TESTING CONNECTIVITY TO SAFE LOCATIONS AFTER A DISASTER USING DRONES

by

GÜLNIHAL ÖZCAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfilment of
the requirements for the degree of Master of Science

Sabancı University

December 2023

Gülnihal Özcan 2023 ©
All Rights Reserved

ABSTRACT

TESTING CONNECTIVITY TO SAFE LOCATIONS AFTER A DISASTER USING DRONES

GÜLNIHAL ÖZCAN
INDUSTRIAL ENGINEERING M.S. THESIS, DECEMBER 2023
Thesis Supervisor: Prof. Dr. Tonguç Ünlüyurt

Keywords: Disaster Management, Drone Technology, Post-Disaster Recovery, Connectivity Analysis, Safe Route Identification, Simulated Annealing, Sequential Testing

In this thesis, we develop a model in order to evaluate post-disaster infrastructure using drones. The study focuses on optimizing drone routes for assessing the connectivity of population centers to safe locations. The routes consist of elements that may fail including bridges, viaducts and road segments. We solve the resulting model using Simulated Annealing approach. With the right parameter settings, the method provides reasonable results in terms of speed and efficiency in completing the assessments. The research explores scenarios involving drones with varied battery constraints and assesses single-route efficiency. In order to assess the effectiveness of the Simulated Annealing approach, we compare the results with Local Search by considering practical constraints like battery life and recharging needs. This study introduces a relevant framework to disaster management and emergency response strategies, offering a scalable framework for rapid infrastructure evaluation in crisis situations.

ÖZET

AFET SONRASINDA GÜVENLİ KONUMLARA BAĞLANTININ DRON KULLANILARAK TEST EDİLMESİ

GÜLNIHAL ÖZCAN
ENDÜSTRİ MÜHENDİSLİĞİ YÜKSEK LİSANS TEZİ, ARALIK 2023
Tez Danışmanı: Prof. Dr. Tonguç Ünlüyurt

Anahtar Kelimeler: Afet Yönetimi, Drone Teknolojisi, Afet Sonrası İyileşme, Bağlantı Analizi, Güvenli Rota Tanımlama, Simüle Tavlama, Ardışık Test

Bu tezde, droneları kullanarak afet sonrası altyapıyı değerlendirmek için bir model geliştiriyoruz. Çalışma, nüfus merkezlerinin güvenli konumlara bağlanabilirliğini değerlendirmek için drone rotalarının optimize edilmesine odaklanmaktadır. Rotalar, köprüler, viyadükler ve yol segmentleri gibi arızalanabilecek unsurlardan oluşmaktadır. Ortaya çıkan modeli Benzetilmiş Tavlama yaklaşımı kullanarak çözüyoruz. Doğru parametre ayarları ile yöntem, değerlendirmelerin tamamlanmasında hız ve verimlilik açısından makul sonuçlar vermektedir. Araştırma, çeşitli batarya kısıtlamalarına sahip dronları içeren senaryoları inceliyor ve tek rota verimliliğini değerlendiriyor. Benzetimli Tavlama yaklaşımının etkinliğini değerlendirmek için, pil ömrü ve şarj ihtiyaçları gibi pratik kısıtlamaları göz önünde bulundurarak sonuçları Yerel Arama ile karşılaştırıyoruz. Bu çalışma, kriz durumlarında hızlı altyapı değerlendirmesi için ölçeklenebilir bir çerçeve sunarak afet yönetimi ve acil durum müdahale stratejilerine uygun bir çerçeve sunmaktadır.

ACKNOWLEDGEMENTS

I extend my deepest gratitude to my advisor, Prof. Tonguç Ünlüyurt, whose guidance, patience, and wisdom were invaluable throughout this challenging journey. His support and belief in me have been instrumental, and I am confident that they will continue to inspire my future endeavors.

I am also grateful to Assist. Prof. Tonguç Yavuz for being a member of my thesis jury. Special thanks go to Assoc. Prof. Kemal Kılıç, whose courses at Sabancı University have been exceptionally enriching for me and who honored me by participating in my jury as well. I would also like to thank all my other professors at Bilkent University and Sabancı University who always inspired me throughout this process and made me feel how lucky I was to study at such valuable universities.

My heartfelt thanks go to my dear brother, Alperen, and my precious friend, Gökçen, who stood closest to me during these times. They have always been the first hand to reach out when I was about to fall, even in the most difficult moments. Also, to my beloved parents, Ruhi Özcan and Bilgehan Özcan, whose unwavering support has been my stronghold. They have never withheld their support and have shown respect and faith in me and my decisions. I also hold dear the memory of Nazlı Özkalp, whose spirit and strength continue to be a guiding light in my life. I can never leave out my friends who have embraced me with love and warmth in Turkey and all over the world, especially Hatice, Irem, Nihan, Fatma, Gökhan and all my dears whose names I can't even begin to include.

"A person lives as long as they dream in this world." - *Yahya Kemal Beyatli*

May we always find the excitement to dream and the strength to strive for it.

This thesis is dedicated to the memory of those who lost their lives in the tragic earthquake on February 6, 2023, in my hometown Osmaniye and the surrounding region. I wish that such disasters are prevented in our beautiful beloved country Turkey and the world and that we are better prepared to minimize the impact of such disasters.

TABLE OF CONTENTS

ABSTRACT iv
ÖZET v
ACKNOWLEDGEMENTS vi
LIST OF FIGURES ix
LIST OF TABLES x
LIST OF ACRONYMS/ABBREVIATIONS xi
1. INTRODUCTION AND LITERATURE REVIEW 1
2. PROBLEM DEFINITION AND MATHEMATICAL MODEL 4
2.1. General Problem Definition and Objective 4
2.2. Computing Expected Time 6
2.2.1. Without Charging Capacity 6
2.2.2. With Charging Capacity 7
2.2.3. Approximate Expected Cost by Simulation 9
2.3. Example 11
3. SOLUTION METHODOLOGY 16
3.1. Brute Force Approach 16
3.2. Local Search 17
3.3. Simulated Annealing 24
4. EXPERIMENTS AND RESULTS 28
4.1. Data Generation 28
4.2. Parameter Selection 30
4.3. Results 32
5. CONCLUSION 35
REFERENCES 37
APPENDIX A: Algorithms and Results 40

LIST OF FIGURES

Figure 2.1. Example - all possible roads. 12

Figure 2.2. Example - single road G to check. 12

Figure 2.3. Example - sequences with charging station in each point. 13

Figure 4.1. Data set generation visualization 30

LIST OF TABLES

Table 4.1. Parameter selection for Simulated Annealing - shortened. 31

Table 4.2. Algorithm comparison without charging case - Prob Type 1. 32

Table 4.3. Algorithm comparison without charging case - Prob Type 2. 32

Table 4.4. Algorithm comparison without charging case - Prob Type 3. 33

Table 4.5. Algorithm comparison with charging case - Prob Type 1. 33

Table 4.6. Algorithm comparison with charging case - Prob Type 2. 34

Table 4.7. Algorithm comparison with charging case - Prob Type 3. 34

Table A.1. Parameter selection for Simulated Annealing - extended. 42

LIST OF ACRONYMS/ABBREVIATIONS

UAV	Unmanned Aerial Vehicles
SA	Simulated Annealing
LS	Local Search

1. INTRODUCTION AND LITERATURE REVIEW

Following natural disasters such as earthquakes, hurricanes, floods or fires, it may be necessary to evacuate many people to safe locations. For this reason, the structural integrity of bridges, viaducts and roads often becomes a critical concern. These vital transportation infrastructures play a crucial role in connecting communities and facilitating the flow of goods and services. However, the impact of these catastrophic events can cause significant damage and jeopardize the safety of bridge structures, viaducts and road structures in general. It is therefore crucial to quickly and efficiently assess the condition and functionality of these to ensure the safety of both rescue teams and the affected population. The rapid identification of safe routes for both human safety and the transportation of rescue and relief teams enables rapid post-disaster planning and coordination immediately after these events [1].

