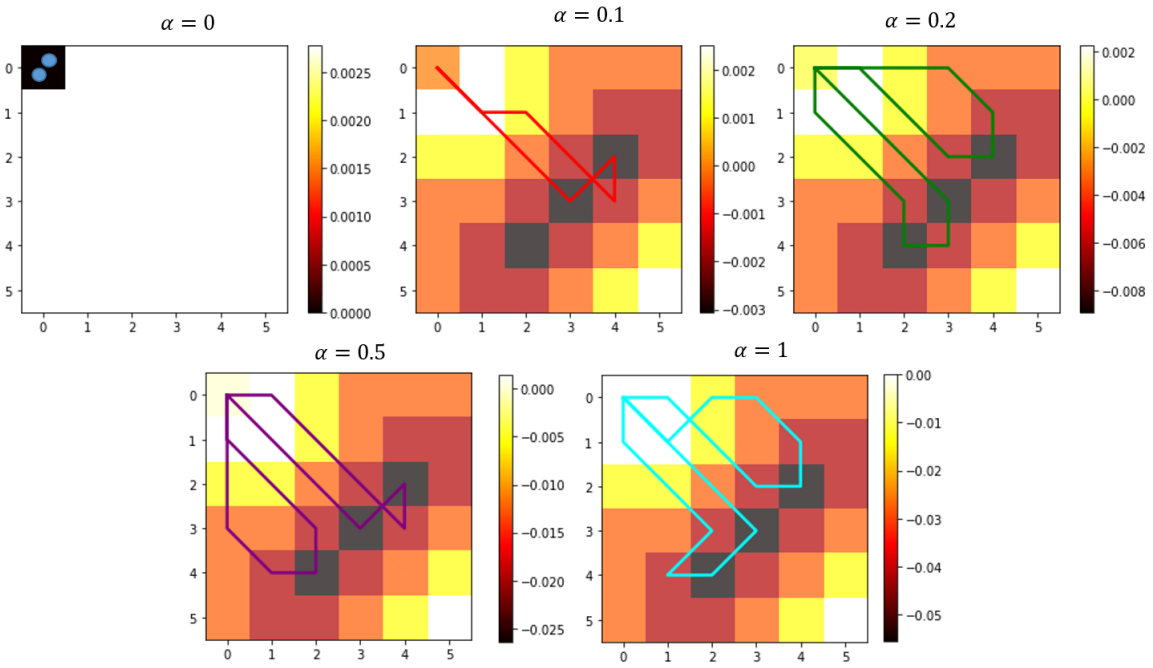

Figure 3. The corresponding route in 2D generated by GLPK for different weightings of alpha on the corresponding scoring map. Note that the illustrated paths is for two UAVs on a 6×6 grid with a time horizon of 10 and start and end in grid cell $[0, 0]$. In addition, for alpha = 0.1, the second UAV stays in the start cell.

