Drone Routing and Optimization for Post-Disaster Inspection

Sudipta Chowdhury

Drone routing and optimization for post-disaster inspection

By

Sudipta Chowdhury

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Industrial and Systems Engineering
in the Bagley College of Engineering

Mississippi State, Mississippi

May 2018

Drone routing and optimization for post-disaster inspection

By

Sudipta Chowdhury

Approved:

Mohammad Marufuzzaman
(Major Professor)

Linkan Bian
(Minor Professor)

Hugh R. Medal
(Committee Member)

Xiaopeng Li
(Committee Member)

Stanley F. Bullington
(Graduate Coordinator)

Jason M. Keith
Dean
College's Name

Name: Sudipta Chowdhury

Date of Degree: May 4, 2018

Institution: Mississippi State University

Major Field: Industrial and Systems Engineering

Major Professor: Mohammad Marufuzzaman

Title of Study: Drone routing and optimization for post-disaster inspection

Pages in Study 59

Candidate for Degree of Master of Science

In this study, we propose a mixed-integer linear programming model for a Heterogeneous Fixed Fleet Drone Routing problem (HFFDRP) that minimizes the post-disaster inspection cost of a disaster-affected area by accounting a number of drone trajectory-specific factors into consideration such as battery recharging costs, servicing costs, drone hovering, turning, acceleration, constant, and deceleration costs, and many others. The trajectories between each pair of nodes are constructed using a path construction model. Two heuristic algorithms are proposed, namely, Adaptive Large Neighborhood Search (ALNS) algorithm and Modified Backtracking Adaptive Threshold Accepting (MBATA) algorithm, to solve the largest instances of our proposed optimization model. Computational results indicate that the proposed MBATA algorithm is capable of producing high-quality solutions consistently within a reasonable amount of time. Finally, a real-life case study is used to visualize and validate the modeling.

DEDICATION

This work is dedicated to my parents and sister Pranabesh Chowdhury, Atasi Barua, and Susmita Chowdhury.

ACKNOWLEDGEMENTS

I humbly acknowledge the discussions on vehicle routing with my friend Musabbir Hasan Sumon, which helped me develop the algorithms used in this study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Unmanned aerial vehicles (UAVs), commonly known as drones, are receiving tremendous attention from the humanitarian organizations due to their ability to monitor/access cutoff regions following a natural/man-made catastrophe. There are a number of disasters where drones have already been utilized such as damage monitoring in Nepal earthquake [1] and Fukushima nuclear power plant [2], humanitarian aid distribution in Haiti and the Dominican Republic [3], and many others. Despite of drones high potential to serve in harsh environments, they are surprisingly still under-utilized in disaster response applications primarily due to their technological limitations, set legislation, and many more [4].

A large technological explosion in the development of drones has been realized in recent years [5]. Military sector is primarily responsible for such advancements [6]. While these advanced drones are not typically being implemented for civilian purposes, nevertheless, miniaturization of electronic control systems and the availability of low-cost sensors in the marketplace have led to the development of more sophisticated civilian drones. These drones are equipped with various features, including but not limited to global positioning system (GPS), wireless communication sensors, high resolution cameras, and many other features which enable drones to perform special functions such as automated flight control, capturing high quality images, object tracking, and collision avoidance. Leveraging these functions, drones posses high potential to apply them in disaster management applications. Drones can support real-time monitoring of a disaster-affected area which enables an administrator to quickly gather relevant information that can be analyzed to plan and efficiently deploy humanitarian relief operations.

When a disaster strikes, critical infrastructures such as transportation network and sensitive facilities (e.g., power plants, nuclear plants) might be heavily impacted. Critical infrastructures' assets, systems, and networks, whether physical or virtual, are considered extremely important since their incapacitation or destruction might have a debilitating effect on nations economy, health, security, safety, or any combination thereof. Due to the potential damage of the traditional transportation network, ensuring logistic support and proper assessment of disaster-affected areas are often challenging. Even if the transportation network is functioning properly, it is preferred very seldom to have direct human involvement in disaster-affected areas due to the exposure of potential hazardous situations. This being the case, drones are considered as one of the ideal substitutes since they are capable of providing information for risk assessment along with mapping and planning disaster-affected areas with minimal human intervention.

Although drone routing has been a research area of interest in recent years, the majority of the existing studies primarily focus on the concept of last-mile delivery of parcels to the customers, where drones can be used either as a standalone mode of transportation or in conjunction with other modes of transportation such as trucks. Murray and Chu [7] introduce a new variant of the traditional Traveling Salesman Problem (TSP), referred to as Flying Sidekick Traveling Salesman Problem (FSTSP), to investigate the challenges of determining optimal customer assignments to a drone working jointly with a delivery truck. Likewise, a mathematical formulation, referred to as TSP with Drone (TSP-D), has been proposed by Agatz et al. [8] for the same problem that optimizes the simultaneous movement of a truck and drone on a road network. Coelho et al. [9] propose a real-time routing problem where different types of drones are allowed to collect and deliver packages simultaneously. Ulmer and Thomas [10] survey how the possibility of combining delivery trucks with drones might reduce the required delivery costs and increase the number of customers served in the same day. The authors label this resultant problem as Same-Day Delivery Routing Problem with Heterogeneous Fleets (SDDPHF). In addition to these studies, numerous large companies, such as Amazon, DHL, and Google already shown their interests in applying drones to deliver parcels in urban areas [11, 7]. Drone applications can also be found in border line surveillance [12], power line inspection [13, 14], soil erosion monitoring [15], security operations in the oil and gas industry [16], and many others. Additionally, Kim et al. [17] study how the stochasticity associated with battery duration, caused by the air temperature, impact the drones flight schedule.

Although it is universally accepted that it is barely possible to neutralize all the negative impacts of disasters, their impacts, however, can be mitigated by adopting appropriate and timely disaster preparedness and management actions. Drones can signifi- cantly benefit in this regard as well as during the entire disaster management cycle, i.e., mitigation, preparedness, response, and recovery stages [18]. Till now few studies focus on drone delivery in emergency situations, mainly concentrating on the combination of drones with alternative means of transport (e.g., trucks), if available.

Chowdhury et al. [4] propose a Continuous Approximation (CA) model that designs the potentiality of using drones as a mode of transportation to supply emergency commodities in a disaster-affected region.

Table 1.1 Distinguishing our study with related vehicle routing and drone routing literature

References	Heterogeneous vehicles/ drones	Fuel/battery level	Service time	Recharging stations	Speed optimization	Energy consumption
Vehicle routing literature						
Erdogan & Miller-Hooks [22]		✓	✓	✓		
Demir et al. [23]		✓	✓		✓	✓
Koc et al. [24]	✓	✓	✓		✓	✓
Markov et al. [25]	✓	✓	✓	✓		
Moshref-Javadi and Lee [26]	✓		✓			
Kwon et al. [27]	✓					
Drone routing literature						
Rabta et al. [21]				✓		✓
Kim et al. [17]	✓					
Lim et al. [13]	✓	✓	✓			
Murray and Chu [7]		✓	✓			
Coelho et al. [9]	✓	✓	✓	✓	✓	✓
Cho et al. [16]	✓	✓	✓			
Our study	✓	✓	✓	✓	✓	✓

Shang et al. [19] and Quaritsch et al. [20] formulate a TSP to get the shortest civil drone flight path in surveying and taking images of a disaster-affected region. Rabta et al. [21] propose a mixed-integer linear programming model that minimizes the total traveling distance of a drone in a disaster-affected region by taking into account payload and energy constraints while recharging stations are installed to allow the extension of the operating distance of the drone. The authors also discussed different priority policies (e.g., relative priority, absolute priority) to serve a disaster-affected region. Different from this study, our study additionally considers heterogeneous drone types, speed optimization, battery capacity, and accurate representation of the energy cost.

In this paper, we propose an optimization modeling framework to solve a new variant of the Heterogeneous Fixed Fleet Vehicle Routing Problem (HFFVRP), referred to as Heterogeneous Fixed Fleet Drone Routing Problem (HFFDRP), to design a safe, reliable, and cost-efficient disaster-affected region inspection plan using battery-driven drones. A mixed-integer linear programming model (MILP) is proposed to minimize the post-disaster inspection cost by considering a number of drone trajectory-specific factors into consideration such as ascending and descending costs, battery recharging costs, servicing costs (i.e., costs associated with taking images at disaster-affected nodes), drone hovering, turning, acceleration, constant, and deceleration costs. We consider six key features of a HFFDRP in addition to the basic features available in a Vehicle Routing Problem (VRP) framework (e.g., visiting each node exactly once, sub-tour elimination constraints), namely, the consideration of heterogeneous drones, energy consumption, recharging stations along with battery level tracking, service time, and drone velocity (speed) optimization. These features are critical to realistically model a drone routing problem in serving a disaster-affected region. Table 1 summarizes the key differences between our study with related vehicle routing and drone routing literature. We note that our model is very similar to the model proposed by Coelho et al. [9] except the case that we more realistically capture the energy cost, develop a path construction model, and apply the model to a disaster-management problem. We further note to the readers that the energy consumption cost components, referred in Table 1, are significantly different in drone transportation compared to regular vehicular transportation (e.g., trucks). In addition to proposing the modeling framework, another important contribution of this paper is to propose a path construction model by obeying the restrictions set forward by the Federal Aviation Agency (FAA), technological, geographical, and environment restrictions of drone transportation. Furthermore, we realize that there is an urgency to quickly inspect a large number of disaster-affected nodes. To alleviate this problem, we propose two heuristic algorithms, namely, Adaptive Large Neighborhood Search (ALNS) algorithm and a Modified Backtracking Adaptive Threshold Accepting (MBATA) algorithm, to efficiently solve our proposed optimization model in a reasonable amount of time. Finally, we use Hancock county from Mississippi State as a test bed to visualize and validate the modeling results. The outcome of this study provides a number of managerial insights such as how the drone depot location, number of disaster affected nodes and battery recharging stations, maximum allowable travel time, and battery recharging time impact the design and management of a drone routing operation.

The exposition of this paper is as follows. Section 2 introduces the problem and the routing and path-construction model formulations. The proposed solution algorithms to solve the optimization problem are then discussed in Section 3. The first part of Section 4 describes the data used to generate instances of the research problem. The second part determines the efficiency and effectiveness of the proposed solution algorithms, while the third part of this section performs sensitivity analysis by varying a number of key parameters from a real-life case example. Finally, Section 5 summarizes the paper and discusses future research directions.

CHAPTER II

PROBLEM DESCRIPTION AND MODEL FORMULATION

This section first introduces a mathematical formulation for a Heterogeneous Fixed Fleet Drone Routing Problem (HFFDRP). The goal is to minimize the overall routing cost of drones to monitor/mapping a disaster-affected region. Next, we present a path construction model to obtain optimal/feasible routes between two disaster-affected nodes by obeying the restrictions set forward by the Federal Aviation Agency (FAA) (i.e., should operate under a ceiling of 400 feet), technological (e.g., limited battery and weight carrying capacities), geographical (i.e., cannot fly over a densely populated area), and environmental (e.g., wind speed and direction) considerations. Finally, variable fixing and valid inequalities are proposed to accelerate the computational performance of the proposed mathematical model formulation.

Mathematical Model Formulation for HFFDRP

In the following, the sets and indices, parameters, and decision variables are briefly explained and followed by the mathematical formulation.

Sets and Indices

- **I:** set of disaster-affected regions, indexed by *i* and *j*
- $\boldsymbol{\mathcal{F}}$: set of battery recharging stations, indexed by *i* and *j*
- $\boldsymbol{\Phi}$: set of dummy recharging nodes for battery recharging stations, indexed by *i* and *j*
- $\boldsymbol{K}$: set of drones, indexed by *k*
- $\boldsymbol{S}$: set of drone speed levels, indexed by *s*

- $\mathcal{F}_\Phi$**:** set of battery recharging stations including dummy recharging nodes, indexed by *i* and *j*
- ***V*:** set of nodes including depot (v_0), disaster-affected regions, battery recharging stations, and dummy recharging nodes, i.e., $V = v_0 \cup I \cup \mathcal{F} \cup \Phi$

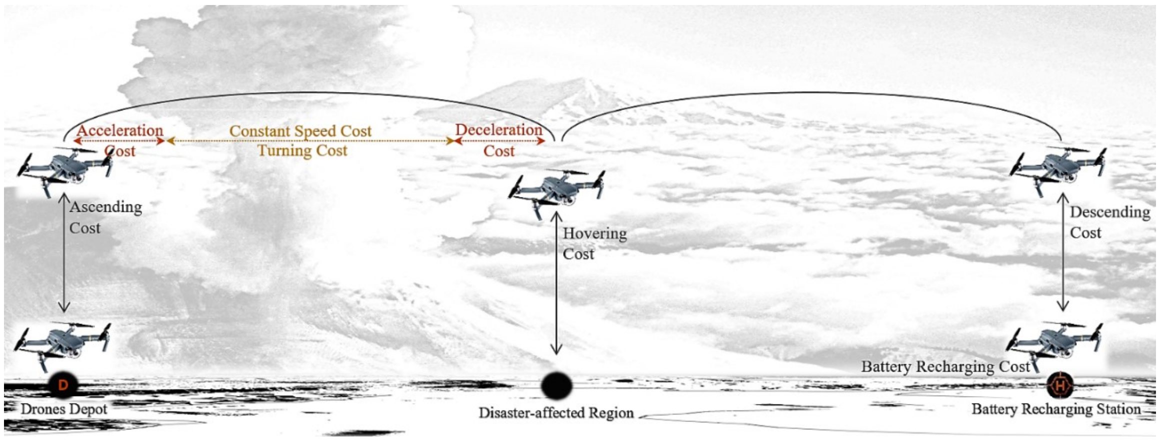


Figure 2.1 Illustration for different stages of a drone flight and its respective cost components

Parameters

- $\boldsymbol{d_{ij}}$: trajectory distance[1] between source node $i \in V$ to destination node $j \in V|i \neq j$
- $\boldsymbol{m}$: number of available drones
- $\boldsymbol{q_k}$: battery capacity of drone $k \in K$
- $\boldsymbol{\psi_{jk}}$: service/recharging time required by drone $k \in K$ at node $j \in V \setminus \{v_0\}$
- $\boldsymbol{e_{jk}}$: unit battery recharging cost by drone $k \in K$ at battery recharging station $j \in \mathcal{F}_\Phi$
- $\boldsymbol{f_{jk}}$: unit service cost by drone $k \in K$ at disaster-affected region $j \in I$
- $\boldsymbol{c_{ijk}}$: unit battery consumption cost by drone $k \in K$ through arc $(i,j) \in V|i \neq j$
- $\boldsymbol{c_{ijks}}$: unit battery consumption cost by drone $k \in K$ through arc $(i,j) \in V|i \neq j$ under speed level $s \in S$
- $\boldsymbol{t_k^{max}}$**:** maximum allowable travel time of drone $k \in K$
- $\boldsymbol{\overline{v}_{ks}}$: average velocity of drone $k \in K$ under speed level $s \in S$
- $\boldsymbol{r_{ks}}$: battery consumption rate by drone $k \in K$ under speed level $s \in S$
- $\boldsymbol{p_k^a}/\ \boldsymbol{p_k^d}$: ascending/descending motor power required by drone $k \in K$
- $\boldsymbol{h_k}$: flying altitude of drone $k \in K$

[1] Trajectories are determined based upon the path construction model.