Traditionally, infrastructure assessment in post-disaster scenarios relies on manual inspections, often constrained by accessibility issues, security risks, and time constraints. The extensive nature of the damage and the urgency for information increase these challenges further, making traditional methods less feasible. Recent advances in drone technology offer a new solution to these challenges [2]. Drones equipped with advanced imaging and sensing technologies can quickly and safely access and assess damaged infrastructure and provide critical data in real time [3]. This capability is particularly useful for inspecting areas that are otherwise inaccessible or dangerous for human inspectors [4].

This thesis aims to develop a comprehensive framework for drone-based assessment of bridges, viaducts, and road integrity in general following earthquakes. The focus is on minimizing the expected time spent for determining whether a given route is operating or not by utilizing the efficiency of drones in collecting and analyzing structural data to facilitate rapid decision making for emergency response and subsequent restoration efforts. Chowdhury et al. introduces insights into the optimization of drone navigation paths for post-disaster inspection, including minimizing post-disaster

inspection costs and taking into account factors specific to the drone trajectory. The research proposes a mixed-integer linear programming model for 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 [5]. Additionally, Pimentel et al. provides insights into drone usage for agricultural optimization, with broader implications for route planning in various scenarios, including disaster management. The dissertation focuses on finding efficient routes for agricultural drones, considering factors like terrain elevation and obstacles [6].

In general terms, in this thesis we discussed the use of drones in post-disaster processes and combined this process with Sequential Testing literature and aimed to develop models that incorporate the use of drones with and without battery restrictions. In addition to the above-mentioned situations, it is aimed to make the fastest and safest path integrity decision in the post-disaster depending on the pre-disaster analysis to be made rather than determining a path.

The problem definition is an extension of the very well studied Sequential Testing problem. In general terms, the Sequential Testing problem tries to find out the correct state (functionality) of a system with the minimum expected cost. The system consists of different components and the state of the system depends on the state of the components through a defined function. There is an associated cost of learning the functionality of the individual components and the probability that a component functions is also known. Then the goal is to determine a strategy that describes the which component to test depending on the states of previously tested components while minimizing the expected cost. More information about the Sequential Testing problem can be found in [7].

In this thesis, the connectivity is defined as all the elements connecting the population center and the safe location being operational. So this corresponds to a simple series system where the system functions if and only if all the components function. This is essentially the simplest system that one can consider and it has been studied in the literature since 1960's (see e.g. [8]). On the other hand, the simple series system is

the building block in more complicated systems such as k-out-of-n systems [9], series parallel systems [10] and the results obtained for series systems can be adapted for these systems. In addition, others models have been proposed with different problem settings, incorporating different assumptions regarding feasibility, different cost functions and different types of testing policies. For instance, the possibility of performing some tests as a batch is considered in [11].

In the problem studied in this thesis, the cost correspond to the time it takes to find out whether a route is safe or not. There are two factors that makes this problem different that the Sequential Testing problem is that there are two types of costs (time spent) incurred. One is for the drone to go from one location to another. So this duration depends on the order that locations are visited. So this is different than the standard sequential testing problem where the testing costs are fixed apriori. This version of the problem has also been studied in the literature. In [12], the authors show that the problem is NP-complete and propose exact and heuristic methods to solve the problem. The second one is the duration that the drone spends at each location. This latter one is similar to the case in Sequential testing problem. The other is that the drones have limited power and this needs to be taken into consideration. Various frameworks can be considered for modeling this important factor. In particular, we consider two different scenarios for this. In one of them, we assume the drone has enough energy to complete the route. In the other one, we assume that the drone will spend sometime to recharge if it does not have enough energy to go to the next location in the solution.

2. PROBLEM DEFINITION AND MATHEMATICAL MODEL

In this chapter, the problem of testing the connectivity of two points on a path by using a drone will be described in detail. The whole section will be divided into two parts according to whether the drone has charging restrictions or not. We will provide illustrative examples and describe the computation of the expected cost.

2.1. General Problem Definition and Objective

In the aftermath of a disaster, it becomes essential to determine the feasibility of moving from one location to another. As mentioned in Chapter 1, our primary focus is on scenarios following seismic events, particularly the feasibility of getting from a starting point X representing population centers to a point Y representing a predetermined safe location. This research includes routes that are critical for logistical operations or routes that take populations to safe areas. The aim of this thesis is to utilize UAVs, commonly known as drones, to accelerate this assessment, thus enabling rapid assessment of specified routes or networks.

This study envisions two different scenarios: one where the drone operates without limitations on battery capacity, and another where battery capacity constraints require periodic recharging at the current location in case the power reserve is insufficient to proceed to the next assessment point. The route or network under study is assumed to consist of various way points, which may include elements such as roads, bridges or viaducts. These components are an integral part of the overall analysis as they potentially affect the drone's flight path and the feasibility of reaching the designated endpoints.

The goal is to find out if it is possible to safely get from X to Y in the shortest expected time. The solution is a permutation of the elements that make up the route

from the population center to the safe location. And the order of elements is the order that the drone will inspect these elements to figure out whether they are functional or not. If all the elements are functional the route is assumed to be safe, otherwise the current route is not safe and one has to test other routes to transfer people to safe locations.

Parameters:

- Let G be the set of points representing rode segments, bridges, viaducts etc on the route.
- Let s_i be the point i in the given sequence based on the route.
- Let p_i be the probability of point s_i being functional.
- Let t_i be spending time to check if point s_i is working or not.
- Let w_i be the time required by the drone to go from the starting point X to point s_i.

An important point for the problem that needs to be mentioned here is how the probabilities (p_i) and inspection times (t_i) can be considered as parameters. As it will be mentioned later in the Data Generation part (section 4.1), the reason why the probabilities are fixed is that especially elements such as bridges and viaducts have a lifetime that is determined from the moment they are built, and at the same time, depending on their age, the damage they have endured, a certain probabilistic value can be assigned to describe the post-disaster situation depending on these analyses. For this work, we assume that the status of the different elements on the route are independent from each other.

For the t_i value, where the drone tests the viaduct road integrity and bridge condition, it can make a comparison between before and after the disaster by using image processing based on the pre-analysis process. The time it can perform can be expressed in an estimated fixed value depending on the capacity of the drone, the area it is testing and other environmental conditions.

2.2. Computing Expected Time

The objective is to determine the optimal path for a drone to navigate from a starting point X to a safe point Y through a set of intermediate points G, where the drone has unlimited or limited battery life.

2.2.1. Without Charging Capacity

The parameters are stated in the section 2.1 above. The sequence S of length n is represented as $S = [X, s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)}, Y]$ where X is the starting point and Y is the last destination. For each sequence S, the arrangement of the points can change, but X and Y are fixed. $s_{\pi(i)}$ represents the i-th intermediate point in the journey as ordered by the permutation π.

The expected time E_s to traverse the path S is given by:

$$E_s = \sum_{i=1}^{n} \left(|w_{\pi(i)} - w_{\pi(i-1)}| + t_{\pi(i)} \right) \times \prod_{j=1}^{i-1} p_{\pi(j)} \tag{2.1}$$

Where;

- $|w_{\pi(i)} - w_{\pi(i-1)}|$ represents the distance (in terms of time) traveled from point $i-1$ to point i. For $i = 1$. this would be the time from the starting point X to $s_{\pi(1)}$.
- $t_{\pi(i)}$ is the time spent to check if i-th point in the sequence is operational.
- $\prod_{j=1}^{i-1} p_{\pi(j)}$ is the product term to represent the cumulative probability of all points until the i-th point being operational in the given order.

Given the route and the properties of each point, the optimization problem can

be represented as:

$$\begin{aligned} \text{Minimize} \quad & E_s = \sum_{i=1}^{n} \left(|w_{\pi(i)} - w_{\pi(i-1)}| + t_{\pi(i)} \right) \times \prod_{j=1}^{i-1} p_{\pi(j)} \\ \text{Subject to} \quad & s_{\pi(i)} \in N, \quad \forall i = 1, 2, \ldots, n \qquad \pi \text{ is a permutation of } 1, 2, \ldots, n \end{aligned}$$

This optimization problem seeks the sequence S that minimizes the expected time E_s for traversing the route.