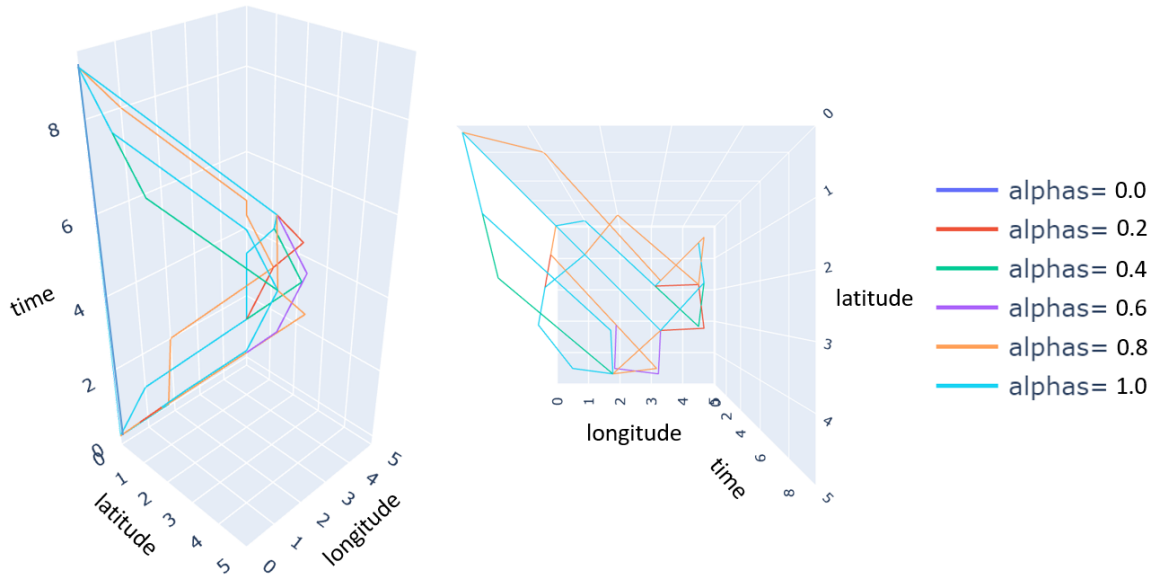

Figure 4. The corresponding route in three dimension generated by GLPK for different weights of alpha. The longitude and latitude axes represent the possible maneuvers on the grid, while time illustrates the time dimension.

5.4. Benefits and Adverse Circumstances Associated with Multi-Objective Framework

The results on the sensitivity clearly showcase some of the dangers when incorporating the bi-objective framework on the UAV pathfinding. It is very difficult to see which alpha enforces that all equipment will be employed and not spending too much time in the landing zone. Clearly, the solution procedure should allow UAVs to return before time, but it is very difficult to identify when it is too early to specify through the alpha parameter.

There is similarly a robustness issue when introducing the multi-objective framework as objectives can be conflicting, and a solution can satisfy an objective that is not of our interest. In the case of this paper, we are clearly interested in searching as many high probability cells as possible in as little time as possible. However, indicating how little time is too much is very difficult in the presented setting. The last thing one wants to introduce is nervousness in the scheduling, so some rules about searching different areas could be of advantage.

Nevertheless, introducing these additional objectives clearly brings us closer to the optimal goal. For these search and rescue Missions, we are interested in accumulating the highest probability of locating the missing target. We are, however, also interested in doing it as quickly as possible by obtaining the best quality images possible. Similarly, there could be a chance that the missing target has a higher probability of survival in some regions than others, which is why we also are interested in locating the target alive. Therefore, additional objectives other than the ones considered in this research could be introduced.

6. Conclusions

The smart city concept is almost around last couple of decades, and one of the critical concepts is to integrate cutting-edge technology without raising costs in improving environmental sustainability and life expectancy. In this direction, we proposed a multi-objective path planning and trajectory mapping problem under the mixed integer programming problem framework for a set of homogeneous UAVs deployed to search for static targets. A graph theory-based directed acyclic network representation is employed to reduce complexities and track the inward and outward movement of each UAV from its respective present cell location by ensuring flow conservation. A modification of the basic VNS algorithm is proposed and implemented in two phases to find the solution. In the first phase, a path is generated and in the second phase, trajectory mapping is done sequentially by considering constraints associated with the problem environment. Numerical simulation on synthetic experimental settings demonstrates that the proposed approach can reduce computational

complexity and provide a solution within reasonable amount of time compared to the exact solver. Moreover, it is found that the exact solver is unable to provide a solution within a time threshold. When we compare the relative performance of VNS with GLPK or Dijkstra's algorithm, it was found that Dijkstra's algorithm's performance is relatively lower as the grid size increases, which justifies the efficiency of the proposed algorithm. To our best knowledge, this is the first work to explore the path for multiple UAVs by using a bi-objective VNS algorithm. Considering the numerical evaluation, one can conclude that the approach presented in this study is a better alternative than the exact solver, and methodology can contribute to intelligent systems.