- $\boldsymbol{v_k^a}$/ $\boldsymbol{v_k^d}$: ascending/descending velocity of drone $k \in K$
- $\boldsymbol{\beta_k}$: minimum power required to hover over a disaster-affected region by drone $k \in K$
- $\boldsymbol{\lambda_k}$: motor speed multiplier of drone $k \in K$
- $\boldsymbol{w_{ks}^{turn}}$: angular velocity required by drone $k \in K$ under speed level s ∈ S
- $\boldsymbol{\bar{\theta}_{ij}}$: average rotational angle between source node $i \in V$ to destination node $j \in V|i \neq j$
- $\boldsymbol{n_{ij}}$: number of turns between source node $i \in V$ to destination node $j \in V|i \neq j$
- $\boldsymbol{p_{ks}^{turn}}$: motor power required by each turn of drone $k \in K$ under speed level s ∈ S
- $\boldsymbol{\Gamma_k}$: maximum payload of drone $k \in K$ including drone's weight
- $\boldsymbol{g_1/g_2/g_3}$**:** percentage of the drone's trajectory during acceleration/intermediate/ deceleration phase associated to arc $(i,j) \in V|i \neq j$
- $\boldsymbol{p_k^{acc}/p_k^{int}/\ p_k^{dec}}$**:** power consumption rate by drone $k \in K$ during acceleration/intermediate/ deceleration phase

Decision Variables

- $\boldsymbol{X_{ijk}}$: 1 if arc $(i,j) \in V|i \neq j$ is traversed by drone $k \in K$; 0 otherwise
- $\boldsymbol{Z_{ijkS}}$: 1 if drone $k \in K$ travels through arc $(i,j) \in V|i \neq j$ under speed level s ∈ S, 0 otherwise
- $\boldsymbol{Y_{ik}}$: battery recharging level of drone $k \in K$ at arrival time on node $i \in V$
- $\boldsymbol{\tau_{ik}}$: arrival time of drone $k \in K$ at node $i \in V$

We now introduce the decision variables of our proposed mathematical model formulation. The first and second sets of binary decision variables, i.e., $\mathbf{X} \coloneqq \{X_{ijk}\}_{(i,j)\in V|i \neq j, k \in K}$ and $\mathbf{Z} \coloneqq \{Z_{ijks}\}_{(i,j)\in V|i \neq j, k \in K, s \in S}$ are defined to determine optimal drone routes with respect to different speed levels. The third set of decision variables, i.e., $\mathbf{Y} \coloneqq \{Y_{ik}\}_{i \in V, k \in K}$ defines the remaining battery based upon arrival time of a drone on each node, while the last set of decision variables, i.e., $\boldsymbol{\tau} \coloneqq \{\tau_{ik}\}_{i \in V, k \in K}$ defines the arrival time of a drone on each node.

The objective of model **[DR]** is to minimize the post-disaster inspection cost by considering a number of drone trajectory-specific factors into consideration such as ascending and descending costs, battery recharging costs, servicing costs (i.e., costs associated with taking images at disaster-affected nodes), drone hovering, turning, acceleration, constant, and deceleration costs. The goal is to design a safe, reliable, and cost-efficient disaster-affected region inspection plan using battery-driven drones.

$$[\mathrm{DR}]^{Minimize}_{X,Z,Y,\tau}\ \underbrace{\sum_{k\in K}\sum_{J\in V\setminus\{v_0\}} c_{0jk}\left(\frac{h_k}{v_k^a}\right)p_k^a X_{0jk} + \sum_{k\in K}\sum_{j\in F_\Phi}\sum_{i\in V} c_{jik}\left(\frac{h_k}{v_k^a}\right)p_k^a X_{jik}+}_{\boxed{\text{Ascending cost}}}$$

$$\underbrace{\sum_{k\in K}\sum_{j\in V\setminus\{v_0\}} c_{j0k}\left(\frac{h_k}{v_k^d}\right)p_k^d X_{jok} + \sum_{k\in K}\sum_{j\in F_\Phi}\sum_{i\in V} c_{ijk}\left(\frac{h_k}{v_k^d}\right)p_k^d X_{ijk} \quad +}_{\boxed{\text{Descending cost}}}$$

$$\underbrace{\sum_{k\in K}\sum_{j\in F_\Phi}\sum_{i\in V}(e_{jk}\Psi_{jk})X_{jik}+}_{\boxed{\text{Battery Recharging}}}\quad \underbrace{\sum_{k\in K}\sum_{j\in I}\sum_{i\in V}(f_{jk}\Psi_{jk})X_{ijk}+}_{\boxed{\text{Disaster-affected Service}}}$$

$$\sum_{s\in S}\sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ijks}\Big(\underbrace{(\beta_k+\lambda_k)\left(\frac{d_{ij}}{\bar{v}_{ks}}\right)}_{\boxed{\text{Hovering}}} + \underbrace{p_{ks}^{turn}\left(\frac{n_{ij}\bar{\theta}_{ij}}{w_{ks}^{turn}}\right)\Big)+}_{\boxed{\text{Turning}}}$$

$$\underbrace{\left(\sqrt{(\Gamma_k\bar{v}_{ks}p_k^{acc})}\right)g_1}_{\boxed{\text{Acceleration Cost}}} + \underbrace{\left(\frac{2p_k^{int}}{\bar{v}_{ks}}\right)g_2}_{\boxed{\text{Constant Speed Cost}}} + \underbrace{\sqrt{(\Gamma_k\bar{v}_{ks}p_k^{dec})g_3)d_{ij})Z_{ijks}}}_{\boxed{\text{Deceleration Cost}}} \qquad (1)$$

subject to:

$$\sum_{k\in K}\sum_{j\in V} X_{ijk} = 1 \qquad \forall i\in I \qquad (2)$$

$$\sum_{i\in V} X_{ijk} = \sum_{i\in V} X_{jik} \qquad \forall j\in V, k\in K \qquad (3)$$

$\sum_{k\in K}\sum_{j\in V} X_{ijk} \leq 1 \qquad \forall i \in F_{\Phi} \qquad (4)$

$\sum_{k\in K}\sum_{j\in V\{v_0\}} X_{ojk} \leq m \qquad (5)$

$\sum_{s\in S} Z_{ijks} = X_{ijk} \qquad \forall k \in K, (i,j) \in V \qquad (6)$

$\tau_{jk} + t_k^{max}(1 - X_{ijk}) \geq \tau_{ik} + \Psi_{ik}X_{ijk} + \frac{(\sum_{s\in S} d_{ij}Z_{ijks})}{\sum_{s\in S}\bar{v}_{ks}} \qquad \forall k \in K, i \in V, j \in V\backslash\{v_0\} \qquad (7)$

$\frac{\sum_{s\in S} d_{0i}Z_{0iks}}{\bar{v}_{ks}} \leq \tau_{ik} \leq t_k^{max} - \frac{\sum_{s\in S} d_{i0}Z_{i0ks}}{\sum_{s\in S}\bar{v}_{ks}} - \psi_{ik} \qquad \forall k \in K, i \in V\backslash\{v_0\} \qquad (8)$

$\tau_{0k} \leq t_k^{max} \qquad \forall k \in K \qquad (9)$

$Y_{ik} = q_k \qquad \forall k \in K,\ i \in F_{\Phi} \cup \{0\} \qquad (10)$

$Y_{jk} \leq Y_{ik} - \sum_{s\in S} r_{ks}d_{ij}Z_{ijks} + q_k(1 - X_{ijk}) \qquad \forall k \in K, i \in V, j \in I \qquad (11)$

$Y_{ik} \geq \sum_{s\in S} r_{ks}d_{i0}Z_{i0ks} \qquad \forall k \in K,\ i \in V\backslash\{v_0\} \qquad (12)$

$Y_{ik} \geq \sum_{s\in S} r_{ks}d_{ij}Z_{ijks} \qquad \forall k \in K,\ i \in I,\ j \in F_{\Phi} \qquad (13)$

$X_{ijk} \in \{0,1\} \qquad \forall k \in K, (i,j) \in V \qquad (14)$

$Z_{ijks} \in \{0,1\} \qquad \forall k \in K, (i,j) \in V, s \in S \qquad (15)$

$Y_{jk,}\tau_{ik} \geq 0 \qquad \forall k \in K, j \in V \qquad (16)$

The first and second terms of the objective function in **[DR]** represent, respectively, the energy consumption costs associated with ascending drones from depot and battery recharging stations. The third and fourth terms provide similar costs (i.e., energy consumption costs) which are incurred due to descending drones at depot and battery recharging stations. Note that the velocity (v_k^a/ v_k^d:), required motor power (p_k^a/ p_k^d) and flying altitude (h_k) contribute in the ascending and descending cost components of the drone transportation. The fifth and sixth terms represent, respectively, the costs associated with battery recharging and disaster-affected region servicing costs by drones.

Finally, the last five terms in the objective function represent, respectively, the costs associated with hovering, turning, accelerating, maintaining constant speed, and decelerating drones at a particular speed. These cost components are crucial since they severely impact the serviceability of drones in a disaster-affected region.

Constraint (2) guarantee that each disaster-affected node is visited by only one drone. Constraints (3) are known as the flow conservation constraints which ensure that at each node of a complete directed graph, i.e., $j \in V\backslash\{v_0\}$, an incoming arc of a drone must be followed by an outgoing arc of that drone. Constraints (4), implemented for connectivity to battery recharging stations, indicate that a battery recharging station might not be visited by drones. Constraints (5) limit the availability of drones at the depot. Constraints (6) ensure that only one speed level is chosen for drone $k \in K$ to traverse an arc $(i, j) \in V$. Constraints (7) determine the arrival time of a drone at each node of its route.

The difference between arrival times of two consecutive nodes of a drone trajectory if determined by the servicing/battery recharging time along with the travel time between the nodes. Constraints (8) restrict the maximum time availability of a drone to serve a disaster-affected region. Constraints (9) indicate that drone $k \in K$ will have maximum t_k^{max} time availability at the depot. Constraints (10) ensure that the drone batteries reach to full charging at depot and battery recharging stations. Constraints (11) guarantee minimum required battery to travel through a trajectory in a disaster-affected region. Constraints (12) and (13) determine the minimum required battery to travel from any node to depot and battery recharging stations, respectively. Finally, constraints (14) and (15) set binary restrictions while constraints (16) are the standard non-negativity constraints.

Path Construction Model for Drone Transportation

Contrary to vehicle transportation, no pre-existing paths are available for drone transportation. Therefore, it is crucial to obtain an optimal or near optimal path between each pair of nodes $(i, j) \in V | i < j$ in order to realistically solve model **[DR]**. However, generating paths between each pair of nodes would be challenging due to a number of factors that impact drone transportation, such as restrictions set forward by the Federal Aviation Agency (FAA) (i.e., should operate under a ceiling of 400 feet), technological (e.g., limited battery and weight carrying capacities), geographical (i.e., cannot fly over a densely populated area), and environmental (e.g., wind speed and direction) factors [1]. All these factors are crucial since they severely limit the effective range in utilizing drones for the disaster surveillance/monitoring operations. This being the case, we now introduce a path construction model that can be used to construct paths between each pair of nodes in model **[DR]**.

Let A_{ij} be the set of points to construct eligible segments between each pair of nodes $(i, j) \in V | i \neq j$, indexed by $(p, p') \in A_{ij}$. We considered a segment eligible based on the definition provided below:

Definition 1: A segment is considered eligible if and only if the segment satisfies the restrictions set forward by FAA (i.e., should operate under a ceiling of 400 feet), technological (e.g., limited battery and weight carrying capacities), geographical (i.e., cannot fly over a densely populated area), and environmental (e.g., wind speed and direction) factors.

Based on the above definition, we first construct eligible segments outside the path construction model so that the model produces optimal/feasible path(s) based on eligible segments. However, before delving into the details of eligible segments and optimal/feasible path construction model, the following set and parameters are introduced first:

Set

- $\boldsymbol{A_{ij}}$: set of points to construct eligible segments between path $(i,j) \in V|i \neq j$, indexed by $(p,p') \in A_{ij}$

Parameters

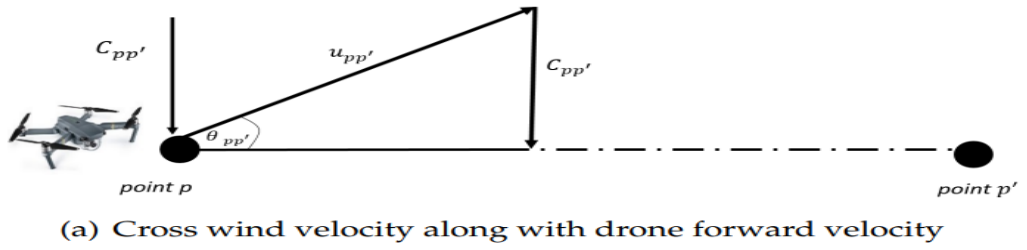
- $\boldsymbol{c_{pp'}}$: unit battery consumption cost through segment $(p,p') \in A_{ij}$
- $\boldsymbol{u_{pp'}}$: forward velocity of drone during moving through segment $(p,p') \in A_{ij}$
- $\boldsymbol{I_{pp'}}$: length of segment $(p,p') \in A_{ij}$
- $\boldsymbol{P_{pp'}}$: flying power required for moving drone through a segment $(p,p') \in A_{ij}$
- $\boldsymbol{t_{pp'}}$: time required for moving through segment $(p,p') \in A_{ij}$
- $\boldsymbol{h_{pp'}/h'_{pp'}}$: minimum/maximum altitude of segment $(p,p') \in A_{ij}$
- $\boldsymbol{w_{pp'}}$: cross wind velocity at segment $(p,p') \in A_{ij}$
- $\boldsymbol{\theta_{pp'}/\theta'_{pp'}}$: maximum deviation/turning angle
- $\boldsymbol{t_{ij}}$: maximum flying time to cross path $(i,j) \in V$
- $\boldsymbol{l_{ij}}$: maximum length of path $(i,j) \in V$

We ensure that the following conditions are satisfied to qualify for an eligible segment:

- Let $I_{pp'}$ be length of segment $(p,p') \in A_{ij}$. It is ensured that the length of segment $(p,p') \in A_{ij}$ should be greater than a minimum segment length, l_{min}, i.e., $I_{pp'} \geq l_{min}$.
- Let $h_{pp'}/h'_{pp'}$ be the minimum/maximum altitude of segment $(p,p') \in A_{ij}$. It is ensured that the following conditions are satisfied: (1) $h_{pp'} \geq h_{min}$ and (2) $h_{pp'} \leq h_{max}; \forall (p,p') \in A_{ij}$ where h_{min} and h_{max} x are the minimum and

maximum altitude, respectively, which are set forward by FAA and are required to be obeyed even under emergency situations.