2.2.2. With Charging Capacity

When considering a drone with limited charging capacity, we need to take into account the energy consumption for traveling between points and the time and cost associated with recharging the battery. This introduces an additional layer of complexity to our optimization problem.

Definitions:

- Let B be the total battery capacity of the drone.
- Let b_i represent the battery consumption to travel from point s_{i-1} to point s_i in terms of time and plus t_i the time spent to check the point s_i.
- Let C denote the charging cost for the drone, incorporating both the time and monetary cost of recharging.
- Let c_i indicate whether charging is required after reaching point s_i (1 if charging is needed, 0 otherwise).

Sequence Representation The sequence S of length n, including charging considerations, is represented as $S = [X, s_{\pi(1)}, s_{\pi(2)}, \ldots, s_{\pi(n)}, Y]$ with the same constraints

as before, but now including battery considerations.

Modified Expected Time Calculation The expected time E_s to traverse the path S, considering charging needs, is given by:

$$E_s = \sum_{i=1}^{n} \left(|w_{\pi(i)} - w_{\pi(i-1)}| + t_{\pi(i)} + c_i \times C \right) \times \prod_{j=1}^{i-1} p_{\pi(j)} \tag{2.2}$$

Where;

- c_i is determined based on the remaining battery after reaching point s_i. If the remaining battery $B - \sum_{k=1}^{i} b_k$ is less than or equal to b_{i+1}, then $c_i = 1$; otherwise, $c_i = 0$. And b_{i+1} calculation can be made as follows: $b_{i+1} = |w_{(i+1)} - w_{(i)}| + t_{i+1}$.
- C is added to the cost whenever charging is required (i.e., $c_i = 1$).

Optimization Problem with Charging Capacity Given the route, the properties of each point, and the drone's limited battery capacity, the optimization problem can be represented as:

$$\begin{aligned}
\text{Minimize} \quad & E_s = \sum_{i=1}^{n} \left(|w_{\pi(i)} - w_{\pi(i-1)}| + t_{\pi(i)} + c_i \times C \right) \times \prod_{j=1}^{i} p_{\pi(j)} \\
\text{Subject to} \quad & s_{\pi(i)} \in N, \quad \forall i = 1, 2, \ldots, n \qquad \pi \text{ is a permutation of } 1, 2, \ldots, n \\
& 0 \leq p_i \leq 1, \quad \forall i \\
& t_i, w_i, b_i \geq 0, \quad \forall i \\
& B \geq 0 \quad \text{(total battery capacity)}
\end{aligned}$$

This revised optimization problem seeks the sequence S that minimizes the expected time E_s to traverse the route, while also taking into account the limitations of the drone's battery life and the associated recharging costs.

2.2.3. Approximate Expected Cost by Simulation

In this thesis, we deal with a single line and represent it with points on the route and in this context we calculate each permutational expected cost. Here, the cost is defined as the time required to test the route with given sequence. This approach, primarily focused on permutational expected costs, is sufficient for the current analysis. However, it can be expanded to the case where multiple routes are considered simultaneously or in a different context. In these cases, it may not be possible to come up with a closed form expression that gives the total expected cost of a particular solution. In such a case, we can utilize a simulation approach to approximate the expected cost. In the rest of this subsection below, the methodology for approximating expected costs through simulation will be explained.

Expected Time Calculation for a Single Repetition

For a single repetition of the given sequence S, the expected time E_s is calculated as follows:

$$E_s = \sum_{i=1}^{n} \left(w_{s_i} - w_{s_{i-1}} + t_{s_i} \right) \tag{2.3}$$

Where:

- the relationship between a random number generated and the probability indicating whether it is working or not is checked. If the random number generated is greater than the probability rand() $> p_i$, it is assumed that the node is not

working and the test stops at that point, otherwise the drone moves to the next node.

- $w_{s_i} - w_{s_{i-1}}$ represents the distance (in terms of time) traveled from point s_{i-1} to point s_i.
- t_{s_i} is the time taken to check point s_i.

Total Expected Time

For the number of repetitions R, the total expected time E_T to traverse the given sequence S is:

$$E_T = \frac{1}{R} \sum_{r=1}^{R} E_{s,r} \tag{2.4}$$

Where $E_{s,r}$ is the expected time for repetition r of sequence S.

To account for the variability, method of calculating the expected time using multiple iterations is employed. The number of iterations, denoted as R, is given as input so that different calculations and scenarios can be taken into account. In addition, randomness is also incorporated into the calculation, thus enhancing the robustness.

In each iteration of the expected duration calculation, probability is integrated as a key factor. In the sequence, the probability of a given point functioning or not functioning is taken as a parameter, i.e. as data. A uniform random number between 0 and 1 is then generated for each point in the sequence and this random number is then compared with the operational probability of the corresponding point. If the random number is less than or equal to the point's operational probability, the test is considered successful, which means that the point is working. Conversely, if the random number exceeds the probability, the point is considered to be not operational and the test is terminated at that point.

By repeating this process R times, each iteration giving a potential scenario

of operational and non-operational points, it is possible to simulate a wide range of outcomes. The average of these results gives us the simulated expected duration. This methodology aligns with the principles of Monte Carlo simulation, a widely used technique for understanding the impact of risk and uncertainty in forecasting and modeling scenarios. Through this Monte Carlo simulation approach, we are able to effectively estimate the expected time, taking into account the probabilistic nature of point functionality and the inherent randomness of the system.

Optimization Problem

Given the route and the properties of each point, the optimization problem can be represented as:

$$\begin{aligned} \text{Minimize} \quad & E_T \\ \text{Subject to} \quad & s_i \in N, \forall i \end{aligned}$$

This optimization problem seeks the sequence S that minimizes the expected time E_T for traversing the route.

2.3. Example

Let's assume that there is a starting point X, last point which is represented as safe point Y and there are points in between. We need to verify if the points in between are functioning/working or not to be able to determine if it is safe to go from X to Y for a certain route. The decision should be done to give a sequence which minimizes the checking the situation of the route in terms of time.

To be more specific, we can actually support this with real-life examples. For example, let's say that there is an earthquake. This means that in the aftermath of

the earthquake, people have gathered at a certain first meeting point. However, in case of an earthquake, there are predetermined final safe points outside the initial meeting points, even if they differ for cities, countries and regions. The important issue here is which route a group of people can take from point X to point Y, which is the main safe point. This decision must be made as quickly as possible and the evacuation process must begin.

Different routing examples that can be used to go from one point to another point as seen in this Figure 4.1. For example, there can be 5 different ways to go from point X to point Y and two of them stated in the figure below.

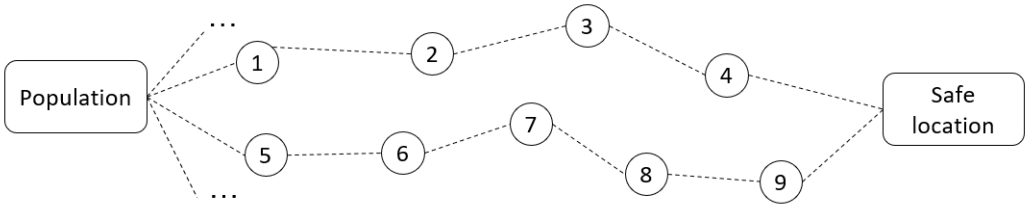

Figure 2.1. Example - all possible roads.

The main point discussed in this thesis is to take one of the existing routes, which is called route G, from X to Y and determine as quickly as possible whether it works or not. Therefore, the drone departing from point X should test whether this given route G, which contains 4 different points, is suitable or not. In this example, let's assume that all 4 points are bridges, as seen in Figure 4.1. The main question is in what sequential order the drone will test this system after taking off and how it can quickly decide whether route G works or not.