For future work, we intend to extend the proposed approach to calculate paths for finding moving targets. We assumed altitude differentiation from the perspective of collision avoidance. We ignored constraints such as fuel, sensor capacity, search pattern, etc., those need to be integrated to formulate a robust path planning model. We compared the outcome of proposed solution approach with exact solver, therefore one can employ other algorithms such as particle swarm optimization [65], bat algorithm [66], A^* algorithm [59], machine learning (ML) algorithms [29] etc. to compare the performance of the proposed VNS algorithm. Finally, one can use a multi-criterion decision-making algorithm [67] to incorporate customizable preferences of decision-makers robustly to take advantage of the inherent flexibility while setting weights.

Author Contributions: Conceptualization, S.S. and A.E.V.; methodology, S.S. and A.E.V.; software, A.E.V.; validation, I.N., A.H. and H.B.; formal analysis, I.N.; investigation, A.E.V.; resources, A.H. and H.B.; writing—review and editing, S.S., A.E.V. and I.N.; and project administration, S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data used in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Nielsen, I.E.; Dang, V.Q.; Nielsen, P.; Pawlewski, P. Scheduling of Mobile Robots with Preemptive Tasks, Advances in Intelligent Systems and Computing. In *Distributed Computing and Artificial Intelligence, 11th International Conference*; Springer: Berlin, Germany, 2014; pp. 19–29.
2. Sung, I., Nielsen, P. Zoning a service area of unmanned aerial vehicles for package delivery services. *J. Intell. Robot. Syst.* **2020**, *97*, 719–731. [CrossRef]
3. Thibbotuwawa, A.; Bocewicz, G.; Radzki, G.; Nielsen, P.; Banaszak, Z. UAV Mission planning resistant to weather uncertainty. *Sensors* **2020**, *20*, 515. [CrossRef]
4. Eagle, J.N. 1984. The optimal search for a moving target when the search path is constrained. *Oper. Res.* **1984**, *32*, 1107–1115. [CrossRef]
5. Pugliese, L.D.P.; Guerriero, F.; Zorbas, D.; Razafindralambo, T. Modelling the mobile target covering problem using flying drones. *Optim. Lett.* **2016**, *10*, 1021–1052. [CrossRef]
6. Samaniego, F.; Sanchis, J.; García-Nieto, S.; Simarro, R. Recursive Rewarding Modified Adaptive Cell Decomposition (RR-MACD): A Dynamic Path Planning Algorithm for UAVs. *Electronics* **2019**, *8*, 306. [CrossRef]
7. San, Juan, V.; Santos, M.; Andújar, J.M. Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity* **2018**. [CrossRef]
8. Stewart, T.J. Search for a moving target when searcher motion is restricted. *Comput. Oper. Res.* **1979**, *6*, 129–140. [CrossRef]
9. Wang, J.; Jiang, C.; Han, Z.; Ren, Y.; Maunder, R.G.; Hanzo, L. Taking drones to the next level: Cooperative distributed unmanned-aerial-vehicular networks for small and mini drones. *IEEE Veh. Technol. Mag.* **2017**, *12*, 73–82. [CrossRef]
10. Yuan, H.; Xiao, C.; Zhan, W.; Wang, Y.; Shi, C.; Ye, H.; Jiang, K.; Ye, Z.; Zhou, C.; Wen, Y. Target detection, positioning and tracking using new UAV gas sensor systems: Simulation and analysis. *J. Intell. Robot. Syst.* **2019**, *94*, 871–882. [CrossRef]
11. Felemban, E.; Sheikh, A.A.; Naseer, A. Improving response time for crowd management in Hajj. *Computers* **2021**, *4*, 46. [CrossRef]
12. Miyano, K.; Shinkuma, R.; Shiode, N.; Shiode, S.; Sato, T.; Oki, E. Multi-UAV allocation framework for predictive crime deterrence and data acquisition. *Internet Things* **2020**, *11*, 100205. [CrossRef]
13. Huang, S.; Gui, J.; Wang, T.; Li, X. Joint Mobile Vehicle–UAV Scheme for Secure Data Collection in a Smart City. *Ann. Telecommun.* **2020**, 1–22. Available online: https://www.researchgate.net/publication/344005230_Joint_mobile_vehicle-UAV_scheme_for_secure_data_collection_in_a_smart_city (accessed on 10 March 2021). [CrossRef]