- Let $P_{pp'}$ be the flying power required for moving drone through a segment $(p, p') \in A_{ij}$. An eligible segment ensures that $P_{pp'} \leq P_{max}$ where P_{max} denotes the maximum flying power for a drone.
- Cross wind velocity, $w_{pp'}$, impacts drones forward velocity, $u_{pp'}$, which is also a function of maximum deviation angle, $\theta_{pp'}$, for each segment $(p, p') \in A_{ij}$. It is ensured that condition $u_{pp'} \geq \frac{w_{pp'}}{\sin(\theta_{pp'})}; \forall (p, p') \in A_{ij}$ is maintained to be considered for an eligible segment. Figure 2.2(a) depicts the effect of cross wind velocity $w_{pp'}$ on drones forward velocity $u_{pp'}$ through segment pp'.
- Turning angle is a crucial element which is usually required to be low in order to conserve energy in drone transportation. Environmental factors and initial starting point have significant impact on the number of turns and the size of turning angles of drones. The maximum turning angle, $\theta_{pp'}$, of a drone is determined based on the coordinates of segment $(p, p') \in A_{ij}$, i.e., $(x_{pp'}, y_{pp'}, z_{pp'})$ as follows $\frac{g_{pp'}^T g_{pp'+1}}{|g_{pp'}||g_{pp'+1}|} \geq \cos(\theta'_{pp'}); \forall (p, p') \in A_{ij}$ where $g_{pp'} = ((x_{pp'} - x_{pp'-1}), (y_{pp'} - y_{pp'-1}))$ and $|g_{pp'}|$ is the length of $g_{pp'}$. Figure 2.2(b) demonstrates the concept of turning angle through a segment from point p to point c and then to point p' for a drone transportation.



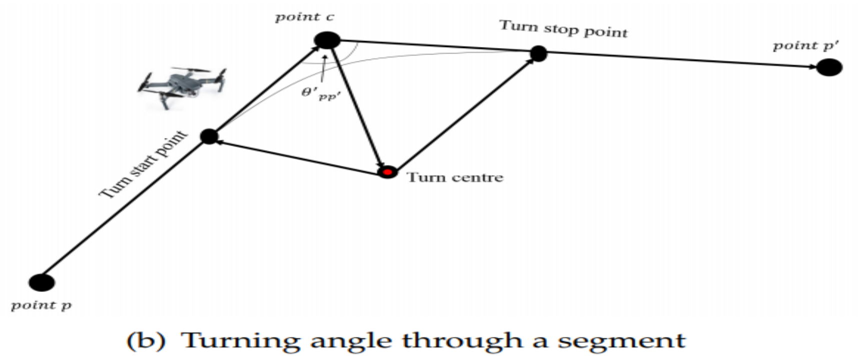


Figure 2.2 Effects of drone forward velocity and turning angle when moving through a segment

After identifying the eligible segments, using the potential points between each pair of nodes, i.e., $\forall(i,j) \in V|i < j$, optimal paths between nodes i and j are determined based on a path construction model introduced below. However, before introducing the model, let us first introduce decision variable $X' \coloneqq \{X_{pp'}\}$ which is equal to 1 if eligible segment (p,p') is considered for path (i,j); 0 otherwise. The following path construction model is applied to determine the optimal paths between each pair of nodes $(i,j) \in V|i < j$.

$$PC_{ij} \coloneqq \sum_{(p,p') \in A_{ij}|p \neq p'} 2c_{pp'}((\frac{I_{pp'}}{u_{pp'}})P_{pp'})\, X_{PP'} \tag{17}$$

Subject to

$$\sum_{p \in A_{ij}} X_{pp'} = \sum_{p' \in A_{ij}|p \neq p'} X_{p'p} \quad \forall p' \in A_{ij} \backslash \{i,j\} \tag{18}$$

$$\sum_{p \in A_{ij}|p \neq i} X_{ip} = 1 \tag{19}$$

$$\sum_{p \in A_{ij}|p \neq j} X_{pj} = 1 \tag{20}$$

$$\sum_{p \in A_{ij}|p \neq p'} X_{pp'} \leq 1 \qquad \forall p' \in A_{ij} \tag{21}$$

$$\sum_{p' \in A_{ij}|p' \neq p} X_{pp'} \leq 1 \qquad \forall p \in A_{ij} \tag{22}$$

$$\sum_{(p,p') \in A_{ij}|p \neq p'} I_{pp'} X_{pp'} \leq l_{ij} \tag{23}$$

$$\sum_{(p,p') \in A_{ij}|p \neq p'} t_{pp'} X_{pp'} \leq t_{ij} \tag{24}$$

$$X_{pp'} \in \{0,1\} \qquad \forall (p,p') \in A_{ij} \tag{25}$$

The objective function (17) minimizes the battery consumption costs, $\left(2c_{pp'}\left(\frac{I_{pp'}}{u_{pp'}}\right)P_{pp'}\right)$, associated with constructing segments between each pair of nodes $(i,j) \in V|i < j$. Constraints (18) are the standard flow balance constraints which connect one segment to the next. Constraints (19) and (20) connect segment $p \in A_{ij}$ with the source node $i \in V$ and destination node $j \in V$. Constraints (21) and (22) ensure that no more than one in-degree and one out-degree segment are permissible in a point. Constraints (23) and (24) indicate that the length and flying time through segments $(p, p') \in A_{ij} | p \neq p'$ should be restricted by the maximum length, l_{ij}, and flying time t_{ij}, of path $(i, j) \in V$, respectively. Finally, constraints (25) set the binary restrictions.

Variable Fixing and Valid Inequalities

By fixing the values of some decision variables and adding valid inequalities, we attempt to accelerate the computational performance of model [DP]. Let us first discuss few variable fixing techniques, which are applied when a particular condition(s) is satisfied.

- Drone $k \in K$ is not capable of flying through a trajectory connecting node $i \in V$ to node $j \in V$ when its traveling time, i.e., $\frac{d_{ij}}{\bar{v}_{ks}}$, is equal or greater than the maximum allowable travel time t_k^{max}, under its maximum speed level $s \in S$.

$$X_{ijk} = 0 \; \& \; Z_{ijks} = 0 \; \forall k \in K, (i, j) \in V | i \neq j, s = s^{max} | (\frac{d_{ij}}{max_{s \in S}\{\bar{v}_{ks}\}}) \geq t_k^{max} \qquad (26)$$

- Drone $k \in K$ is not capable of moving through a trajectory connecting node $i \in V$ to node $j \in V$ when its battery consumption, i.e., $r_{ks} d_{ij}$, , is equal or greater than its battery capacity q_k, under its minimum speed level $s \in S$.

$$X_{ijk} = 0 \; \& \; Z_{ijks} = 0 \; \forall k \in K, (i, j) \in V | i \neq j, s = s^{min} | (min_{s \in S}\{r_{ks}\} d_{ij}) \geq q_k \qquad (27)$$

In addition to fixing variables, the following valid inequalities are added to model **[DP]**.

- The upper bound of the number of nodes visited by drone $k \in K$, i.e., UB_k, is determined in terms of trajectories' distances amongst nodes. Define set D as a non-decreasing order of $d_{ij}, \forall (i, j) \in V | i \neq j$, where $|d_m|$ represents the value of the m^{th} member of set $D, m \leq |D|$.

$$\sum_{(i,j) \in V | i \neq j} X_{ijk} \leq UB_k \quad \forall k \in K \; | \sum_{m=1}^{UB_k} |d|_m \leq t_k^{max} \qquad (28)$$

- Symmetries may result due to the fact that a set of same dummy battery recharging stations may be visited by drones multiple times. To alleviate this problem, the following lexicographical ordering constraints can be applied for each $\mathcal{F}'_\Phi$ where $\mathcal{F}'_\Phi$ can be defined as a subset of battery recharging stations, i.e., $\mathcal{F}'_\Phi \subset \mathcal{F}_\Phi$, related to the same physical station and $j'_g \subset \mathcal{F}'_\Phi$, where $g \in \{1, \dots, |\mathcal{F}'_\Phi|\}$ represents a set of non-decreasing order of the members belonging to $\mathcal{F}'_\Phi$. The goal is to provide a priority of utilizing dummy recharging stations for the drones.

$$\sum_{i\in I} iX_{ij'_gk} \geq \sum_{i\in I} iX_{ij'_{g+1}k} \qquad \forall k \in K, g \in \{1, \dots, |\mathcal{F}'_\Phi| - 1\} \tag{29}$$

- Drones of the same type are deployed to inspect disaster-affected nodes by obeying the technological, geographical, and environmental restrictions set forward by FAA. Define K' as a subset of drones belonging to the same type, i.e., $K' \in K$, and $k'_g \subset K'$, where $g \in \{1, \dots, |K'|\}$ which represents a set of non-decreasing order of the members belonging to K' to determine the priority of utilizing drones of the same type.

$$X_{0jk'_g} \geq X_{0jk'_{g+1}} \qquad \forall j \in V, g \in \{1, \dots, (|K'| - 1)\} \tag{30}$$

$$\frac{\left(\sum_{s\in S}\sum_{(i,j)\in V|i\neq j} d_{ij} Z_{ijk'_gs}\right)}{\bar{v}_{k'_gs}} \geq \frac{\left(\sum_{s\in S}\sum_{(i,j)\in V|i\neq j} d_{ij} Z_{ijk'_{g+1}s}\right)}{\bar{v}_{k'_{g+1}s}} \ \forall\, g \in \{1, \dots, |K'|-1\} \tag{31}$$

$$\sum_{s\in S}\sum_{(i,j)\in V|i\neq j} c_{ijk'_gs} \left(\left(\beta_{k'_g} + \lambda_{k'_g} h_{k'_g}\right)\left(\frac{d_{ij}}{\bar{v}_{k'_gs}}\right) + p^{turn}_{k'_gs}\left(\frac{n_{ij}\bar{\theta}_{ij}}{w^{turn}_{k'_gs}}\right) + \right.$$

$$\left(\sqrt{\left(\Gamma_{k'_g}\bar{v}_{k'_gs}p^{acc}_{k'_g}\right)}\, g_1 + \left(\frac{2p^{int}_{k'_g}}{\bar{v}_{k'_gs}}\right) g_2 + \sqrt{\left(\Gamma_{k'_g}\bar{v}_{k'_gs}p^{dec}_{k'_g}\right)}\, g_3\right) d_{ij}\Big) Z_{ijk'_gs}\Big) \geq$$

$$\sum_{s\in S}\sum_{(i,j)\in V|i\neq j} c_{ijk'_{g+1}s} \left(\left(\beta_{k'_{g+1}} + \lambda_{k'_{g+1}} h_{k'_{g+1}}\right)\left(\frac{d_{ij}}{\bar{v}_{k'_{g+1}s}}\right) + p^{turn}_{k'_{g+1}s}\left(\frac{n_{ij}\bar{\theta}_{ij}}{w^{turn}_{k'_{g+1}s}}\right) + \right.$$

$$\left(\sqrt{\left(\Gamma_{k'_{g+1}}\bar{v}_{k'_{g+1}s}p^{acc}_{k'_{g+1}}\right)}\, g_1 + \left(\frac{2p^{int}_{k'_{g+1}}}{\bar{v}_{k'_{g+1}s}}\right) g_2 + \right.$$

$$\sqrt{\left(\Gamma_{k'_{g+1}}\bar{v}_{k'_{g+1}s}p^{dec}_{k'_{g+1}}\right)}\, g_3\Big) d_{ij}\Big) Z_{ijk'_{g+1}s} \qquad \forall\, g \in \{1, \dots, |K'|-1\} \tag{32}$$

CHAPTER III

SOLUTION METHODOLOGY

The initial computational experiences expose our inability to solve model **[DR]** in largescale problem settings. However, there is a craving need to solve large-scale instances of model **[DR]** in a reasonable amount of time to perform a quick initial inspection of any disaster-affected region. To serve this purpose, two solution algorithms, known as an Adaptive Large Neighborhood Search (ALNS) algorithm and a Modified Backtracking Adaptive Threshold Accepting (MBATA) algorithm, are proposed in this section. Before delving into the details of the proposed algorithms, we discuss an initial route generation mechanism that can be applied in both ALNS and MBATA algorithm.

Initial Routes Generation Mechanism

Our study utilizes nearest neighbor heuristic algorithm [28] to generate initial feasible routes for the drones. Contrary to using random initial routes (IR), the algorithm is capable of generating high quality initial feasible routes which may help in reducing the overall computational time of the ALNS/MBATA algorithm. The algorithm proceeds by gradually adding arcs starting from drone depot to all nearest disaster-affected nodes within a determined range. The insertion of a disaster-affected node is determined based upon the allowable travel time and remaining battery level of drone $k \in K$ (i.e., $t_k^{max}$and Yik, respectively) as well as the availability of battery recharging stations within the determined range. The next inserted node in a route can be a disaster-affected node, a battery recharging station, or drones depot based upon arrival time to the node, τ_{ik}, and remaining battery level, Y_{ik}, respectively. The process is repeated to construct initial feasible routes for all the remaining drones $k \in K$.

Adaptive Large Neighborhood Search (ALNS) Algorithm

The Adaptive Large Neighborhood Search (ALNS) Algorithm, first introduced by Ropke and Pisinger [29], belongs to the class of very large-scale neighborhood search algorithm [30]. The authors essentially extended the large neighborhood search algorithm developed by Shaw [31], while allowing the possibility of implementing multiple destruction and insertion operators simultaneously to generate high-quality solutions.

ALNS algorithm proceeds by defining a set of destruction and insertion operators in an attempt to develop high quality routes for the drones. Amongst a set of destruction and insertion operators, only one type of destruction operator, i.e., $o_d \in D,$ along with one type of insertion operator, i.e., $o_r \in R,$ are selected to generate a new solution in each iteration of the ALNS algorithm. Note that in each iteration of the ALNS algorithm, the choice of different types of o_d and o_r operators is non-trivial since they are crucial for the success of the ALNS algorithm. This selection is governed by Roulette Wheel Selection Mechanism proposed by Ropke and Pisinger [29], where each operator $\frac{o_d}{o_r}$ has a weight $\frac{w_{o_d}}{w_{o_r}}$ and an score $\frac{\pi_{o_d}}{\pi_{o_r}}$ which is obtained from the prior performance of the operator. Roulette wheel mechanism is the most common and widely used selection mechanism which assigns a selection probability to each member of an operator proportional to its relative fitness. Irrespective to the type of the selection mechanism employed, p_{o_d}, belonging to operator type o_d, is calculated as $p_{o_d} \coloneqq \frac{w_{o_d}}{\sum_{d \in D} w_{o_d}}$, where each operator is assumed to occupy some space in a pie graph proportional to its fitness value, i.e., probability of selection. Likewise, $p_{o_r} \coloneqq \frac{w_{o_r}}{\sum_{r \in R} w_{o_r}}$. In the first iteration of the ALNS algorithm, all operators are assumed to be equally likely, i.e., $p_{o_d} \coloneqq \frac{1}{|D|}$; $\forall d \in D$ and $p_{o_r} \coloneqq \frac{1}{|R|}$; $\forall r \in R$ along with zero score, i.e., $\pi_{o_d} = 0$; $\forall d \in D$ and $\pi_{o_r} = 0$; $\forall r \in R$.