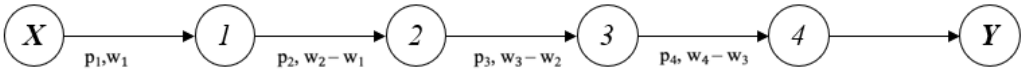

Figure 2.2. Example - single road G to check.

For this reason, the sequential testing approach mentioned in chapter 1 of the

literature will be used. We will discuss in what order this test should be conducted and the results given, considering the probabilities of these bridges being destroyed or not, in order to minimize the time spent.

As we discussed in the previous sections 2.1 and 2.2, this study focuses on two different cases. One is the case where the drone does not have a battery capacity restriction and the battery is not taken into account, and the other is the case where the battery is taken into account and the drone operates with a certain capacity of battery and can be recharged when necessary, which is reflected in the total cost. In this second case, it is assumed that there are charging points at each of the points to be tested and that the drone can charge itself if necessary based on the condition we defined.

Different routing examples that can be used to go from one point to another point as seen in this Figure 4.1. For example, there can be 5 different ways to go from point X to point Y and two of them stated in the figure below.

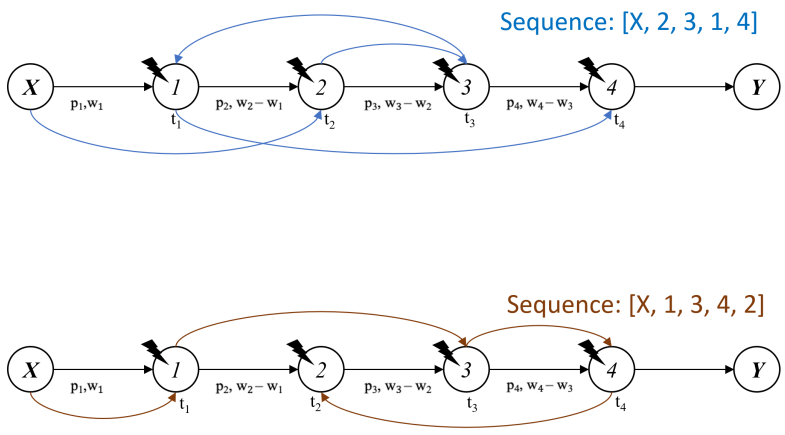

Figure 2.3. Example - sequences with charging station in each point.

In the aftermath of a disaster, such as an earthquake, it becomes crucial to assess the safety of routes for evacuation or relief efforts. We consider a simplified model for such a scenario, where a drone is tasked with verifying the safety of a route from a starting point X to a safe endpoint Y, with intermediate checkpoints. For our example,

we define the sequence of points as [X, 1, 2, 3, 4, Y].

Given this setup, there are 4! (24) possible permutations of the points (1, 2, 3, 4). These permutations represent different routes the drone could follow to assess the safety of the route.

Example of Expected Cost Calculation with Charging Capacity

Let's consider the sequence [X, 1, 3, 4, 2, Y] as an example. To calculate the expected cost for this sequence, with the consideration of the drone's charging capacity, we use the following parameters and formulation as explained in the previous sub sections:

- B - total battery capacity of the drone: $B = 30$
- b_i - battery consumption to travel from point s_{i-1} to point s_i and plus t_i to check the point if it is working or not time-wise.
- C - charging cost for the drone, incorporating both time and monetary cost: $C = 15$
- c_i - indicator whether charging is required after reaching point s_i (1 if yes, 0 otherwise).
- p_i - probability of point s_i being operational: $p = [0.3, 0.5, 0.4, 0.2]$
- t_i - time to check if point s_i is operational: $t = [5, 8, 3, 4]$
- w_i - time to travel from the starting point X to point s_i: $w = [12, 15, 17, 23]$

The expected time E_s for the sequence [X, 1, 3, 4, 2, Y] is calculated as (by using the formula 2.2):

$$E_{[1,3,4,2]} = \sum_{i=1}^{4} \left(|w_{\pi(i)} - w_{\pi(i-1)}| + t_{\pi(i)} + c_i \times C \right) \times \prod_{j=1}^{i} p_{\pi(j)} \tag{2.5}$$

The expected cost for this example can be calculated as follows:

$E_s = (12 + 5) + ((|12 - 17| + 3) + 15)(0.3) + (|17 - 23| + 4)(0.3 * 0.4) + (|23 - 15| + 8)(0.3 * 0.4 * 0.2) = 25.484$

Through this calculation, we can determine the most efficient sequence to follow for the drone, considering both the need to verify the safety of the route and the limitations of the drone's battery life.

Additional important aspect that needs to be mentioned as a note is that when considering all these calculations, we never actually think about going from the last point to point Y, it is ignored. Because at the last point we are actually done testing whether the system works in general or not. For this reason, it is important to assume that we go from the starting point to the first point and then travel to the other points in the given sequence but not to the final destination.

3. SOLUTION METHODOLOGY

In this section, we summarize the algorithms used to obtain the sequence in which the drone test can be performed in the shortest time. Two strategies were considered: Local Search and Simulated Annealing. Both strategies have been applied to scenarios without and with consideration of battery. In addition, the brute force approach is also mentioned.

3.1. Brute Force Approach

The Brute Force Approach involves generating all possible sequence permutations of the points on the route. For each permutation, the algorithm calculates the total time cost associated with testing the points in that sequence. This time cost includes the travel time between points and the time spent testing each point. After evaluating all permutations, the algorithm selects the sequence with the minimum total time cost.

The primary challenge of this approach is the computational complexity. The number of permutations increases exponentially with the number of points in the route $n!$ for n points, which makes this method impractical for larger routes due to the exponential growth in computation time and resource requirements [13].

Despite its limitations, the Brute Force Approach can be valuable in scenarios with a small number of points, where exhaustive enumeration is possible. It provides a benchmark for evaluating the effectiveness of heuristic methods such as simulated annealing and local search.

- **Initialization:** The algorithm starts by initializing variables to store the minimum cost found and the corresponding sequence so far.
- **Permutation Generation:** It then generates each possible permutation of the points in the route and add them into the list as sequences.
- **Cost Calculation:** For each permutation, the algorithm calculates the total

Algorithm 1 Brute Force Approach Algorithm

```
Initialize minCost to infinity
Initialize bestSequence to an empty sequence
Calculate all possible n! many sequences with points in G as permSequence list
for each permSequence do
    currentCost ← ExpectedCostCalculation(permSequence)
    if currentCost < minCost then
        minCost ← currentCost
        bestSequence ← permSequence
    end if
end for
```

time cost by adding up the times between consecutive points and the checking times at each point.

- **Comparison and Update:** If the total time cost for a permutation is less than the current minimum, the algorithm updates the minimum cost and the best sequence.
- **Result:** After evaluating all permutations with for loop, the algorithm returns the sequence with the lowest total time cost.

3.2. Local Search

Building on the foundational problem definition and mathematical model presented in the previous chapter 2, this section focuses on algorithmic strategies to determine the optimal set of test points for drone navigation in a post-disaster scenario. For each methodology, scenarios with and without battery constraints will be considered. In this section, well known heuristic Local search algorithm will be considered.

The Local Search Heuristic is a technique that starts with an initial sequence and iteratively improves it by making local modifications [14]. In general, the local search algorithm searches for an initial solution and then continuously seeks its neighborhood for a better solution. If such a neighbor is found, it substitutes the current solution.

The algorithm stops as soon as there is no better neighbor of the current solution [15]. This approach is particularly effective in scenarios where it is computationally infeasible to find an exact solution due to the complexity of the problem [16]. The algorithm uses a swapping scheme to discover neighboring sequences of the current sequence and then evaluates each one to find the sequence that minimizes the expected duration of the drone's route.