14. Trummel, K.E.; Weisinger, J.R. The complexity of the optimal searcher path problem. *Oper. Res.* **1986**, *34*, 324–327. [CrossRef]
15. Brown, S.S. Optimal search for a moving target in discrete time and space. *Oper. Res.* **1980**, *28*, 1275–1289. [CrossRef]
16. Benkoski, S.J.; Monticino, M.G.; Weisinger, J.R. A survey of the search theory literature. *Nav. Res. Logist.* **1991**, *38*, 469–494. [CrossRef]
17. Washburn, A.R. Branch and bound methods for a search problem. *Nav. Res. Logist.* **1998**, *45*, 243–257. [CrossRef]
18. Lau, H.; Huang, S.; Dissanayake, G. Optimal search for multiple targets in a built environment. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 3740–3745.
19. Lau, H.; Huang, S.; Dissanayake, G. Discounted mean bound for the optimal searcher path problem with non-uniform travel times. *Eur. J. Oper. Res.* **2008**, *190*, 383–397. [CrossRef]
20. Rogge, J.A.; Aeyels, D. Multi-robot coverage to locate fixed and moving targets. In Proceedings of the 2009 IEEE Control Applications (CCA) & Intelligent Control (ISIC), St. Petersburg, Russia, 8–10 July 2009; pp. 902–907.
21. Li, X.; Yao, H.; Wang, J.; Xu, X.; Jiang, C.; Hanzo, L. A near-optimal UAV-aided radio coverage strategy for dense urban areas. *IEEE Trans. Veh. Technol.* **2019**, *68*, 9098–9109. [CrossRef]
22. Berger, J.; Lo N. An innovative multi-agent search-and-rescue path planning approach. *Comput. Oper. Res.* **2015**, *53*, 4–31. [CrossRef]
23. Perez-Carabaza, S.; Besada-Portas, E.; Lopez-Orozco, J.A.; Jesus, M. Ant colony optimization for multi-UAV minimum time search in uncertain domains. *Appl. Soft Comput.* **2018**, *62*, 789–806. [CrossRef]
24. Ye, F.; Chen, J.; Tian, Y.; Jiang, T. Cooperative task assignment of a heterogeneous multi-UAV system using an adaptive genetic algorithm. *Electronics* **2020**, *4*, 687. [CrossRef]
25. Lu, Y.; Ma, Y.; Wang, J.; Han, L. Task assignment of UAV swarm based on Wolf Pack algorithm. *Appl. Sci.* **2020**, *23*, 8335. [CrossRef]
26. Luo, R.; Zheng, H.; Guo, J. Solving the multi-functional heterogeneous UAV cooperative mission planning problem using multi-swarm fruit fly optimization algorithm. *Sensors* **2020**, *18*, 5026. [CrossRef]
27. Alhaqbani, A.; Kurdi, H.; Youcef-Toumi, K. Fish-inspired task allocation algorithm for multiple unmanned aerial vehicles in search and rescue missions. *Remote Sens.* **2021**, *1*, 27.
28. Xiong, C.; Chen, D.; Lu, D.; Zeng, Z.; Lian, L. Path planning of multiple autonomous marine vehicles for adaptive sampling using Voronoi-based ant colony optimization. *Robot. Auton. Syst.* **2019**, *115*, 90–103. [CrossRef]
29. Wang, J.; Jiang, C.; Zhang, H.; Ren, Y.; Chen, K.C.; Hanzo, L. Thirty years of machine learning: The road to Pareto-optimal wireless networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1472–1514. [CrossRef]
30. Kundid Vasić, M.; Papić, V. Multimodel Deep Learning for Person Detection in Aerial Images. *Electronics* **2020**, *9*, 1459. [CrossRef]