At ith iteration of the ALNS algorithm, $p_{o_z}^i (z \in \{d, r\})$ is updated as follows: $p_{o_z}^i \coloneqq p_{o_z}^{i-1}(1-\eta) + \eta \left(\frac{\pi_{o_z}}{w_{o_z}}\right)$, where η is a reaction factor generated from a uniform distribution, $\eta \in [0,1]$, to show a reaction to changes on the operator performance. In addition to updating the probability of selecting each operator in each iteration of the ALNS algorithm, $\pi_{o_z}(z \in \{d, r\})$ is updated as follows: the score is increased by σ_1 when the implementation of the operator results in a global best solution; otherwise, the score is increased by $\frac{\sigma_2}{\sigma_3}$ when the fitness value of the obtained solution is better/worse than the the fitness value of the solution generated in the previous iteration. The relationship between different amounts of increase in the score is $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$. Two terminations criteria are adopted to terminate the ALNS algorithm as (1) the maximum iteration limit (max_{itr}) and (2) the maximum computational time limit (max_{CT}).

ALNS Operators

We now provide a brief description of different types of destruction and insertion operators used in the ALNS algorithm. A destruction operator removes a visited disaster-affected node(s) from a route(s), while an insertion operator inserts a disaster-affected node(s) into a route(s). A removal list for visited disaster-affected nodes, $L_{Removal}$, and a return list for unvisited disaster-affected nodes, L_{Return}, are developed for destruction and insertion operators, respectively, where $L_{Removal} \cap L_{Return} = \emptyset$. It is worth noting that all disaster-affected nodes (IR) are inserted into L_{Return} before implementing the operators in the ALNS algorithm, i.e., $L_{Return} = I, L_{Removal} = \emptyset$. After each iteration, the new solution is determined based on updated L_{Return}. In the following, five destruction and four insertion operators are discussed that can be used within the ALNS algorithm.

Destruction Operators:

Random removal (RR): This operator simply chooses p visited disaster-affected nodes randomly from L_{Return} according to a discrete uniform distribution, U[1,|$(L_{Return})^{Pr}$|], inserts them into $L_{Removal}$, and updates L_{Return}. Here, $(L_{Return})^{Pr}$ is the set of visited disaster-affected nodes determined in the previous iteration of the algorithm. The value chosen for $p(p < |I|)$ has an impact on the number of iterations of the ALNS algorithm. The random selection of visited disaster-affected nodes increases the diversification into the search mechanism.

Worst removal (WR): This operator removes visited disaster-affected nodes from LReturn with high-visited costs in a hope of replacing them by disaster-affected nodes with low-visited costs from $L_{Removal}$. Then, both L_{Return} and $L_{Removal}$ are updated. This operator simply chooses q visited disaster-affected nodes randomly from L_{Return} according to a non-increasing order of visited costs, inserts them into $L_{Removal}$, and updates L_{Return}. More precisely, the visited cost of a disaster-affected node is defined as a difference between the cost when the disaster-affected node is inserted into a route (i.e., visited) and removed from that route (i.e., unvisited). Then, this visited cost is normalized by dividing it to the average visited cost of all possible disaster-affected nodes which can be visited on this route (equal or less than q disaster-affected nodes). The purpose of this normalization is to avoid repeatedly choosing disaster-affected nodes located far away from the remaining ones.

Route removal (RR): This operator removes all disaster-affected nodes visited on a route from L_{Return}. The target route is selected randomly from the set of developed routes. Then, insert all disaster-affected nodes visited on this route into $L_{Removal}$ and update both L_{Return} and $L_{Removal}$.

History-based removal (HKR): This operator is similar to the neighbor graph removal operator introduced by Ropke and Pisinger [32]. This operator preserves a memory of the position-based value for each disaster-affected node $i \in I$ which is determined as the sum of the distances between disaster-affected node $i \in I$ with its preceding and following disaster-affected nodes, all visited on the same route. The position-based value of a disaster-affected node i, p_i, can be determined as $p_i \coloneqq d_{(i-1,i)} + d_{(i,i+1),}$ where $d_{(i,j)}$ represent a euclidean distance between two consecutive disaster-affected nodes i and j visited on a route. This operator is designed in such a way that the best position value of disaster-affected node $i, p_i',$, is updated to be the minimum of all p_i values related to the previous iterations of the algorithm. This operator selects disaster-affected node i' on a route with the maximum deviation from its best position value, i.e., $i' = {}_{i \in L_{Return}}^{argmax}(p_i - p_i')$. Then, disaster-affected node i' is removed from L_{Return} and inserted to $L_{Removal}$. Finally, both L_{Return} and $L_{Removal}$ are updated.

Neighborhood removal (NR): This operator removes a disaster-affected node with the maximum distance deviation from the average distance of its corresponding route. Consider a set of developed routes as R^h in the hth iteration of the algorithm. The average distance of route $r \in R^h$ as a set of n_r disaster-affected nodes sorted by visited regions' numbers, $r = \{i_1, i_2, \dots, i_n\}$, is determined as $\bar{d}_r \coloneqq \frac{(\sum_{(i_1,i_2)\in r | i_1 \neq i_2} d_{(i_1,i_2)})}{(n_r - 1)}$, where $d_{(i_1,i_2)}$is a euclidean distance between two consecutive disaster-affected nodes visited on route r. Then, remove each disaster-affected node j from its route and recalculate $\bar{d}_{r\backslash\{j\}}, \forall r \in R$. Afterwards, a disaster affected node s with the maximum deviation between $\bar{d}_r$ and $\bar{d}_{r\backslash\{s\}}$ is selected, removed from L_{Return}, and inserted to $L_{Removal}$. Finally, both L_{Return} and $L_{Removal}$ are updated.

Insertion Operators:

Greedy insertion (GI): This operator repeatedly inserts disaster-affected nodes of $L_{Removal}$ in the best possible position of existing routes based upon its insertion cost which is calculated as $\underset{i \in L_{Removal,}(j,k) \in L_{Return}}{argmin} (p'_{jik})$ and $p'_{jik} \coloneqq d_{(j,i)} + d_{(i,k)} - d_{(j,k)}$, where j an k are consecutive disaster-affected nodes visited on the same route before inserting disaster-affected node i between them. In other words, j and k are disaster-affected nodes visited immediately before and after inserted disaster-affected node i on the developed route.

Greedy insertion with noise function (GIN): Although this operator performs similar to greedy insertion operator, it considers a degree of freedom as a noise term. The insertion cost based on this degree of freedom is expressed as $p'_{jik} \coloneqq d_{(j,i)} + d_{(i,k)} - d_{(j,k)} + \alpha\rho\delta$, where α is the maximum euclidean distance between disaster-affected nodes j, i, and k; ρ is a noise parameter used for the diversification which is set to 0.1; and δ is a random number obtained from $[-1, 1]$.

Regret insertion (RI): For each disaster-affected node i of $L_{Removal}$, δf_i^j designates the insertion cost of disaster-affected node i visited on the jth best route at its best position. Disaster-affected node i' is chosen as $i' = \underset{i \in L_{Removal}}{argmax}(\delta f_i^1 - \delta f_i^2)$, where δf_i^1 and δf_i^2 are the first and second best insertion cost of disaster-affected node i visited on the first and second best route, respectively, at their best positions. Then, both $L_{Removal}$ and L_{Return} are updated and this insertion mechanism stops whenever all disaster-affected nodes are inserted to the routes, i.e., $L_{Removal} = \emptyset$.

Regret insertion with noise (RIN): Similar to greedy insertion with noise function, this operator is an extension of the regret insertion operator using the same noise function described in the GIN operator.

After implementing any operator described above, the developed/current routes might be updated as follows:

- By removing a disaster-affected node(s) from a route(s), the necessity of visiting an existing battery recharging station(s) on the route is surveyed. If q_k of k^{th} drone assigned to the route is capable of visiting remained disaster-affected nodes on the route without visiting one or more battery recharging stations, those stations can be removed from the route.
- By inserting a disaster-affected node(s) to a route(s), the feasibility of the developed route(s) must be checked with respect to t_k^{max} and q_k of k^{th} drone assigned to the developed route. If drone $k \in K$ is not capable of visiting all

disaster-affected nodes on the developed route due to violation on t_k^{max}, the developed route is ignored. In addition, if drone $k \in K$ is not capable of visiting all disaster-affected nodes on the developed route due to violation on q_k, one or more battery recharging stations are inserted into the developed route to satisfy the drone requirement.

The pseudo-code of the ALNS algorithm is provided in Algorithm 1.

Algorithm 1: ALNS

```
Input: $max_{itr}, max_{CT}, \omega_{o_z}, \pi_{o_z};\ \forall z \in \{r,d\}, r \in \mathcal{R}, d \in \mathcal{D}$;
Output: Best developed routes ($S^{best}$);
Initialize $\sigma_1$, $\sigma_2$, and $\sigma_3$;
$P_{o_r} := 1/|\mathcal{R}|;\ \forall r \in \mathcal{R}\ \ \&\ \ P_{o_d} := 1/|\mathcal{D}|;\ \forall d \in \mathcal{D}$;
$\pi_{o_r} := 0;\ \forall r \in \mathcal{R}\ \ \&\ \ \pi_{o_d} := 0;\ \forall d \in \mathcal{D}$;
$S^{IR} \leftarrow$Route.Initialization();
$S^{best} \leftarrow S^{IR}$;
$S \leftarrow S^{IR}$;
while $i < max_{itr}$ or $CPU \leq max_{CT}$ do
    Select a destruction operator $o_d$ and insertion operator $o_r$ according to
      roulette-wheel mechanism;
    $S' \leftarrow$Destruction.Operator.Implementation($S$);
    $S' \leftarrow$Insertion.Operator.Implementation($S$);
    if $(CF(S') < CF(S))$ then
        if $(CF(S') < CF(S^{best}))$ then
            $S^{best} \leftarrow S'$;
            Increase the score of the operators by $\sigma_1$;
        else
            Increase the score of the operators by $\sigma_2$;
        end
    else
        Increase the score of the operators by $\sigma_3$ ;
    end
    Update $\omega_{o_z}\ \forall z \in \{r,d\}, r \in \mathcal{R}, d \in \mathcal{D}$;
    Update $P_{o_r} := w_{o_r}/\sum_{r\in\mathcal{R}} w_{o_r};\ \forall r \in \mathcal{R}\ \ \&\ \ P_{o_d} := w_{o_d}/\sum_{d\in\mathcal{D}} w_{o_d};\ \forall d \in \mathcal{D}$;
    Update $P^{i}_{o_z} := P^{i-1}_{o_z}(1-\eta) + \eta(\frac{\pi_{o_z}}{w_{o_z}});\ \forall z \in \{r,d\}, r \in \mathcal{R}, d \in \mathcal{D}$;
    $S \leftarrow S'$;
    $i \leftarrow (i+1)$;
end
return $S^{best}$
```

Modified Backtracking Adaptive Threshold Accepting (MBATA) Algorithm

The Modified Backtracking Adaptive Threshold Accepting (MBATA) algorithm is inspired from the works proposed by Tarantilis et al. [33, 34], where a non-monotonic schedule is established inside the traditional threshold accepting algorithm. The MBATA algorithm applies and explores most of the important features of the basic backtracking adaptive threshold accepting (BATA) algorithm including threshold parameter, backtracking, move evaluation, strategic oscillation, setting acceptance criteria, intensification, and diversification. However, the unique difference between the two algorithms i.e., BATA and MBATA algorithm, is related to the shaking mechanism implemented inside the MBATA algorithm. After finding the best solution and towards the end of the search process, shaking the entire solutions might be the only way for not being trapped in local optima and, consequently, generate better results. To summarize, the MBATA algorithm operates at two dependent loops moving back and forth as follows: an inner loop to diversify into the solution space; an outer loop to intensity the quality of the solutions. In the following, some features of the MBATA algorithm are explained in details.

Local Search

Local search is performed in an inner loop of the MBATA algorithm. Defining the neighborhoods play a crucial role in local search since they determine the extent of the solution space explored. The termination of the local search in each iteration of the algorithm is determined based on the maximum number of iterations, max_{inner}, in the inner loop. We now discuss four different types of inter-route and intra-route moves which are applied in the local search operation.

Inter-route moves:

The inter-route neighborhood structures include a single-insertion, double-insertion, triple-insertion, and swap moves. These moves are capable of converting a route to another one as well as eliminating criss-crosses of edges related to the same or different routes. A brief description of these inter-route moves are provided below.

Single insertion: A single node $i \in V$ is randomly selected and removed from its current route. Afterwards, a trial insertion of node $i \in V$ is determined on the remaining routes containing either the closest or furthest insertion points of node $i \in V$. The selection of the closest or the furthest node depends on a Bernoulli distribution. Although this concept has been adopted by several researches [34], most authors use a parameter δ as a maximum distance measure so that node $i \in V$ is only inserted into a route with δ unit of distance from a particular node. This partially restricts the diversification into the solution space; hence, in this study, both the minimum and maximum distance measures are used.

Double insertion: Similar to the single insertion move, a double insertion move operates on a pair of consecutive nodes (i, j) belonging to the same route. In other words, two consecutive nodes $i \in V$ and $j \in V$ are inserted into a new route including at least one node which is either closest or farthest to any of inserted nodes

Triple insertion: Likewise, in a triple insertion move, a consecutive chain of three nodes (i, j, k) are selected, removed from their original route, and inserted into the destination route including at least one node which is either closest or furthest to any of the inserted nodes.

Swap: The position of node $i \in V$ from a route is exchanged with the position of node $j \in V$ from a different route. This is only permitted if and only if selected node i is associated to at least one node j, which is either closest or furthest from that node.

Intra-route moves:

Following any possible improvement by implementing an inter-route move(s), the inner loop implements four well-known intra-route neighborhood moves including reinsertion, exchange, Or-opt2, and Or-opt3 [35,36]. In the following, these moves are briefly described.

Reinsertion: One node is removed from its current position in a route and inserted into another position of the same route.

Exchange: The position of a node in a route is changed with the position of another node in the same route.

Or-opt2: Two adjacent nodes are removed and inserted into another position of the same route.

Or-opt3: Three adjacent nodes are removed and inserted into another position of the same route.

After implementing any inter-or intra-route move on a particular iteration of the local search, the newly generated solution $s'$is compared with the solution generated in the previous iteration, s, i.e., c(s') − c(s) < T_h where c(s) and c(s') represents the objective function value of s and s', respectively and T_h represents a predefined threshold value at h^{th} iteration of the algorithm (T_h > 0). If this condition is satisfied, then s is replaced by s'. However, if c(s') is less than c(s^{best}), $s^{best}$is set to s' , where s^{best} is the best solution obtained so far during the iterations of the algorithm. On the other hand, if c(s') − c(s) > T_h, no changes are made on s, s^{best}, and s'. Once an improved solution is found, then intra-route moves are implemented.

Important Notes:

- All the inter- and intra-route moves are not applied for each iteration of the MBATA algorithm since it not only increases the computational time, but also might lead to unwarranted diversification of the solution space.

Hence, these moves are implemented randomly by the local search of the inner loop.