- **Initialization**: Starts with a specific initial sequence. In our case, a randomly created initial sequence is given in order to increase the performance of the algorithm.
- **Evaluation**: At per step, calculates the expected time for the current and neighboring sequences.
- **Iteration**: Recursively swaps elements in the sequence to explore neighboring configurations.
- **Optimization**: Seeks to continuously optimize the sequence according to the calculated expected time.
- **Termination**: Terminates when no further optimization is found by considering also the given stopping condition.

In this section, we present different pseudo-codes for the with and without charging cases. The expected time calculation algorithm, the local search algorithm and the swap sequence for the local search algorithm are shown in 3 different pseudo-codes.

Algorithm 2 Calculate Expected Time for a Network Visit Without Charging

Require: A network represented as a route, a sequence of nodes to visit.

Ensure: The expected time to visit the given sequence of nodes in the route.

```
function CalculateExpectedTime(network, sequence)
    total_time_sequence ← 0
    current_node ← 'X'                                  ▷ Start from node 'X'
    probAll ← 1
    for each node in sequence do
        p ← properties['p']                             ▷ Probability
        t ← properties['t']                             ▷ Time to check if the point is working or not
        w ← properties['w']                             ▷ Distance to node X
        distance ← |distanceNow − distancePrevious|     ▷ Calculate the distance between the current node and previous node
        probPrevious ← network.nodes[previous_node]['p']
        probAll ← probAll × probPrevious
        total_time_sequence ← total_time_sequence + ((distance + t) × probAll)
        previous_node ← node
    end for
    return total_time_sequence
end function
```

Algorithm 3 Local Search Without Charging

Require: A network represented as a route and an initial sequence of nodes.

Ensure: The best sequence of nodes with the lowest expected cost.

```
function LocalSearch(network, initialSeq)
    currentSeq ← initialSeq
    currentCost ← CalculateExpectedTime(network, currentSeq)
    swappedBestSeq, swappedBestCost ← SwapSequence(network, currentSeq)
    while currentCost ≥ swappedBestCost and swappedBestSeq ≠ initialSeq
  and swappedBestSeq ≠ currentSeq do
        currentCost ← swappedBestCost
        currentSeq ← swappedBestSeq
        swappedBestSeq, swappedBestCost ← SwapSequence(network, swappedBestSeq)
    end while
    return swappedBestSeq, swappedBestCost
end function
```

Algorithm 1: Calculate Expected Time for a Network Visit.

- **Initialization:** It sets the total duration and the start node 'X'. This initialization is essential to establish a reference point in the network.
- **Node Properties:** It derives the key features (probability, inspection time, distance) that are critical in calculating the expected time for each node.
- **Distance Calculation:** Calculates the distance between consecutive nodes, an important factor in time estimation.
- **Probability Accumulation:** It updates the cumulative probability of reaching each node, an integral part of the probabilistic nature of the model.
- **Time Accumulation:** Taking into account distance, inspection time and cumulative probabilities, it sums the total time and provides the expected time for the sequence.

Algorithm 2: Local Search.

- **Initialization and Cost Calculation:** Starts with an initial sequence and calculates its cost, determines the baseline for improvement.
- **Best Sequence Identification:** Determines the best sequence after the swap operation, which is critical for iterative improvement.
- **Iterative Improvement:** It continuously updates the sequence and cost by searching for a local optimum. Loop termination criteria prevent the algorithm from stagnating.

Algorithm 3: Swap Sequence for Local Search (Algorithm 7 in Appendix A).

- **Initialization:** Sets the initial best cost and sequence, preparing for the swap operations.
- **Iterative Swapping:** Examines pairs of nodes for potential swaps, a key mechanism for exploring neighboring configurations.
- **Cost Evaluation and Update:** Evaluates and records the cost of each swapped sequence, updating the best found sequence based on the lowest cost.

Local Search with Charging

Local Search is extended to include charging for drone navigation in a post-disaster scenario. The algorithms are modified to account for the battery life of the drone, making the approach more realistic for real-world applications. Charging considerations are embedded into the expected time calculation and the local search process. In this part, as described in section 2, the possibility that your drone can recharge itself when it needs to be charged is assumed, thanks to the wireless charge stations on the points.

The following algorithm calculates the expected time for a drone to traverse a given sequence of nodes, considering the drone's need to recharge its battery. The only change in the local search and swapping algorithms will be the use of the CalculateExpectedTimeWithCharging function, so that when calculating the cost during the local

search, the charge state is taken into account. As described in the CalculateExpectedTimeWithCharging algorithm below, the drone's state of charge is checked and if it does not have enough battery to reach the next node and test it, then it recharges itself without traveling beyond the current node, so the cost of recharging is added to the cost in units of time and at the same time the total battery is updated to be recharged repeatedly.

In the annealing simulation section, section 3.3, Calculate Expected Time with and without charging algorithms will be used and be directly referenced.

Algorithm 4 Calculate Expected Time with Charging

Require: A network represented as a route, a sequence of nodes to visit, and initial charge C.

Ensure: The expected time to visit the given sequence of nodes in the route, accounting for charging time.

```
function CalculateExpectedTimeWithCharging(network, sequence, C))
    ReChargingTime ← RC
    total_time_sequence ← 0
    currentNode ← 'X'                                   ▷ Start from node 'X'.
    probAll ← 1
    currentCharge ← C
    for index, node in enumerate(sequence) do
        properties ← network.nodes[node]
        p ← properties['p'] , t ← properties['t'] and w ← properties['w']
        distance ← |distanceNow − distancePrevious|
        if index + 1 < len(sequence) then
            nextNode ← sequence[index + 1]              ▷ Verify the next node to visit.
            nextDistance ← |network.nodes[nextNode]['w'] − distanceNow|
            nextT ← network.nodes[nextNode]['t']
            currentCost ← distance + t
            if currentCharge < (distance + t + nextDistance + nextT) then   ▷ Check if the charge is enough to travel to the next node and testing it.
                currentCost ← currentCost + RC and currentCharge ← C
            else
                currentCharge ← currentCharge − currentCost
            end if
        end if
        total_time_sequence ← total_time_sequence + (currentCost) × probAll   ▷ Update probAll and previousNode.
    end for
    return total_time_sequence
end function
```

3.3. Simulated Annealing

In parallel to the Local Search approach presented in the previous section, this section focuses on the application of Simulated Annealing to optimize drone routing in post-disaster scenarios. Simulated Annealing is an heuristic inspired by the process of physical annealing in metallurgy. Its ability to systematically avoid sub-optimal solutions and discover the most effective overall solutions makes it particularly suitable for our specific problem context. The adaptability of this method and its effectiveness in finding optimal solutions in challenging cases is key to its application in operational research. [17].

Similar to the Local Search algorithm, Simulated Annealing will be considered in scenarios with and without charging constraints. The algorithm uses the Compute Expected Time algorithm to calculate the cost of drone routes (see Algorithms 2 and 4).

Overview of the Simulated Annealing Process:

(i) **Initialization**: It starts with an initial sequence and a high 'temperature' that makes it easy to explore.

(ii) **Exploration and Acceptance**: The algorithm searches for neighboring solutions at each step (Annealing Swap Operation - Algorithm 8 in Appendix A). Lower cost solutions are always accepted, while higher cost solutions can be accepted based on a probability that decreases with temperature.

(iii) **Cooling Schedule**: By gradually reducing the temperature according to a cooling scheme, the probability of accepting higher cost solutions is reduced and convergence towards the optimum solution is achieved.

(iv) **Termination**: The process continues until the system 'cools down' to a predefined lower temperature limit, at which point it returns to the best solution found.

The following pseudo-code describes in detail the implementation of the Simulated

Annealing algorithm for routing drone, both without and with charging in consideration.

Algorithm 5 Simulated Annealing Without Charging

Require: A network represented as a route, an initial sequence of nodes, initial temperature $initialTemp$, cooling rate $coolingRate$, number of iterations per temperature $tempIteration$, and stopping temperature $stopTemp$.