31. Li, Y.; Yuan, X.; Zhu, J.; Huang, H.; Wu, M. Multiobjective Scheduling of Logistics UAVs Based on Variable Neighborhood Search. *Appl. Sci.* **2020**, *10*, 3575. [CrossRef]
32. Lo N.; Berger, J.; Noel, M. Toward optimizing static target search path planning. In Proceedings of the 2012 IEEE Symposium on Computational Intelligence for Security and Defence Applications, Ottawa, ON, Canada, 11–13 July 2012; pp. 1–7.
33. Nielsen, L.D.; Sung, I.; Nielsen, P. Convex decomposition for a coverage path planning for autonomous vehicles: Interior extension of edges. *Sensors* **2019**, *19*, 4165. [CrossRef]
34. Perez-Carabaza, S.; Scherer, J.; Rinner, B.; López-Orozco, J.A.; Besada-Portas, E. UAV trajectory optimization for Minimum Time Search with communication constraints and collision avoidance. *Eng. Appl. Artif. Intell.* **2019**, *85*, 357–371. [CrossRef]
35. Raap, M.; Meyer-Nieberg, S.; Pickl, S.; Zsifkovits, M. Aerial vehicle search-path optimization: A novel method for emergency operations. *J. Optim. Theory Appl.* **2017**, *172*, 965–983. [CrossRef]
36. Sitek, P.; Wikarek, J.; Rutczyńska-Wdowiak, K.; Bocewicz, G.; Banaszak, Z. Optimization of capacitated vehicle routing problem with alternative delivery, pick-up and time windows: A modified hybrid approach. *Neurocomputing* **2021**, *423*, 670–678. [CrossRef]
37. Thibbotuwawa, A.; Bocewicz, G.; Zbigniew, B.; Nielsen, P. A solution approach for UAV fleet mission planning in changing weather conditions. *Appl. Sci.* **2019**, *9*, 3972. [CrossRef]
38. Mladenovic, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [CrossRef]
39. Loudni, S.; Boizumault, P.; David, P. On-line resources allocation for ATM networks with rerouting. *Comput. Oper. Res.* **2006**, *33*, 2891–2917. [CrossRef]
40. Geiger, M.J.; Wenger, W. On the assignment of students to topics: A Variable Neighborhood Search approach. *Socio-Econ. Plan. Sci.* **2010**, *44*, 25–34. [CrossRef]
41. Brusco, M.J.; Singh, R.; Steinley, D. Variable neighborhood search heuristics for selecting a subset of variables in principal component analysis. *Psychometrics* **2009**, *74*, 705. [CrossRef]
42. Schilde, M.; Doerner, K.F.; Hartl, R.F.; Kiechle, G. Metaheuristics for the bi-objective orientation problem. *Swarm Intell.* **2009**, *3*, 179–201. [CrossRef]
43. Anghinolfi, D.; Paolucci, M. Parallel machine total tardiness scheduling with a new hybrid metaheuristic approach. *Comput. Oper. Res.* **2007**, *34*, 3471–3490. [CrossRef]
44. Qian, B.; Wang, L.; Huang, D.X.; Wang, X. Multi-objective flow shop scheduling using differential evolution. In *Intelligent Computing in Signal Processing and Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 1125–1136.
45. Fleszar, K.; Osman, I.H.; Hindi, K.S. A variable neighborhood search algorithm for the open vehicle routing problem. *Eur. J. Oper. Res.* **2009**, *195*, 803–809. [CrossRef]