- Any selected node(s) removed from its original route is inserted into another route, either after/before drones depot or between any two nodes of the selected route. In addition, in relation to the double and triple insertion moves of inter-route moves as well as the Or-opt2 and Or-opt3 moves of intra-route moves, different orders of the position of selected nodes in the new route provide more neighbor solutions.
- If an inter- or intra-route move(s) results in an infeasible or lower-quality solution, this move is discarded and another move is randomly chosen. This procedure continues until a better solution is generated or the maximum number of iterations for applying inter- and intra-route moves is reached, i.e., max_{inter} and max_{intra}, respectively.
- An infeasible solution is a solution which violates the maximum allowable travel time and/or battery capacity of drones.
- An inserted node(s) in a particular position of a route is a disaster-affected node(s), a battery recharging station(s), or a combination of both.

Threshold Controlling Procedure

A threshold controlling procedure is performed on the outer loop of the MBATA algorithm which is depending upon the solutions quality obtained from the inner loop. The value of threshold T_h is decreased if $c(s') - c(s) < T_h$ is satisfied at least once during the iterations of the inner loop. In other words, a solution has been found to be better than the current solution at least once. On the other hand, if no improvement is obtained, the value of T_h is increased or backtracked. However, during the backtrack step, the amount of increase of T_h is always preserved to be smaller compared to the amount of last increase in $T_{h'} (\forall h' \in \{1,2,\ldots,h-1\})$, before the backtrack step. This ensures that there is a very tight balance between the diversification and intensification on the solution space.

If $c(s') - c(s) < T_h$ and $c(s') \leq c(s)$ are satisfied for at least one iterations of the inner loop, T_{h+1} is decreased and set as $T_{h+1} \coloneqq T_h * T_t$; otherwise, if $c(s') - c(s) < T_h$ and $c(s') > c(s)$ for all iterations of the inner loop, T_{h+1} is increased and set as $T_{h+1} \coloneqq T_h * (1 + T_i)$. Note that T_t and T_i are the reduction and increase rates of T_h, where $\{T_t, T_i\} \in [0,1]$. If T_h is not able to provide a better solution at least once in one of the iterations of the inner loop, i.e., $c(s') - c(s) > T_h$, the value of T_{h+1} is increased (backtracked) close to its previous value. This new value is represented as $T_{h+1} \coloneqq T_{h-1} - (T_{h-1} - T_h) * (1 - T_b)$, where T_b is the percentage of threshold backtracking $(T_b > 0)$. The termination of the MBATA algorithm is determined based on the maximum number of iterations of the outer loop i.e., max_{outer}.

Shaking

The shaking technique is applied when two consecutive outer loops failed to generate a better solution. This technique has been applied extensively by many researchers in Variable Neighborhood Search (VNS) algorithm when a local optimum is found within a given neighborhood [37]. This technique is applied by significantly transferring the solution space to another area in the hope of obtaining a better solution;

otherwise, the new solution moves towards the original solution space in the prosecution of the search. However, too many transformation of the solution space might potentially lead to an inefficient solution space, where it would be challenging to find good-quality solutions. Hence, a trade-off between these two extremes is to perform the shaking technique in such a way that it does not fundamentally change the solution space while not to be trapped in local optima. In this study, the shaking technique is performed by randomly removing a subset of nodes from their current positions to different routes and then reinserting them to the existing routes based on the minimum insertion cost. This cost is referred to the disaster-affected nodes visited at the minimum cost. It is worth noting that the shaking technique is performed even if an generated route(s) is found to be infeasible. This being the case, the position reassignment of nodes to routes is necessary to ensure the feasibility of the developed routes. The pseudo-code of the MBATA algorithm is provided below.

Algorithm 2: MBATA

```
Input: $max_{inner}$, $max_{outer}$, $max_{inter}$, and $max_{intra}$;
Output: Best developed routes ($S_{best}$);
Initialize $T_1$;
$S^{IR} \leftarrow$ Route.Initialization();
$S^{best} \leftarrow S^{IR}$;
$S \leftarrow S^{IR}$;
while ($h < max_{outer}$) do
    while ($k < max_{inner}$) do
        while ($i < max_{inter}$) do
            $S' \leftarrow$ Inter.Route.Move.Implementation($S$);
            if $c(s') - c(s) < T_h$ then
                $s \leftarrow s'$;
                if $c(s) < c(s_{best})$ then
                    $s_{best} \leftarrow s$;
                end
                while ($j < max_{intra}$) do
                    $S' \leftarrow$ Intra.Route.Move.Implementation($S$);
                    if $c(s') - c(s) < T_h$ then
                        $s \leftarrow s'$;
                        if $c(s) < c(s_{best})$ then
                            $s_{best} \leftarrow s$;
                        end
                    else
                        continue;
                    end
                end
            else
                continue;
            end
        end
    end
    if $\big(c(s') - c(s) < T_h \ \ \& \ \ c(s') \leq c(s)\big)$ during at least one iteration of inner loop then
        $T_t \leftarrow U[0,1]$;
        $T_{h+1} \leftarrow T_h \times T_t$;
    end
    if $\big(c(s') - c(s) < T_h \ \ \& \ \ c(s') > c(s)\big)$ during all iteration of inner loop then
        $T_i \leftarrow U[0,1]$;
        $T_{h+1} \leftarrow T_h \times (1 + T_i)$;
    end
    if $\big(c(s') > T_h + c(s)\big)$ during all iterations of inner loop then
        $T_{h+1} \leftarrow T_{h-1} - (T_{h-1} - T_h) \times (1 - T_b)$;
    end
    if $s^{best}$ is not updated during two consecutive iterations of outer loop then
        $s \leftarrow$Shaking.Implementation($s$);
    end
end
```

CHAPTER IV

COMPUTATIONAL STUDY

The exposition of this section is as follows. First, a brief description of the data used to generate different test instances is provided. Second, the computational performances of the proposed ALNS and MBATA algorithms in solving model [DR] over GUROBI are provided. Finally, a real-life case study to demonstrate the applicability of the proposed optimization model to serve a disaster-affected county in Mississippi is provided. Additionally, sensitivity analyses are carried out to study the impact of battery recharging station distributions, maximum travel time and required recharging time of drones as well as drones' depot location on the overall routing cost are analyzed. All numerical experiments are coded in Python 2.7 on a desktop computer equipped with an Intel Core i7 processor 3.60 GHz and a 16 GB RAM. The optimization solver used is GUROBI Optimizer 7.0 [38].

Data Description

Drone Parameters: In this study we consider two types of drones: (1) Vanguard (shown in Figure 4.1(a)[2]) and (2) Mavic Pro (shown in Figure 4.1(b)[3]). Although Vanguard is a long range and high endurance drone over Mavic Pro, this drone is approximately 40 times more expensive than Mavic Pro. However, both drones are

[2] Available from: https://www.airbornedrones.co/vanguard/

[3] Available from: https://www.dji.com/

equipped with high resolution cameras (4K camera) that make them suitable for disaster-affected region monitoring and surveillance operations. Table 4.1 provides the characteristics of two types of drones considered in this study.

Table 4.1 Attributes of utilized drones [39, 40]

Characteristic	Vanguard	Mavic Pro	Unit
Maker	Airborne drones	DJI	-
Speed	40	40	mph
Range	22.0	4.3	mile
Endurance	94	27	Min
Maximum altitude	15,000	16,000	ft
Weight	2.0	0.7	kg

In addition to specifying the characteristics of drones, the following parameters are set in the path construction model (discussed in Section 2.2) to construct paths between each pair of nodes: $\theta_{pp'} = 45^0$; $\theta'_{pp'} = 30^0$; $p^{max} = 75{,}000\ W$; $l_{min} = 0.1\ mile$; $h_{min} = 200\ ft$; $h_{max} = 400\ ft$; and $t_{ij} = 40\ min$. Surrounding environment parameters $w_{pp'}(mph)$ and $u_{pp'}(m/s)$ are set based on the weather condition on each segment. For the sake of simplicity, the maximum deviation and turning angles, cross wind velocity, and drone forward velocity are considered same for all eligible segments between each pair of disaster-affected nodes. Furthermore, Table 4.2 summarizes the key drone input parameters used in model **[DR]**. We set the average drone velocities $\bar{v}_{ks} = \{20,30,35\}(mph)$ under three different types of speed levels $s \in S$. For the sake of simplicity, the battery consumption rate $\left(c_{ijks}\right)$, angular velocity (w_{ks}^{turn}), and required power for each turn (p_{ks}^{turn}), are considered constant for each speed level $s \in S$. Note that choosing a payload for each type of drones, Γ_k, guarantee to carry the camera(s) in relation to the standard operations of the drones. Finally, we set the acceleration phase

(g_1), intermediate phase (g_2), and deceleration phase (g_3) to be 5%, 90%, and 5%, respectively, for each trajectory between each pair of disaster-affected nodes.

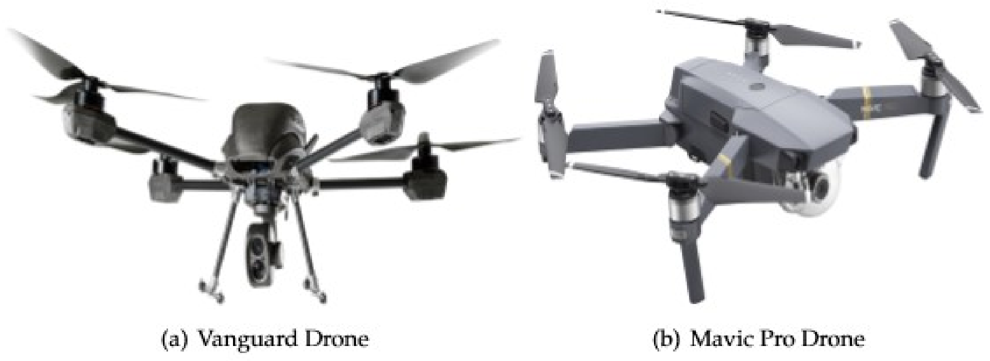


Figure 4.1 The types of drone used for monitoring/inspecting disaster-affected region

Illustration of Geographical Location: Our study uses Hancock county from Mississippi State as a test bed to visualize and validate the modeling results. Hancock county, one of the highly disaster-affected coastal counties in Mississippi along with Harrison and Jackson counties, has been significantly impacted by Hurricanes and Tornadoes over the last few decades [4]. Therefore, in our study we investigate how drones can be economically routed for performing a preliminary damage assessment for the Hancock county following a natural catastrophe. Figure 4.2 shows the geographical locations of Mississippi State, three aforementioned counties, and selected county as the base case for analysis.

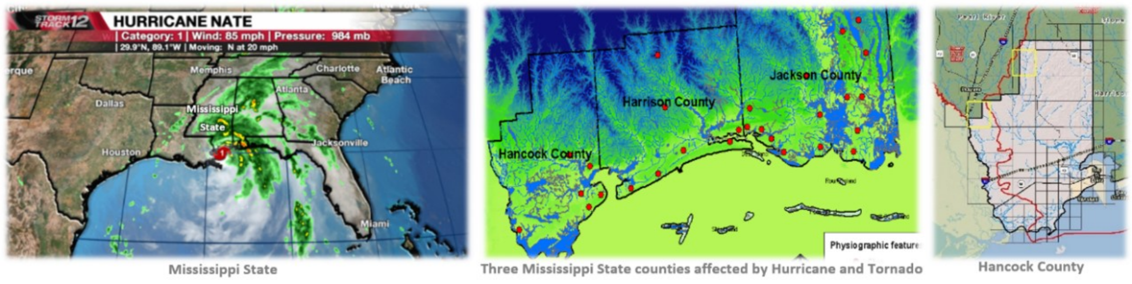

HURRICANE NATE
Category: 1 | Wind: 85 mph | Pressure: 984 mb
Mississippi State
Jackson County
Harrison County
Hancock County
Three Mississippi State counties affected by Hurricane and Tornado
Hancock County

Figure 4.2 Illustration for geographical location of Hancock county [4]

For the case study, we randomly distribute 65 disaster-affected nodes and 13 battery recharging stations in the tested region i.e., $|\mathrm{I}|=65$ and $|\mathcal{F}| = 13$. Further, we place the drones depot at the center of the county. Figure 4.3 shows the geographical locations of the disaster-affected nodes, battery recharging stations, and drones depot on Hancock county. We assume 20 Mavic Pro and 10 Vanguard drones are available with 50 minutes and 120 minutes maximum travel time, respectively. The drone recharging time is set to be 5 minutes.

Computational Performance of the Proposed Algorithms

The efficiency and effectiveness of the algorithms proposed previously are evaluated by solving model **[DR]**. In relation to the research problem, there is no benchmark instances available in the literature to evaluate the performance of the proposed algorithms. Hence, a new set of problem instances are generated with respect to three geographical distributions of disaster-affected regions. In the following, the numerical experiments conducted with the newly generated set of problem instances are explained.

Table 4.2 Key parameters

Parameters	Vanguard	Mavic Pro	Unit
m	10	20	-
q_k	5,100	3,830	mAh
ψ_{jk}	5	5	min
e_{jk}	0.012	0.012	\$
f_{jk}	0.01	0.01	\$
c_{ijk}	0.06	0.06	\$
c_{ijks}^*	0.06	0.06	\$
t_k^{max}	120	50	min
$\bar{v}_{ks}^*$	28	28	mph
r_{ks}	1.0	1.0	-
p_k^a	12,000	12,000	W
p_k^d	5,000	5,000	W
h_k	400	400	ft
v_k^a	30	30	mph
v_k^d	20	20	mph
β_k	90,000	90,000	W
λ_k	1.0	1.0	-
w_{ks}^{turn}	2.1	2.1	rad/s
$\bar{\theta}_{ij}$	1.0	1.0	rad
n_{ij}	3	3	-
p_{ks}^{turn}	1,000	1,000	W
Γ_k	2.0	1.2	kg
g_1	5%	5%	-
g_2	90%	90%	-
g_3	5%	5%	-
p_k^{acc}	90,000	90,000	W
p_k^{int}	85,000	85,000	W
p_k^{dec}	70,000	70,000	W

*Average value under three speed levels

Generation of the Problem Instances

Three sets of problem instances are generated for comparison purposes: small-, medium, and large-size instances including 12, 12, and 16 problems, respectively. Note that the ratio between the number of disaster-affected nodes, |I|, to the number of battery recharging stations, |F|, is maintained to be either 2.5 or 5. We locate the depot at the center of the disaster-affected region and randomly distribute the battery recharging stations within the region. Further, the number of drones, $|K|$, and speed levels, $|S|$, are kept fixed in all the generated instances i.e., $|K| = 2$ and $|S| = 3$. Following the same procedure proposed by Solomon [41], disaster-affected nodes are distributed throughout a county based on three distribution functions: a random uniform distribution (R), a clustered distribution (C),

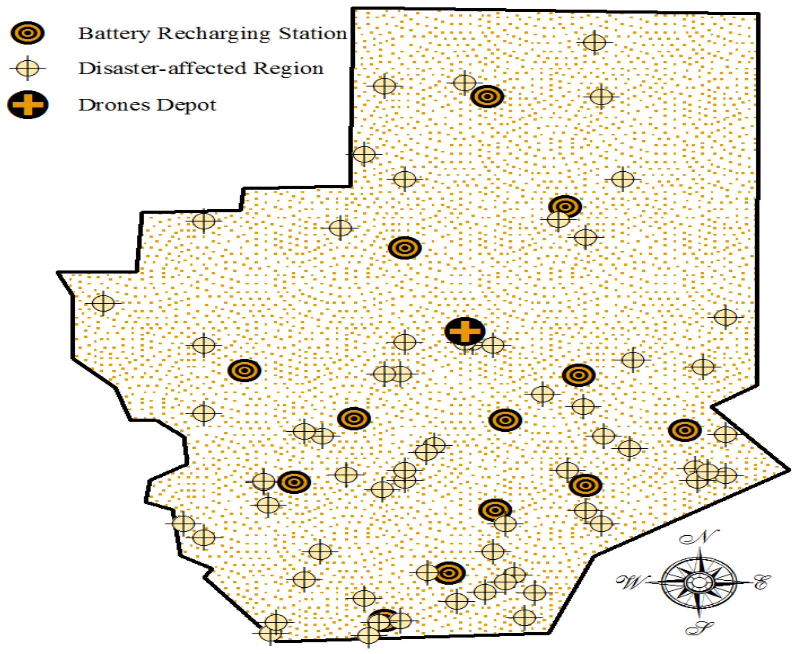


Figure 4.3 Illustration for geographical locations of disaster-affected nodes, battery recharging stations, and drones depot on Hancock county

and a semi-clustered distribution (RC). It is worth noting that RC is a mixture distribution of R and C.