Ensure: The best sequence of nodes with the lowest expected cost.

```
function SimulatedAnnealing(network, initialSeq, initialTemp, coolingRate, tempIteration, stopTemp)
    currentSeq ← initialSeq
    currentCost ← CalculateExpectedTime(network, currentSeq)
    bestSeq ← currentSeq
    bestCost ← currentCost
    temperature ← initialTemp
    while temperature > stopTemp do
        for i ← 0 to tempIteration − 1 do
            swappedSeq, swappedCost ← AnnealingSwap(network, currentSeq)
            if swappedCost < currentCost then
                currentSeq ← swappedSeq
                currentCost ← swappedCost
                if swappedCost < bestCost then
                    bestSeq ← swappedSeq
                    bestCost ← swappedCost
                end if
            else
                deltaCost ← swappedCost − currentCost
                if Exp(−deltaCost/temperature) > Random(0, 1) then
                    currentSeq ← swappedSeq
                    currentCost ← swappedCost
                end if
            end if
        end for
        temperature ← temperature × coolingRate
    end while
    return bestSeq, bestCost
end function
```

Simulated Annealing with Charging

The adaptation of Simulated Annealing for charged scenarios involves integrating charging restrictions into the cost assessment process. The algorithm follows a similar procedure as above but uses the Calculate Expected Time With Charging algorithm (Algorithm 4) for cost evaluation. By applying these methodologies, the study aims to find the most efficient pathways for drones under different operational constraints and thus improve the effectiveness of post-disaster management efforts.

When the charge restriction is introduced in simulated annealing, both in simulated annealing and in the annealing swap algorithms, the calculate expected time with charging algorithm, where the charge was considered in the previous section 3.2, comes into play, which only causes the cost evolution to change. In both cases, with or without charging, the main theme for simulated annealing remains the same temperature and cooling concept.

4. EXPERIMENTS AND RESULTS

In this chapter, it will be firstly explained the data generation process, which assumptions are made in relation to the subject matter. It will then present how the choice of the parameter for the SA algorithm is made and the results of the fixing of the parameters will be presented. Finally, the results and comparisons of the algorithms described in chapter 3 will be presented based on the generated data.

4.1. Data Generation

During data generation, first the data size is determined and then the parameters to be used in the algorithms are randomly generated according to certain data sizes. As explained in the Problem Definition part of the thesis (Chapter 2), the parameters that will be given to the model as constants are: the probability, the distance of the point from the starting point, and the time it takes the drone to test the point to see if it works. The distance and test time are randomly generated and the unit of time is minutes for both.

Where charging comes into play, the total charging capacity of the drone and the re-charge time at the point where it will re-charge itself if necessary is also created in advance as a constant. Again, these values are also measured in time (min). The charge capacity is always assigned based on the data set created so that the drone always has at least enough time to go to the farthest point from the starting point and test it: $max(|x\ to\ each\ point| + testing)$.

At this point, in order to increase the contribution of the problem, data sizes have been selected where the performance of the algorithms will make significant improvements. This selection was based on the Brute Force Approach described in section 3.1. As we have already mentioned, since the Brute Force Approach is mainly an enumerative approach, and since it generates sequences to be tried as many times as the permutation of the size, a threshold has been set at which the Brute Force Approach

cannot make a computational contribution.

- Brute Force Approach data size = 10, running time 12592.05ms and optimal value is 57.67 where simulated annealing algorithim can find the optimal value in half time.
- Brute Force Approach data size = 20, time limit 10min and no result.

For this reason, the data size is set to *10, 20, 30, 50* in this framework.

Another important aspect here is the assumptions that go into the creation of the probabilities. The basis for creating this is as follows: first of all, as mentioned earlier (in Chapter 2), these probabilities are a value assigned based on the stability of the point where the drone is going to test, be it a bridge, a certain point on the road or a viaduct. At this point, if this is taken, for example, especially on the bridge case, there is a life time assigned from the moment the bridge was built, as well as the damage caused by natural disasters, accidents, etc. [18] [19] Considering all these, the probability assignment for these points has been done in 3 different ways:

- Prob type 1 - Uniform probability $(0, 1)$,
- Prob type 2 - Low range probability $(0, 0.5)$ assuming routes with high attrition,
- Prob type 3 - High range probability $[0.5, 1)$ assuming routes with low attrition.

As you can see in the Figure 4.1 below, data generation is categorized based on instances and probabilities. 10 different data are generated for each instance and probability type intersection. In total 120 different data are generated as shown in the figure.

It should be noted that this is done as 120 data for the without charging case and 120 data for the with charging case. The data sets for testing the with and without charging cases are created independently of each other. In total 240 data are studied.

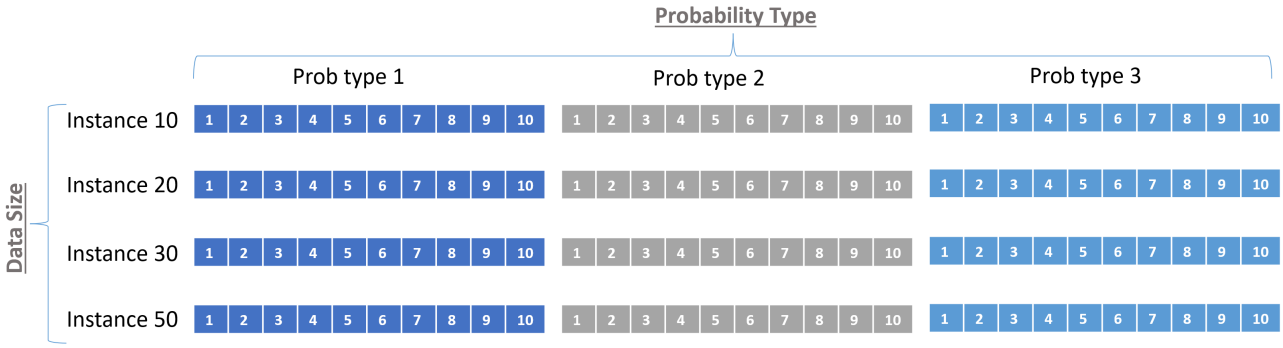

Figure 4.1. Data set generation visualization

4.2. Parameter Selection

This section describes in general terms how parameter selection is done. As explained in section 3.2, the Local Search algorithm has no input parameters, so the focus in this section is on simulated annealing. In general, SA is a method that becomes efficient when certain criteria and parameters are taken into account. For this reason, the parameter selection is of great influence and importance for the performance of the algorithm.

First of all, the critical parameters for the SA algorithm are: **initial temperature, cooling rate, temperature iteration and stop temperature**. In selecting the parameters to be fixed, certain values from the literature that are recommended to be used for SA were considered and various combinations of these parameters were created and the parameters that gave the best solution in the best time were fixed according to the results of the SA algorithm run with these parameters on the same data set [20] [21].

As can shown in the table, the values considered are as follows:

- Initial Temperature (InitialTemp): 800, 900, 1000,
- Cooling Rate (CoolRate): 0.8, 0.9, 0.95, 0.99,
- Temperature Iteration (TempIter): 750, 1000, 1500.

After the results and time comparisons, the values selected and fixed for the SA algorithm to be used in the next section are indicated in the table. In order to make this comparison, we have chosen data size as 10 where the Brute Force Approach gives the optimal value. Therefore, the best result and time to make a choice is controlled. Selection is showed in the following table (extended version of the table is Table $A.1$ in Appendix):

Inst.	**Alg.**	**Time (ms)**	**Result**	**InitialTemp**	**CoolRate**	**TempIter**	**StopTemp**
10	SA	890.52	57.99	800	0.95	750	10
10	SA	4345.19	58.01	800	0.99	750	10
10	SA	5747.78	58.08	800	0.99	1000	10
10	SA	8933.19	57.98	800	0.99	1500	10
10	SA	444.61	58.35	900	0.8	1500	10
10	SA	875.765	58.08	900	0.9	1500	10
10	SA	1206.49	58.82	900	0.95	1000	10
10	SA	4422.04	57.98	900	0.99	750	10
10	SA	5800.96	57.69	900	0.99	1000	10
10	SA	8977.97	58.06	900	0.99	1500	10
10	SA	306.33	58.33	1000	0.8	1000	10
10	SA	434.91	58.08	1000	0.8	1500	10
10	SA	607.93	58.15	1000	0.9	1000	10
10	SA	894.55	58.32	1000	0.9	1500	10
10	SA	1779.64	58.08	1000	0.95	1500	10
10	SA	4528.01	58.33	1000	0.99	750	10
10	SA	**5995.72**	**57.67**	**1000**	**0.99**	**1000**	**10**
10	SA	10664.06	57.67	1000	0.99	1500	10

Table 4.1. Parameter selection for Simulated Annealing - shortened.