46. Montemanni, R.; Smith, D.H. Construction of constant GC-content DNA codes via a variable neighborhood search algorithm. *J. Math. Model. Algorithms* **2008**, *7*, 311. [CrossRef]
47. Hansen, P.; Mladenović, N.; Pérez, J.A.M. Variable neighborhood search: Methods and applications. *Ann. Oper. Res.* **2010**, *175*, 367–407. [CrossRef]
48. Ribeiro, C.C.; Aloise, D.; Noronha, T.F.; Rocha, C.; Urrutia, S. An efficient implementation of a VNS/ILS heuristic for a real-life car sequencing problem. *Eur. J. Oper. Res.* **2008**, *191*, 596–611. [CrossRef]
49. Gosiewski, Z.; Kwaśniewski, K. Time Minimization of Rescue Action Realized by an Autonomous Vehicle. *Electronics* **2020**, *9*, 2099. [CrossRef]
50. Lanillos, P.; Yañez-Zuluaga, J.; Ruz, J.J.; Besada-Portas, E. A bayesian approach for constrained multi-agent minimum time search in uncertain dynamic domains. In Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, Amsterdam, The Netherlands, 6–10 July 2013; pp. 391–398.
51. Peng, Z.H.; Wu, J.P.; Chen, J. Three-dimensional multi-constraint route planning of unmanned aerial vehicle low-altitude penetration based on coevolutionary multi-agent genetic algorithm. *J. Cent. South Univ. Technol.* **2011**, *18*, 1502. [CrossRef]
52. Sakawa, M.; Yano, H.; Nishizaki, I.; Nishizaki, I. *Linear and Multiobjective Programming with Fuzzy Stochastic Extensions*; Springer US: New York, NY, USA, 2013.
53. Chen, L.; Peng, J.; Zhang, B. Uncertain goal programming models for bicriteria solid transportation problem. *Appl. Soft Comput.* **2017**, *51*, 49–59. [CrossRef]
54. Chung, C.K.; Chen, H.M.; Chang, C.T.; Huang, H.L. On fuzzy multiple objective linear programming problems. *Expert Syst. Appl.* **2018**, *114*, 552–562. [CrossRef]
55. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T.A.M.T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
56. Moon, I.; Jeong, Y.J.; Saha, S. Fuzzy bi-objective production-distribution planning problem under the carbon emission constraint. *Sustainability* **2016**, *8*, 798. [CrossRef]
57. Davoodi, M.; Panahi, F.; Mohades, A.; Hashemi, S.N. Multi-objective path planning in discrete space. *Appl. Soft Comput.* **2013**, *13*, 709–720. [CrossRef]
58. Yuan, Z.; Yang, Z.; Lv, L.; Shi, Y. A Bi-Level Path Planning Algorithm for Multi-AGV Routing Problem. *Electronics* **2020**, *9*, 1351. [CrossRef]
59. Sathyaraj, B.M.; Jain, L.C.; Finn, A.; Drake, S. Multiple UAVs path planning algorithms: A comparative study. *Fuzzy Optim. Decis. Mak.* **2008**, *7*, 257. [CrossRef]
60. Nielsen, I.E.; Bocewicz, G.; Saha, S. Multi-Agent Path Planning Problem Under a Multi-objective Optimization Framework. In *DCAI 2020: Distributed Computing and Artificial Intelligence, Special Sessions, 17th International Conference, Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence, L'Aquila, Italy, 17–19 June 2020*; Springer: Cham, Switzerland, 2020; pp. 5–14.
61. Wu, Y.K.; Liu, C.C.; Lur, Y.Y. Pareto-optimal solution for multiple objective linear programming problems with fuzzy goals. *Fuzzy Optim. Decis. Mak.* **2015**, *14*, 43–55. [CrossRef]
62. Marler, R.T.; Arora, J.S. Survey of multi-objective optimization methods for engineering. *Struct. Multidiscip. Optim.* **2004**, *26*, 369–395. [CrossRef]
63. Makhorin, A. GLPK (GNU Linear Programming Kit). 2008. Available online: http://www.gnu.org/s/glpk/glpk.html (accessed on 12 March 2019).
64. Hidalgo-Paniagua, A.; Vega-Rodríguez, M.A.; Ferruz, J. Applying the MOVNS (multi-objective variable neighborhood search) algorithm to solve the path planning problem in mobile robotics. *Expert Syst. Appl.* **2016**, *58*, 20–35. [CrossRef]
65. Jeong, Y.; Saha, S.; Chatterjee, D.; Moon, I. Direct shipping service routes with an empty container management strategy. *Transp. Res. Part E Logist. Transp. Rev.* **2018**, *118*, 123–142. [CrossRef]
66. Saha, S.; Chatterjee, D.; Sarkar, B. The ramification of dynamic investment on the promotion and preservation technology for inventory management through a modified flower pollination algorithm. *J. Retail. Consum. Serv.* **2021**, *58*, 102326. [CrossRef]
67. Vasegaard, A.E.; Picard, M.; Hennart, F.; Nielsen, P.; Saha, S. Multi criteria decision making for the multi-satellite image acquisition scheduling problem. *Sensors* **2020**, *20*, 1242. [CrossRef]