Computational Experiments

This sub-section evaluates the computational performances of the proposed algorithms, ALNS and MBATA, in solving model [DR] over GUROBI. Table 4.3 to 4.5 presents the computational performances of GUROBI, ALNS, and MBATA algorithm in solving model [DR] under three different problem instances: small (Case C1- C12), medium (Case C13-C24), and large (Case C25-C40) instances. The Δf1(%), Δf2(%), and

Δf3(%), reported in Table 4.3 to 4.5, represent the gap between the upper bounds obtained from GUROBI, ALNS, and MBATA to the lower bound obtained from GUROBI, respectively, i.e., $\Delta f_i(\%) = \left(\frac{UB - LB_{GUROBI}}{LB_{GUROBI}}\right) * 100\%,$ where $i = \{\{\text{GUROBI}, \text{ALNS}, \text{MBATA}\}$ and UB and LB stand for upper bound and lower bound, respectively. Further, we set 0.0% optimality gap for GUROBI and 7,200 computational time restriction (in seconds) for both GUROBI and proposed algorithms (ALNS and MBATA). Finally, the boldface letters under T(s) column in Tables 4.3 through 4.5 indicate the best computational time (in seconds) between proposed algorithms and GUROBI.

Results indicate that GUROBI, ALNS, and MBATA are capable of solving all the small and medium-size problem instances under each distribution function i.e., random, clustered, and semi-clustered, within the pre-specified time limit. It is observed that ALNS and MBATA algorithms are 9.41 and 43.31 times faster than GUROBI in relation to smallsize problems and 8.68 and 63.95 times faster than GUROBI in relation to medium-size problems, respectively. Note that these savings in computational times are achieved without sacrificing any solution qualities. Additionally, MBATA algorithm is 4.59 and 7.36 times faster than the ALNS algorithm in relation to small- and medium-size problems, respectively. In overall, we observe that the MBATA algorithm presents superior computational performances in solving small- and medium-size problems of model [DR] over GUROBI and ALNS.

Table 4.3 GOROBI, ALNS, and MBATA performances under small-size problem instances

Disaster-affected node distribution		GUROBI		ALNS		MBATA	
	Case	$\Delta f_1(\%)$	$T(s)$	$\Delta f_2(\%)$	$T(s)$	$\Delta f_3(\%)$	$T(s)$
Random	C1	0.0	9.25	0.0	1.82	0.0	**0.66**
	C2	0.0	9.39	0.0	1.84	0.0	**0.71**
	C3	0.0	76.29	0.0	6.22	0.0	**1.01**
	C4	0.0	76.22	0.0	6.41	0.0	**1.06**
	C5	0.0	152.36	0.0	14.29	0.0	**3.28**
	C6	0.0	158.32	0.0	15.14	0.0	**3.29**
	C7	0.0	198.54	0.0	24.11	0.0	**4.34**
	C8	0.0	202.36	0.0	24.88	0.0	**4.41**
	C9	0.0	399.59	0.0	39.58	0.0	**9.69**
	C10	0.0	416.58	0.0	45.94	0.0	**10.55**
	C11	0.0	543.56	0.0	61.29	0.0	**13.84**
	C12	0.0	549.33	0.0	63.22	0.0	**14.98**
Average		0.0	232.64	0.0	25.39	0.0	**5.65**
Clustered	C1	0.0	8.15	0.0	1.64	0.0	**0.59**
	C2	0.0	8.42	0.0	1.58	0.0	**0.64**
	C3	0.0	68.54	0.0	5.41	0.0	**0.98**
	C4	0.0	71.24	0.0	5.89	0.0	**1.02**
	C5	0.0	146.59	0.0	13.04	0.0	**2.95**
	C6	0.0	151.25	0.0	13.14	0.0	**3.12**
	C7	0.0	188.24	0.0	21.39	0.0	**3.98**
	C8	0.0	191.28	0.0	22.67	0.0	**4.09**
	C9	0.0	387.48	0.0	37.44	0.0	**7.84**
	C10	0.0	392.22	0.0	38.45	0.0	**10.24**
	C11	0.0	540.28	0.0	59.87	0.0	**12.47**
	C12	0.0	542.18	0.0	60.24	0.0	**13.69**
Average		0.0	224.65	0.0	23.39	0.0	**5.13**
Semi-clustered	C1	0.0	8.14	0.0	1.79	0.0	**0.59**
	C2	0.0	8.36	0.0	1.81	0.0	**0.69**
	C3	0.0	71.29	0.0	5.98	0.0	**1.06**
	C4	0.0	73.44	0.0	6.77	0.0	**1.27**
	C5	0.0	148.69	0.0	14.04	0.0	**3.11**
	C6	0.0	150.36	0.0	14.18	0.0	**3.18**
	C7	0.0	191.27	0.0	21.58	0.0	**4.09**
	C8	0.0	193.48	0.0	22.42	0.0	**4.38**
	C9	0.0	398.57	0.0	36.48	0.0	**7.57**
	C10	0.0	399.84	0.0	37.24	0.0	**8.51**
	C11	0.0	540.43	0.0	62.49	0.0	**13.09**
	C12	0.0	561.28	0.0	64.10	0.0	**13.33**
Average		0.0	228.75	0.0	24.07	0.0	**5.07**
Total Average		0.0	228.68		24.28		**5.28**

Table 4.4 GOROBI, ALNS, and MBATA performances under medium-size problem instances

Disaster-affected node distribution		GUROBI		ALNS		MBATA	
	Case	$\Delta f_1(\%)$	$T(s)$	$\Delta f_2(\%)$	$T(s)$	$\Delta f_3(\%)$	$T(s)$
Random	C13	0.0	984.51	0.0	107.49	0.0	**17.18**
	C14	0.0	993.54	0.0	109.86	0.0	**17.22**
	C15	0.0	1,136.84	0.0	147.58	0.0	**24.36**
	C16	0.0	1,144.88	0.0	151.29	0.0	**24.87**
	C17	0.0	1,458.33	0.0	184.36	0.0	**27.59**
	C18	0.0	1,468.29	0.0	188.66	0.0	**27.84**
	C19	0.0	2,145.98	0.0	259.85	0.0	**30.87**
	C20	0.0	2,289.68	0.0	278.52	0.0	**31.22**
	C21	0.0	2,872.48	0.0	331.58	0.0	**44.28**
	C22	0.0	2,998.24	0.0	357.18	0.0	**45.94**
	C23	0.0	3,842.24	0.0	393.54	0.0	**48.94**
	C24	0.0	4,339.25	0.0	428.35	0.0	**52.67**
Average		0.0	2,139.52	0.0	244.85	0.0	**32.74**
Clustered	C13	0.0	989.54	0.0	112.11	0.0	**18.54**
	C14	0.0	1,011.26	0.0	121.22	0.0	**18.98**
	C15	0.0	1,128.58	0.0	146.55	0.0	**22.21**
	C16	0.0	1,139.54	0.0	149.74	0.0	**23.59**
	C17	0.0	1,439.58	0.0	184.38	0.0	**26.88**
	C18	0.0	1,482.66	0.0	195.69	0.0	**28.54**
	C19	0.0	2,144.58	0.0	257.84	0.0	**30.84**
	C20	0.0	2,188.35	0.0	289.54	0.0	**33.28**
	C21	0.0	2,841.19	0.0	330.24	0.0	**44.21**
	C22	0.0	2,945.74	0.0	344.59	0.0	**44.59**
	C23	0.0	3,814.22	0.0	389.57	0.0	**47.86**
	C24	0.0	4,335.29	0.0	426.58	0.0	**52.14**
Average		0.0	2,121.71	0.0	245.67	0.0	**32.63**
Semi-clusered	C13	0.0	993.58	0.0	108.66	0.0	**18.21**
	C14	0.0	1,024.29	0.0	119.67	0.0	**18.56**
	C15	0.0	1,143.29	0.0	149.58	0.0	**23.57**
	C16	0.0	1,147.85	0.0	150.14	0.0	**24.58**
	C17	0.0	1,448.29	0.0	189.55	0.0	**29.67**
	C18	0.0	1,459.84	0.0	195.51	0.0	**33.24**
	C19	0.0	2,152.62	0.0	264.21	0.0	**38.24**
	C20	0.0	2,199.27	0.0	269.58	0.0	**39.58**
	C21	0.0	3,024.23	0.0	345.28	0.0	**45.88**
	C22	0.0	3,059.84	0.0	368.54	0.0	**47.84**
	C23	0.0	4,124.22	0.0	410.24	0.0	**50.28**
	C24	0.0	4,329.87	0.0	438.52	0.0	**53.66**
Average		0.0	2,175.59	0.0	250.79	0.0	**35.27**
Total Average		0.0	2,145.61		247.10		**33.55**

Table 4.5 GOROBI, ALNS, and MBATA performances under large-size problem instances

Disaster- affected node distribution		GUROBI		ALNS		MBATA	
	Case	$\Delta f_1(\%)$	$T(s)$	$\Delta f_2(\%)$	$T(s)$	$\Delta f_3(\%)$	$T(s)$
Random	C25	0.0	6,422.21	0.0	584.27	0.00	**67.55**
	C26	4.27	LCT	1.90	596.48	1.90	**72.43**
	C27	2.89	LCT	0.67	680.09	0.67	**83.59**
	C28	4.67	LCT	1.70	734.56	1.70	**88.48**
	C29	7.56	LCT	1.88	811.88	1.88	**99.64**
	C30	7.24	LCT	1.73	941.06	1.73	**103.21**
	C31	8.67	LCT	1.43	979.28	1.43	**118.54**
	C32	9.91	LCT	1.38	1,013.07	1.38	**121.62**
	C33	18.34	LCT	1.59	1,159.61	1.59	**189.59**
	C34	20.27	LCT	1.91	1,188.57	1.91	**194.56**
	C35	21.32	LCT	0.61	1,243.28	0.61	**201.66**
	C36	22.46	LCT	0.99	1,387.44	0.99	**210.47**
	C37	30.24	LCT	0.65	1,421.08	0.65	**229.24**
	C38	31.93	LCT	0.94	1,447.13	0.94	**237.74**
	C39	33.31	LCT	1.48	1,564.08	1.48	**255.21**
	C40	33.38	LCT	1.35	1,588.03	1.35	**269.28**
Average		16.03	7,151.38	1.26	1083.74	1.26	**158.92**
Clustered	C25	0.0	6,387.19	0.00	579.19	0.00	**66.04**
	C26	0.0	6,245.28	0.00	584.22	0.00	**67.26**
	C27	4.01	LCT	0.79	677.41	0.79	**82.19**
	C28	5.89	LCT	1.82	689.18	1.82	**84.57**
	C29	5.79	LCT	1.70	812.08	1.70	**84.96**
	C30	6.22	LCT	0.79	819.56	0.79	**88.35**
	C31	7.53	LCT	2.03	975.24	2.03	**116.87**
	C32	8.63	LCT	1.10	986.24	1.10	**119.28**
	C33	21.06	LCT	1.75	1,138.28	1.75	**187.28**
	C34	21.73	LCT	1.88	1,180.27	1.88	**191.36**
	C35	22.48	LCT	0.59	1,237.28	0.59	**200.57**
	C36	23.26	LCT	0.82	1,234.08	0.82	**211.38**
	C37	30.41	LCT	0.98	1,424.01	0.98	**225.11**
	C38	32.60	LCT	0.83	1,439.25	0.83	**231.57**
	C39	32.51	LCT	0.96	1,501.16	0.96	**250.11**
	C40	32.60	LCT	0.88	1,576.04	0.88	**266.34**
Average		15.92	7,089.52	1.06	1,053.34	1.06	**154.57**
Semi-clustered	C25	2.71	LCT	1.08	595.28	1.08	**68.35**
	C26	4.62	LCT	2.22	599.33	2.22	**69.87**
	C27	5.95	LCT	1.87	658.31	1.87	**81.33**
	C28	5.69	LCT	0.52	674.24	0.52	**85.33**
	C29	6.39	LCT	1.95	819.26	1.95	**88.28**
	C30	6.74	LCT	1.68	826.34	1.68	**89.38**
	C31	6.59	LCT	1.17	992.21	1.17	**119.35**
	C32	7.47	LCT	1.86	1,012.08	1.86	**128.45**
	C33	18.90	LCT	1.30	1,089.24	1.30	**148.36**
	C34	19.27	LCT	0.57	1,114.27	0.57	**153.24**
	C35	23.98	LCT	0.77	1,268.54	0.77	**188.65**
	C36	25.09	LCT	1.11	1,293.53	1.11	**198.27**
	C37	33.35	LCT	1.33	1,435.24	1.33	**224.87**
	C38	34.85	LCT	1.19	1,487.21	1.19	**234.82**
	C39	25.75	LCT	0.87	1,524.26	0.87	**269.35**
	C40	25.52	LCT	0.56	1,548.25	0.56	**279.04**
Average		15.80	LCT	1.25	1,058.59	1.25	**151.68**
Total Average		15.91	LCT	1.19	1,065.22	1.19	**155.06**

LCT represents limited computational time set as 7,200 sec.

For large-size problems, we observe that GUROBI fails to solve most of the problem instances (solves only 3 out of 48 instances) by obeying the pre-specified termination criterion. On the other hand, experimental results indicate that both the ALNS and MBATA algorithms are capable of solving all the problem instances with an average of 1.19% optimality gap within the pre-specified time limit. On average, the ALNS and MBATA algorithms are found to be 6.70 and 46.09 times faster than GUROBI in solving the largeinstances of model [DR]. Moreover, it is observed that the MBATA algorithm is 6.86 times faster than the ALNS algorithm under the similar experimental conditions. In overall, MBATA algorithm consistently produces high quality solutions in a reasonable amount of time for the all the test instances considered in this study.