4.3. Results

This section presents a summary of the solutions obtained by using the solution methodologies described in Chapter 3 to solve the problem described in Chapter 2 of this thesis. These results are obtained by using the generated data and the parameters chosen for SA as described in the previous sections (4.1 and 4.2). The main purpose of this section is to reflect these results for specific data sets through tables and to allow comparisons between the algorithms based on the results obtained.

As explained in section 4.1, data generation is based on data size and probability types. In this results section, we will run the algorithms with different data sizes generated in each probability type by clustering on the probability type. Then, the reflection of this in the table will be presented as the average of the results of 10 different data for each instance.

As a consequence of all these descriptions, there are 6 different tables in total in this section. 3 for the with-charge case and 3 for the without-charge case and in each case different probability type datasets are considered.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	3.31	58.07	-	-	-	-
10	SA	8182.60	58.01	1000	0.99	1000	10
20	LS	77.11	123.43	-	-	-	-
20	SA	13917.89	115.15	1000	0.99	1000	10
30	LS	352.13	76.06	-	-	-	-
30	SA	18883.01	71.56	1000	0.99	1000	10
50	LS	3294.47	102.52	-	-	-	-
50	SA	28938.53	83.11	1000	0.99	1000	10

Table 4.2. Algorithm comparison without charging case - Prob Type 1.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	8.54	20.34	-	-	-	-
10	SA	6458.01	20.34	1000	0.99	1000	10
20	LS	76.73	16.30	-	-	-	-
20	SA	10949.81	16.10	1000	0.99	1000	10
30	LS	282.95	12.78	-	-	-	-
30	SA	16214.04	12.77	1000	0.99	1000	10
50	LS	1342.86	12.17	-	-	-	-
50	SA	22041.46	12.12	1000	0.99	1000	10

Table 4.3. Algorithm comparison without charging case - Prob Type 2.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	5.10	56.90	-	-	-	-
10	SA	6368.67	50.11	1000	0.99	1000	10
20	LS	82.26	32.27	-	-	-	-
20	SA	11752.13	34.24	1000	0.99	1000	10
30	LS	444.07	32.81	-	-	-	-
30	SA	17114.33	34.01	1000	0.99	1000	10
50	LS	2171.35	23.56	-	-	-	-
50	SA	25464.91	31.03	1000	0.99	1000	10

Table 4.4. Algorithm comparison without charging case - Prob Type 3.

In the comparative analysis between LS and SA for with and without charging cases, while LS often yields smaller results indicating a small advantage, the subtle effectiveness of SA especially in complex problem scenarios cannot be ignored. Particularly in Problem Type 1, SA demonstrates its strong ability to explore a wider solution space leading to marginally better results, as seen in the 20 dimensions example where SA scores 115.15 versus 123.43 for LS (without charging) and SA scores 25.67 versus 29.61 for LS. This subtle advantage is indicative of SA's sophisticated exploration mechanism, which is particularly useful in uniformly distributed-probability scenarios where LS's simplistic approach may miss more subtle solutions. While LS shows a numerical superiority in Problem Types 2 and 3, SA's approach of methodically searching over a wider range of probabilities demonstrates its potential to uncover more optimized solutions in scenarios with complex constraints. Thus, while LS may be preferable in simpler situations due to its speed and directness, SA's strategic depth makes it a valuable tool for more complex problems where exploring a variety of potential solutions is crucial.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	5.86	27.14	-	-	-	-
10	SA	6910.52	21.47	1000	0.99	1000	10
20	LS	85.62	29.61	-	-	-	-
20	SA	9447.68	25.67	1000	0.99	1000	10
30	LS	586.14	23.72	-	-	-	-
30	SA	15664.02	23.69	1000	0.99	1000	10
50	LS	2586.54	21.21	-	-	-	-
50	SA	20694.06	21.08	1000	0.99	1000	10

Table 4.5. Algorithm comparison with charging case - Prob Type 1.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	6.90	24.52	-	-	-	-
10	SA	5890.52	24.52	800	0.95	750	10
20	LS	86.49	15.36	-	-	-	-
20	SA	10747.71	14.73	800	0.99	1000	10
30	LS	566.23	11.13	-	-	-	-
30	SA	14654.19	11.13	1000	0.99	1500	10
50	LS	2327.46	12.01	-	-	-	-
50	SA	21564.45	12.03	1000	0.99	1500	10

Table 4.6. Algorithm comparison with charging case - Prob Type 2.

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	LS	6.45	68.01	-	-	-	-
10	SA	5901.45	68.01	1000	0.99	1000	10
20	LS	91.51	30.88	-	-	-	-
20	SA	12741.23	33.07	1000	0.99	1000	10
30	LS	489.49	25.54	-	-	-	-
30	SA	15144.33	30.24	1000	0.99	1000	10
50	LS	2311.36	35.81	-	-	-	-
50	SA	22457.09	41.30	1000	0.99	1000	10

Table 4.7. Algorithm comparison with charging case - Prob Type 3.

5. CONCLUSION

In general terms, the research presented in this thesis has shown that drone technology has the potential to be used in post-disaster infrastructure assessment and that this can be extended to a wider scope by considering the charging status of the drone. By integrating the sequential testing approach to post-disaster situations, the drone was also included to identify safe roads as soon as possible. To this end, we used local search and simulated annealing approaches to show how they work with the right parameters and data-set setups.

In this research, we assume that a certain amount of research has been done in advance of the post-disaster and that it has been determined which routes will be included in this testing process. In principle, safe points have already been identified in the pre-disaster and the routes to these safe points have been determined. Correspondingly, the points on the road where the drone will be tested, bridges, viaducts, etc., are assumed to be already known in the pre-disaster through specific surveys and their current state, and their analysis and data are assumed to be available to allow image processing to test the drone's integrity later on. If the charging status is included, it is known at which points it can be charged and charging stations are set up at those points in the pre-disaster. In fact, with all these assumptions, the drone starts testing in post disaster. In this study, the drone develops a strategy to test only a given path, even though there might be multiple other possibilities. Another restrictive assumption is the independence of the status of the elements that constitutes the routes. In a post-disaster situation, this will not be typically true if for close by segments of the route and needs to be taken into consideration.

Looking forward, the scope for expanding this research is vast and multifaceted. One major restrictive assumption in this research is that we assume that a route is safe if and only if all the elements on the route are safe. In real life, the road networks are more general. It would be quite difficult to consider the most general road networks. It would be a promising idea to study some more general but still special networks and try to

adapt the results for these networks to the general case. One promising direction is the enhancement of the decision-making process in choosing the most strategic routes for drone assessment. This could involve developing algorithms that not only determine the best paths for initial assessment but also dynamically adapt to changing circumstances, such as newly discovered impassable routes or changing priorities in the disaster area.

Further, the integration of other heuristic methods, such as Tabu Search, or exploring exact optimization techniques could add depth to this research. Such advancements could refine the balance between speed and accuracy, ensuring that the most critical areas are assessed first in a disaster scenario. Additionally, extending the research to consider multi-route assessments could offer a more holistic view of the disaster impact, enabling a more comprehensive response strategy.