Sensitivity Analysis

The first set of experiments study the impact of drones depot location on utilized drones and consequently to the overall network cost. More specifically, we place the drone depot on different locations of the county such as county center, densely populated regions, northern, southern, eastern, and western part of the county and examine their impact on utilized drones and overall network/service cost. Figure 4.4 demonstrates the nine geographical locations considered for the case experiments: (1) county center, (2)-(5) four densely populated areas in the Hancock county, and (6)-(9) northern, southern, eastern, and western part of the county. To run the experiments, we ensure that the distribution of battery recharging stations (F_{Φ}) remain fixed in all the experiments.

Results in Figure 4.5 clearly supports that the number of utilized drones and consequently the overall network cost are significantly impacted by the drone depot

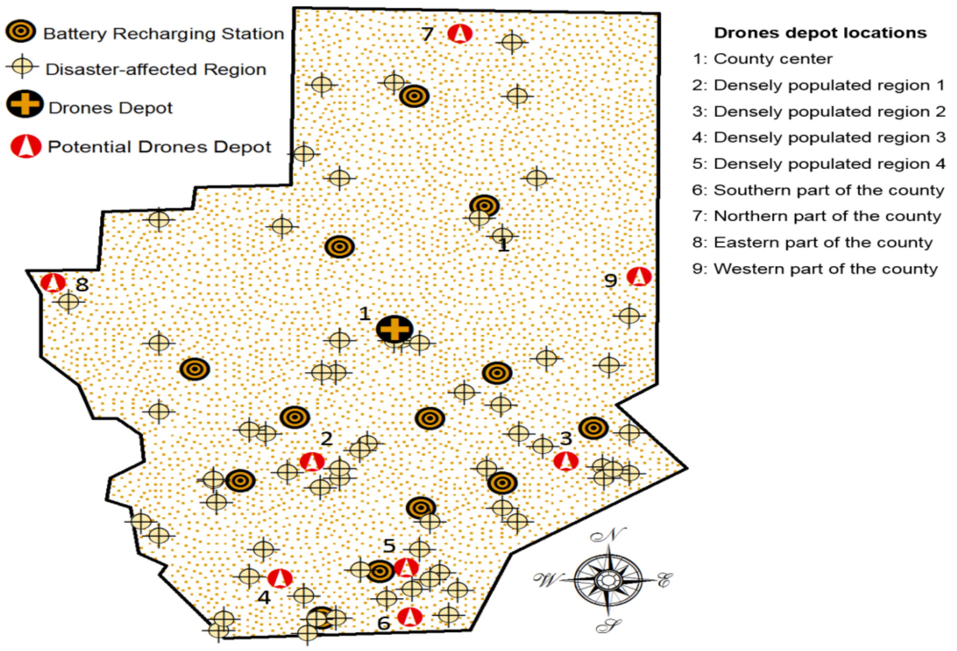

Battery Recharging Station
Disaster-affected Region
Drones Depot
Potential Drones Depot
Drones depot locations
1: County center
2: Densely populated region 1
3: Densely populated region 2
4: Densely populated region 3
5: Densely populated region 4
6: Southern part of the county
7: Northern part of the county
8: Eastern part of the county
9: Western part of the county
N
E
S
W

Figure 4.4 Illustration for potential locations of drones depot on the network

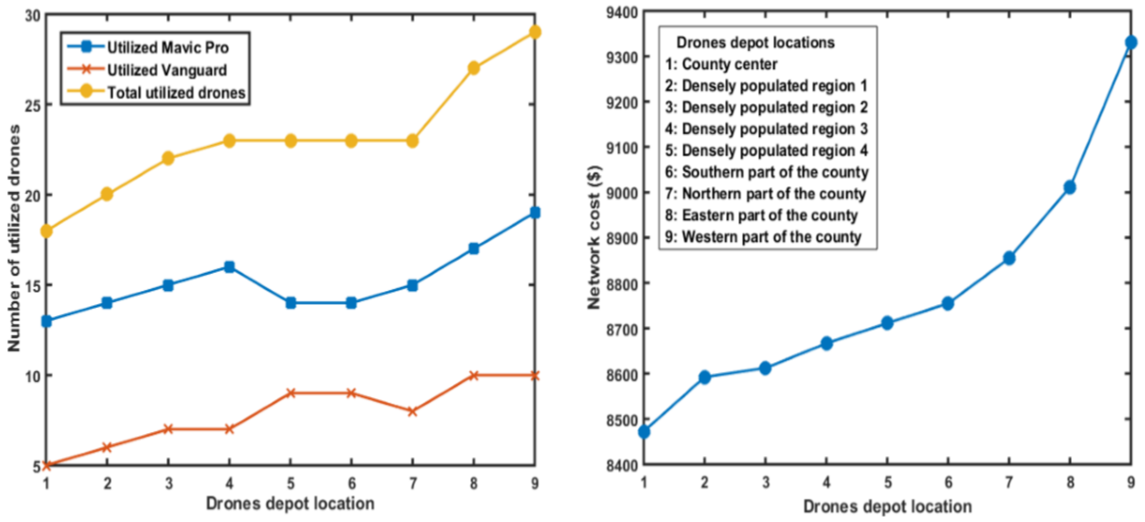


Figure 4.5 Impact of drones depot location on (a) utilized drones and (b) network cost

location. For our case study, we observe that if the depot is placed in the center of the county, then the design shall minimize the total number of utilized drones and results in a minimum network/service cost. However, if the drone depot center is placed at any densely populated region (2)-(5) or cardinal region (6)-(9), then both the number of drones required to serve a disaster-affected region and the overall network/service cost increases simultaneously. This is primarily due to the reason that the depot location significantly impacts drone's maximum allowable travel time t_k^{max}. It is worth noting that even though the number of utilized drones remained same when the depot location is placed in the third and fourth densely populated region as well as the northern and southern part of the county i.e., location (4)-(7), the overall network costs are found to be different for them. This may incur due to different combinations of utilized drone types and/or different developed routes for utilized drones.

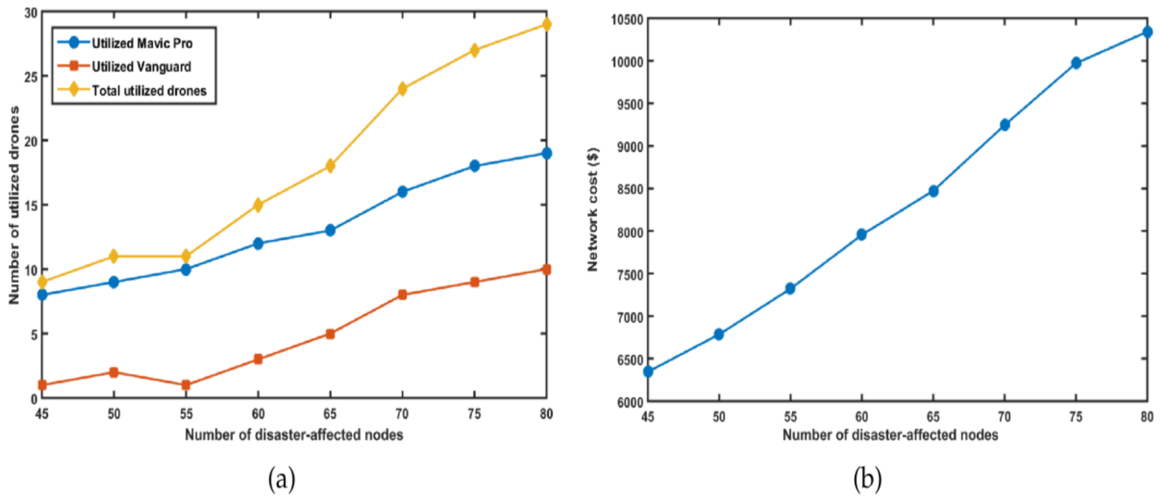


Figure 4.6 Impact of number of disaster-affected nodes on (a) utilized drones and (b) network cost

The second set of experiments study the impact of number of disaster-affected nodes, $|I|$, on the number of utilized drones to serve a disaster-affected region and overall network cost. To run these experiments, we vary $|I|$ from 45 to 80 and randomly place them in the tested region. However, in all the experiments, the number of battery recharging stations $\mathcal{F}_{\Phi}$ are remained fixed. Results in Figure 4.6 clearly indicate that there is a direct relationship between the number of disaster-affected nodes $|I|$ with the total number of utilized drones and the overall network cost. For instance, if the node size $|I|$ increases from 45 to 80, then the overall drone requirement to serve a disaster-affected region and network cost are increased by approximately 222.2% and 62.5%, respectively. This clearly indicates that $|I|$ has a significant impact on the overall system performance.

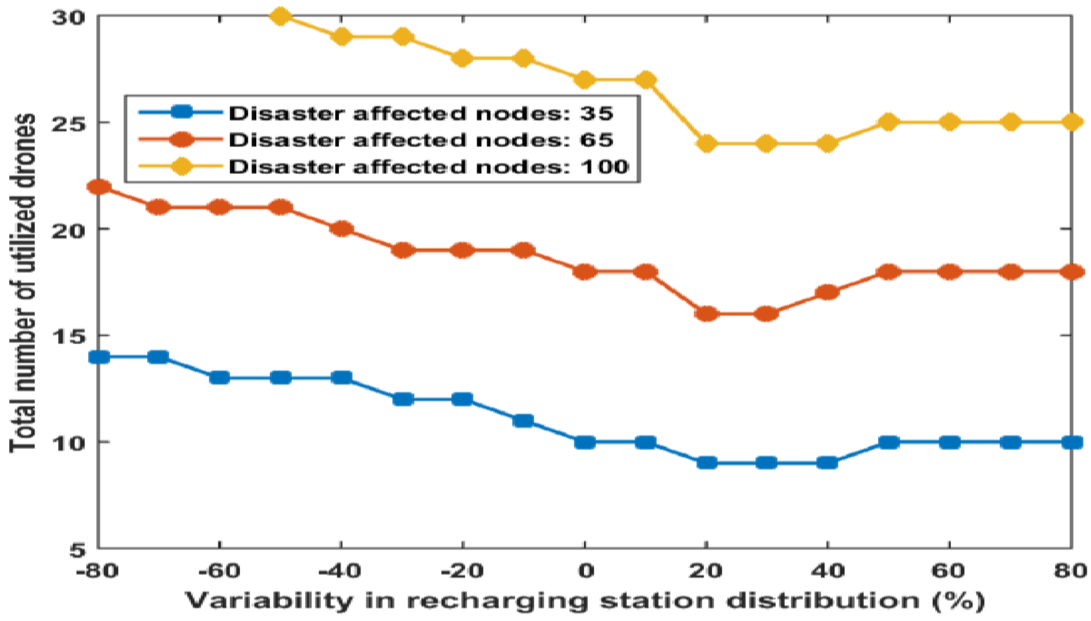


Figure 4.7 Impact of battery recharging stations distribution on drone utilization

The third set of experiments study the impact of battery recharging stations distribution $\mathcal{F}_{\Phi}$ on drone utilization. To perform these experiments, we vary the base $|\mathcal{F}_{\Phi}| = 13$ by ±80% and examine them on three different node sets $|I| = \{35,65,100\}$. Figure 4.7 presents the impact of variability on $|\mathcal{F}_{\Phi}|$ to the total number of utilized drones (Mavic Pro and Vanguard) in serving our tested region. Results indicate that the number of utilized drones decreases with an increase in $|\mathcal{F}_{\Phi}|$ for up to a certain level. Beyond that level, the requirement for drone to serve the same geographical area increases. This level can be considered as the minimum economic level beyond and after which the number of drones required to serve a disaster-affected region increases. Experimental results indicate that setting $|\mathcal{F}_{\Phi}|$ between 20-40, 20-30, and 20-40 provides the minimum economic level for $|I| = 35, 65,$ and $100,$ respectively.

Table 4.6 Utilized drones and network cost based on the drone's t_k^{max}

Mavic Pro t_k^{max} (mins)	Vanguard t_k^{max} (mins)	Utilized Mavic Pro	Utilized Vanguard	Total utilized drones	Network cost ($)
50	120	13	5	18	8,472
50	110	13	5	18	8,472
50	100	14	5	19	8,568
50	90	15	5	20	8,594
50	80	16	6	22	9,656
50	70	18	6	24	9,945
50	120	13	5	18	8,472
60	120	13	5	18	8,472
70	120	13	5	18	8,472
80	120	13	5	18	8,472
90	120	13	5	18	8,472
100	120	13	5	18	8,472
50	120	13	5	18	8,472
60	110	13	5	18	8,472
70	100	12	5	17	8,249
80	90	11	4	15	7,749
90	80	10	4	14	7,552
100	70	9	4	13	7,329

The next set of experiments study the impact of drone's maximum allowable travel time, t_k^{max}, on the number of utilized drones to serve a disaster-affected region and overall network cost. Selection of t_k^{max} typically depends on the discretion of the central disaster management planner. The value can be set to a low number when obtaining the information to a disaster-affected region is of utmost importance. However, depending upon the severity of the situation and urgency of collecting information, t_k^{max} can be varied but up to the drone's manufacturing range. To test the impact of t_k^{max} on system performance, we create three different scenarios where in the first scenario we vary the t_k^{max} for the Vanguard drones but kept it fixed for the Mavic Pro drones. The second scenario is created by varying the t_k^{max} for Mavic Pro drones while keeping the values fixed for the Vanguard drones. In the last scenario, we vary the t_k^{max} for both Mavic Pro and Vanguard drones.

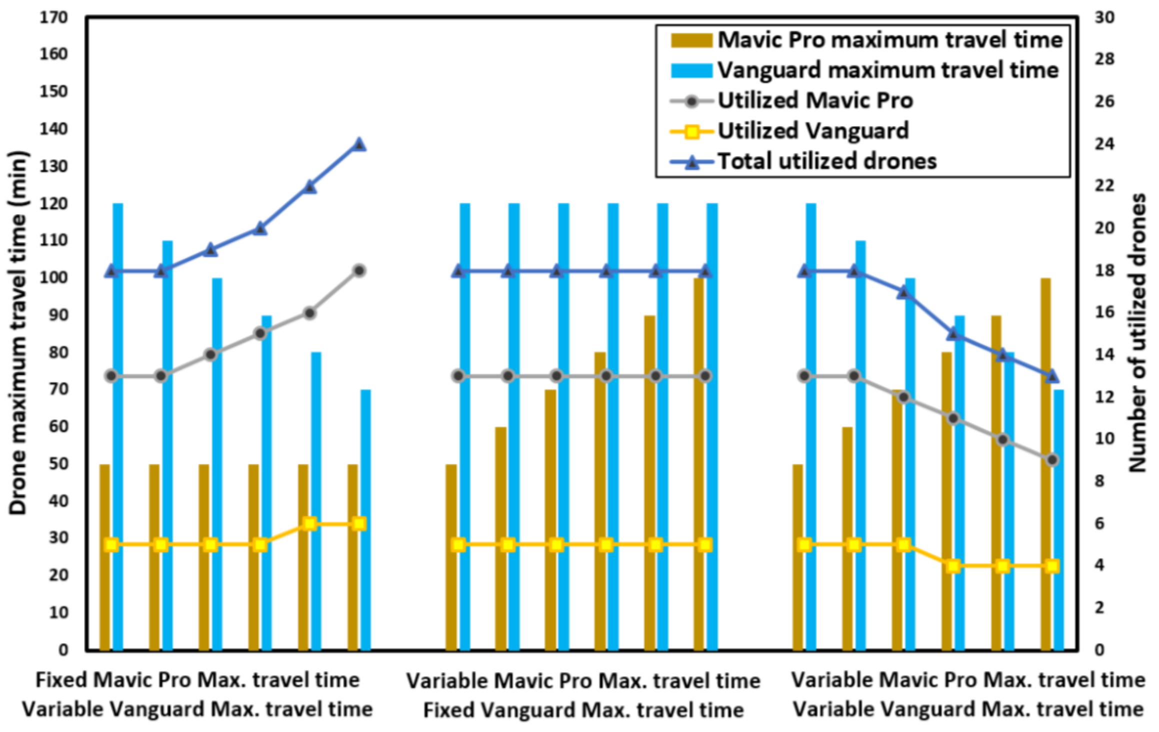


Figure 4.8 Impact of the t_k^{max} on utilized drones

Table 4.6 and Figure 4.8 show the impact of t_k^{max} on total utilized drones and network cost. Results clearly indicate that the number of utilized drones and consequently the overall network cost are not sensitive to the changes of t_k^{max} for the Mavic Pro drones. For instance, if the t_k^{max} changes from 50 minutes to 100 minutes, the total utilized drones to serve a disaster-affected region and the overall network cost still remain unchanged. However, we observe that the number of utilized drones and the overall network cost are sensitive to the changes of t_k^{max} from the Vanguard drones, but fixed for the Mavic Pro drones, or the case when t_k^{max} for both the Mavic Pro and Vanguard drones vary. For instance, the number of utilized drones and network cost increase by 33% and 17%, respectively, when Vanguard drones t_k^{max} are reduced by 42%. Similarly, we observe a 28% and 13% decrease in utilized drones, when t_k^{max} for the Mavic Pro and Vanguard drones are increased and decreased by 100% and 42%, respectively. In summary, more drones are needed to serve a disaster-affected region if t_k^{max} for the drones becomes low. Further, utilizing Mavic Pro drones with high t_k^{max} and Vanguard drones with low t_k^{max} can potentially reduce the overall network/service cost.

The last set of experiments study the impact of drone recharging time ψ_{jk} on drone utilization. The experimental results will reveal that how the improvement of ongoing drone battery technology may help to reduce the number of utilized drones to serve a disaster-affected region. To perform these experiments, we vary the base ψ_{jk} by ±80% and examine them on three different node sets $|I| = \{35, 65, 100\}$. Figure 4.9 presents the impact of ψ_{jk} on utilized drones (Mavic Pro and Vanguard). It is observed from the figure that on average, a 60% increase in ψ_{jk} will require an additional 12.2% drones to serve a disaster-affected region from the base case value. Similarly, an average of 14.69% less number of drones will now require to serve the same disaster-affected area when the ψ_{jk} value decreases by 60% from the base value. Note that in Figure 4.9, we did not report any solution of $\psi_{jk} \geq 60\%$ for $|I| = 100$. This is primarily due to the violation of maximum allowable travel time constraints (8) and are caused by the increment in recharging time.

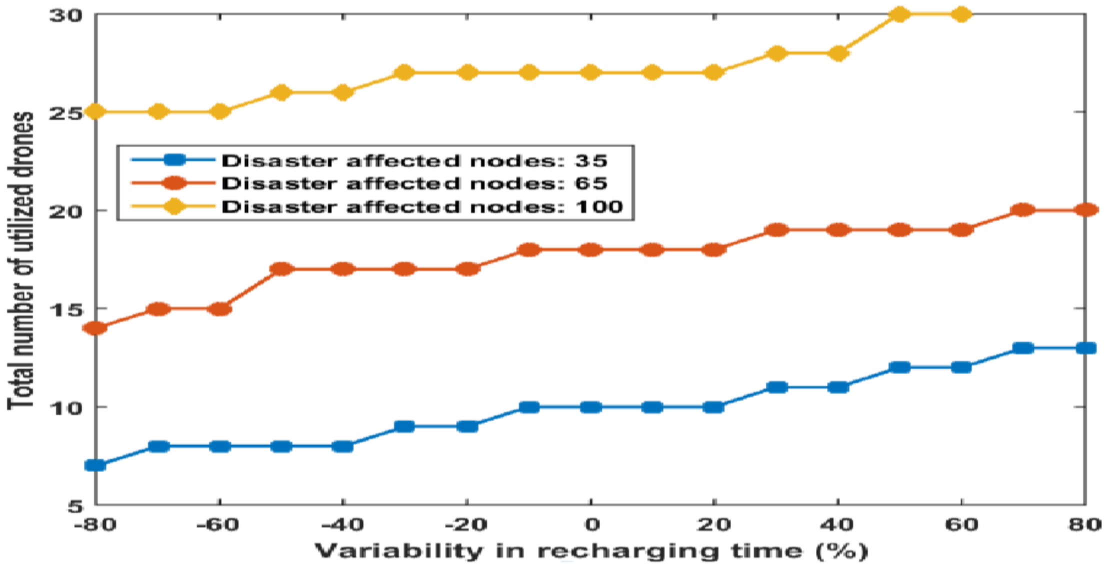


Figure 4.9 Impact of ψ_{jk} on drone utilization

CHAPTER V

CONCLUSIONS

This study investigates a Heterogeneous Fixed Fleet Drone Routing problem (HFFDRP) to design a safe, reliable, and cost-efficient disaster-affected region inspection plan using battery-driven drones. A mixed-integer linear programming model (MILP) is proposed to minimize the post-disaster inspection cost by considering a number of drone trajectory-specific factors into consideration such as ascending and descending costs, battery recharging costs, servicing costs (i.e., costs associated with taking images at disasteraffected nodes), drone hovering, turning, acceleration, constant, and deceleration costs. The trajectories between each pair of nodes are constructed using a path construction model that obeys the restrictions set legislated by the Federal Aviation Agency (FAA), technological performance of drones, along with geographical and environmental restrictions set for drone flights. To the best of the authors' knowledge, this is the first study that integrates drone trajectory-specific factors into routing decisions to serve/monitor a disaster-affected region. Additionally, due to the need to solve our proposed optimization framework in a realistic-size network problem, two solution algorithms, known as an Adaptive Large Neighborhood Search (ALNS) algorithm and Modified Backtracking Adaptive Threshold Accepting (MBATA) algorithm, are proposed. Computational results indicate that the proposed MBATA algorithm is capable of producing high-quality solutions consistently within a reasonable amount of time.

Finally, we use Hancock county from Mississippi State as a test bed to visualize and validate the modeling results. A number of managerial insights are drawn such as how the drone depot location, number of disaster affected nodes and battery recharging stations, maximum allowable travel time, and battery recharging time impact the design and management of a drone routing operation.

To summarize, the contributions of this study include: (i) introducing a new class of drone routing problem involving management of heterogeneous types of drones with limited batter capacity and charging station availability, speed optimization, and trajectory specific cost components; (ii) proposing a path construction model by obeying the restrictions set forward by FAA, technological, geographical, and environment restrictions of drone transportation; (iii) developing and testing of efficient heuristics, namely, ALNS and MBATA algorithms, to solve realistic-size network design problems; and (vi) managerial insights drawn from a real-life disaster-affected monitoring case study. We believe the proposed methodologies and managerial insights obtained from this study will help humanitarian organizations better manage the post-disaster recovery operations.

This study can be extended in several research directions. First, it would be interesting to see how the stochasticity associated with battery degradation and service priority impact our model framework. Further, the authors would like to examine how smart grid systems can be integrated with the proposed optimization framework to serve a disaster-affected region. These issues can be addressed in future studies.

REFERENCES

[1] Murphy R.R. Disaster Robotics. Cambridge MA: The MIT Press, Intelligent Robotics and Autonomous Agents series, 2014.

[2] Reuters. New Drone to Autonomously Inspect Crippled Fukushima Reactors. Available from: https://www.rt.com/news/ 266533-japan-drones-measure-radiation/, 2015.

[3] Cohen R. Humanitarian Aid Delivered by Drones: A New Frontier for NGOs? Available from: https://nonprofitquarterly.org/2014/07/16/ humanitarian-aid-delivered-by-drones-a-new-frontier-for-ngos/, 2014.

[4] Chowdhury S., Emelogu A., Marufuzzaman M., Nurre S.G., Bian L. Drones for disaster response and relief operations: A continuous approximation model. International Journal of Production Economics, 188:167–184, 2017.

[5] Gupta S.J., Ghonge M.M., Jawandhiya P.M. Review of unmanned aircraft system (UAS). International Journal of Advanced Research in Computer Engineering and Technology, 2(4):1646–1658, 2013.

[6] Gertler J. US Unmanned Aerial Systems. Available from: https://fas.org/sgp/ crs/natsec/R42136.pdf, 2012.

[7] Murray C.C., Chu A.G. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. Transportation Research Part C: Emerging Technologies, 54:86–109, 2015.

[8] Agatz N., Bouman P., Schmidt M. Optimization Approaches for the Traveling Salesman Problem with Drone. Available from: https://pdfs.semanticscholar.org/ 1112/cb5d327e4cd50b07e353298ca5c0aa094bb5.pdf, 2015.

[9] Coelho B.N., Coelho V.N., Coelho I.M., Ochi L.S., Zuidema D., Lima M.S., Da-Costa A.R. A multi-objective green UAV routing problem. Computers and Operations Research, 88:306–315, 2017.

[10] Ulmer M.W., Thomas B.W. Same-Day Delivery with a Heterogeneous Fleet of Drones and Vehicles. Available from: https://www.researchgate.net/profile/Marlin_ Ulmer/publication/315809092_Same-Day_Delivery_with_a_Heterogeneous_ Fleet_of_Drones_and_Vehicles/links/58e75628458515252 8de6060/ Same-Day-Delivery-with-a-Heterogeneous-Fleet-of-Drones-and-Vehicles. pdf, 2017.

[11] Dorling K., Heinrichs J., Messier G.G., Magierowski S. Vehicle routing problems for drone delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 47(1):70– 85, 2017.

[12] Kim S.J., Lim G.J. Drone-aided border surveillance with an electrification line battery charging system. Journal of Intelligent and Robotic Systems, pages 1–14, 2018.

[13] Lim G.J., Kim S.J., Cho J., Gong Y., Khodaei A. Multi-UAV pre-positioning and routing for power network damage assessment. IEEE Transactions on Smart Grid, doi: 10.1109/TSG.2016.2637408, 2016.

[14] Deng C., Wang S., Huang Z., Tan Z., Liu J. Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications. Journal of Communications, 9(9):687–692, 2014.

[15] d'Oleire Oltmanns S., Marzol I., Peter K.D., Ries J.B. Unmanned aerial vehicle (UAV) for monitoring soil erosion in morocco. Remote Sensing, 4(11):3390–3416, 2012.

[16] Cho J., Lim G., Biobaku T., Kim S.J., Parsaei H. Safety and security management with unmanned aerial vehicle (UAV) in oil and gas industry. Procedia Manufacturing, 3:1343–1349, 2015.

[17] Kim S.J., Lim G.J., Cho J. Drone flight scheduling under uncertainty on battery duration and air temperature. Computers and Industrial Engineering, In press, 2018.

[18] Montoya L. Geo-data acquisition through mobile GIS and digital video: an urban disaster management perspective. Environmental Modelling and Software, 18:869–876, 2003.

[19] Shang B., Wu C., Hu Y., Yang J. An Algorithm of Visual Reconnaissance Path Planning for UAVs in Complex Spaces. Journal of Computational Information Systems, 8(1):1–8, 2014.

[20] Quaritsch M., Kruggl K., Wischounig-Strucl D., Bhattacharya S., Shah M., Rinner B. Networked UAVs as aerial sensor network for disaster management applications. E and I Elektrotechnik und Informationstechnik, 127(3):56–63, 2010.

[21] Rabta B., Wankmller C., Reiner G. A drone fleet model for last-mile distribution in disaster relief operations. International Journal of Disaster Risk Reduction, In press, 2018.

[22] Erdogan S., Miller-Hooks E. A green vehicle routing probem. Transportation Research Part E, 48:100–114, 2012.

[23] Demir E., Bektas¸ T., Laporte G. An adaptive large neighborhood search heuristic for the pollution-routing problem. European Journal of Operational Research, 223:346–359, 2012.

[24] Koc C., Bektas¸ T., Jabali O., Laporte G. The fleet size and mix pollution-routing problem. Transportation Research Part B, 70:239–254, 2014.

[25] Markov I., Varone S., Bierlaire M. Integrating a heterogeneous fixed fleet and a flexible assignment of destination depots in the waste collection VRP with intermediate facilities. Transportation Research Part B, 84:256–273, 2016.

[26] Moshref-Javadi M., Lee S. The latency location-routing problem. European Journal of Operational Research, 255(2):604–619, 2016.

[27] Kwon Y. Choi Y., Lee D. Heterogeneous fixed fleet vehicle routing considering carbon emission. Transportation Research Part D, 23:81–89, 2013.

[28] Kizilates¸ G., Nuriyeva F. On the nearest neighbor algorithms for the traveling salesman problem. Advances in Computational Science, Engineering and Information Technology, 225:111–118, 2013.

[29] Ropke S., Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40:455–472, 2006.

[30] Pisinger D., Ropke S. Large neighborhood search: Handbook of Metaheuristics. Springer, International series on Operations Research and Management Science, 2010.

[31] Shaw P. A new local search algorithm providing high quality solutions to vehicle routing problems. Technical Report, Dept of Computer Science, University of Strathclyde, Glasgow, Scotland, UK, 1997.

[32] Ropke S., Pisinger D. A unified heuristic for a large class of vehicle routing problems with backhauls. European Journal of Operational Research, 171:750–775, 2006.

[33] Tarantilis C.D., Kiranoudis C.T., Vassiliadis S. A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. Journal of the Operational Research Society, 54:65–71, 2003.

[34] Tarantilis C.D., Kiranoudis C.T., Vassiliadis S. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research, 152:148–158, 2004.

[35] Subramanian A., Uchoa E., Ochi L.S. A hybrid algorithm for a class of vehicle routing problems. Computers and Operations Research, 40:2519–2531, 2013.

[36] Chiarandini M. Vehicle Routing Heuristic Methods. Available from: http://www.imada.sdu.dk/~marco/Teaching/Fall2008/DM87/Slides/dm87-lec19-2x2.pdf, 2017.

[37] Hansen P., Mladenovic N. Variable neighborhood search: Principles and applica- ´ tions. European Journal of Operational Research, 130:449–467, 2001.

[38] Gurobi. Gurobi Optimization. Available from: http://www.gurobi.com/, 2017.

[39] DJI. Mavic Pro. Available from: https://store.dji.com/product/mavic-pro, 2017.

[40] Airborne drones. The Vanguard. Available from: https://www.airbornedrones. co/, 2017.

[41] Solomon M.M. Algorithms for the vehicle routing and scheduling problems with time window constraints. Operations Research, 35(2):254–265, 1987.