Another significant area for future exploration is the detailed consideration of drone operational parameters, such as battery life, charging conditions, and the establishment of potential charging stations in the field. Addressing these logistical challenges would not only improve the effectiveness of drone usage in post-disaster scenarios but also enhance the overall resilience of emergency response systems. Moreover, the methodologies developed in this thesis can be adapted beyond the realm of disaster response. Applications in areas such as facility management, urban planning, and environmental monitoring are conceivable, where rapid, efficient, and accurate assessments are equally crucial.

In conclusion, this thesis lays a robust foundation for future research and practical application in drone technology for post-disaster assessment and beyond. It is an important step towards creating more resilient, efficient, and responsive emergency management systems, capable of addressing the complex challenges posed by natural disasters and other crises.

REFERENCES

1. Omenzetter P., Ramhormozian S., Mangabhai P., Singh R., "A framework for rapid post-earthquake assessment of bridges and restoration of transportation network functionality using structural heatlh monitoring", *Proceedings of SPIE - The International Society for Optical Engineering 8692*, 2013.

2. Mohd Daud, S.M.S., Mohd Yusof, M.Y.P., Heo, C.C., Khoo, L.S., Singh, M.K.C., Mahmood, M.S., Nawawi, H., "Applications of drone in disaster management: A scoping review", *Science Justice, Volume 62, Issue 1, 30-42*, 2022.

3. 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.

4. 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.

5. Chowdhury, S., Shahvari, O., Marufuzzaman, M., Li, X., Bian, L., "Drone routing and optimization for post-disaster inspection", *Computers Industrial Engineering, Volume 159*, 2021.

6. Pimentel, S.B., "Drone Route Optimization using Constrained Based Local Search", *Thesis (MSc), Universidade Nova de Lisboa*, 2018.

7. Ünlüyurt, T., "Sequential Testing of Complex Systems: A Review", *Discrete Applied Mathematics*, Vol. 142,(1-3), pp. 189–205, 2004.

8. Mitten, L., "An Analytic Solution to the Least Cost Testing Sequence Problem", *The Journal of Industrial Engineering*, p. 17, January-February 1960.

9. Ben-Dov, Y., “A Branch and Bound Algorithm for Minimizing the Expected Cost of testing Coherent Systems”, Vol. 7, pp. 284–289, 1981.

10. Boros, E. and T. Ünlüyurt, “Sequential testing of series-parallel systems of small depth”, M. Laguna and J. Velarde (Editors), *Computing Tools for Modeling, Optimization and Simulation*, pp. 39–74, Kluwer, 2000.

11. Segev, D. and Y. Shaposhnik, “A Polynomial-Time Approximation Scheme for Sequential Batch Testing of Series Systems”, *Operations Research*, Vol. 70, pp. 1153–1165, 2022.

12. Teller, R., M. Zofi and M. Kaspi, “Minimizing the average searching time for an object within a graph”, *Computational Optimization and Applications*, Vol. 74, pp. 517–545, 2019.

13. Arora, S. and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, NY, USA, 2009.

14. H. R. Lourenço, O. C. Martin, and T. Stützle, “Iterated local search”, *Handbook of Metaheuristics (International Series in Operations Research Management Science), vol.57*, 2003.

15. Vaessens R.J.M., Aarts E.H.L. and Lenstra J.K., “A local search template”, *Computers & Operations Research, 25(11), 969-979*, 1998.

16. Talbi E., *Metaheuristics: from design to implementation*, John Wiley Sons, Hobokeny, 2009.

17. Eglese, R.W., “Simulated annealing: A tool for operational research”, *European Journal of Operational Research 46, 271–281*, 1990.

18. Stewart MG, “Reliability-based assessment of ageing bridges using risk ranking and life cycle cost decision analyses”, *Reliability Engineering & System Safety*

74(3):263–73, 2001.

19. Omenzetter P., "Identification of unusual events in multi-channel bridge monitoring data", *Mech Syst Sig Process*, 2004.

20. H Shakouri G.K Shojaee, and B. T M., "Investigation on the choice of the initial temperature in the Simulated Annealing: A mushy state SA for TSP[C]", *Mediterranean Conference on Control and Automation IEEE Computer Society, 1050-1055*, 2009.

21. M.-W. Park, Y.-D. Kim, "A systematic procedure for setting parameters in simulated annealing algorithms", *Computers & Operations Research, 25(3), 207–217*, 1998.

APPENDIX A: Algorithms and Results

Algorithm 6 Local Search - Swap Sequence

Require: A network and a current sequence of nodes.

Ensure: The best swapped sequence with its corresponding expected cost.

```
function SwapSequence(network, currentSeq)
    seqCosts ← {}
    swappedBestCost ← ∞
    swappedBestSeq ← None
    i ← 1
    while i ≤ length(currentSeq) − 3 do
        j ← i + 1
        while j ≤ length(currentSeq) − 2 do
            switchedSeq ← currentSeq
            Swap switchedSeq[i] and switchedSeq[j]
            switchedCost ← CalculateExpectedTime(network, switchedSeq)
            seqCosts[tuple(switchedSeq)] ← switchedCost
            if switchedCost < swappedBestCost then
                swappedBestCost ← switchedCost
                swappedBestSeq ← switchedSeq
            end if
            j ← j + 1
        end while
        i ← i + 1
    end while
    return swappedBestSeq, swappedBestCost
end function
```

Algorithm 7 Simulated Annealing - Annealing Swap Operation

Require: A network represented as a route and a current sequence of nodes $currentSeq$.

Ensure: A swapped sequence of nodes and its corresponding expected cost.

```
function AnnealingSwap(network, currentSeq)
    i ← RandomRange{1, LengthcurrentSeq − 1}
    j ← RandomRange{1, LengthcurrentSeq − 1}
    while i = j do
        j ← RandomRange{1, LengthcurrentSeq − 1}
    end while
    swappedSeq ← currentSeq
    Swap swappedSeq[i] and swappedSeq[j]
    swappedCost ← CalculateExpectedTime{network, swappedSeq}
    return swappedSeq, swappedCost
end function
```

Inst.	Alg.	Time (ms)	Result	InitialTemp	CoolRate	TempIter	StopTemp
10	SA	224.21	60.47	800	0.8	750	10
10	SA	295.92	59.01	800	0.8	1000	10
10	SA	430.02	58.47	800	0.8	1500	10
10	SA	441.20	59.66	800	0.9	750	10
10	SA	586.44	58.67	800	0.9	1000	10
10	SA	861.77	58.77	800	0.9	1500	10
10	SA	890.52	57.99	800	0.95	750	10
10	SA	1160.79	58.63	800	0.95	1000	10
10	SA	1753.61	58.49	800	0.95	1500	10
10	SA	4345.19	58.01	800	0.99	750	10
10	SA	5747.78	58.08	800	0.99	1000	10
10	SA	8933.19	57.98	800	0.99	1500	10
10	SA	241.93	60.58	900	0.8	750	10
10	SA	304.28	58.61	900	0.8	1000	10
10	SA	444.61	58.35	900	0.8	1500	10
10	SA	457.14	59.09	900	0.9	750	10
10	SA	595.28	59.13	900	0.9	1000	10
10	SA	875.765	58.08	900	0.9	1500	10
10	SA	907.31	59.77	900	0.95	750	10
10	SA	1206.49	58.82	900	0.95	1000	10
10	SA	1777.98	59.40	900	0.95	1500	10
10	SA	4422.04	57.98	900	0.99	750	10
10	SA	5800.96	57.69	900	0.99	1000	10
10	SA	8977.97	58.06	900	0.99	1500	10
10	SA	234.94	60.30	1000	0.8	750	10
10	SA	306.33	58.33	1000	0.8	1000	10
10	SA	434.91	58.08	1000	0.8	1500	10
10	SA	457.92	58.82	1000	0.9	750	10
10	SA	607.93	58.15	1000	0.9	1000	10
10	SA	894.55	58.32	1000	0.9	1500	10
10	SA	917.97	59.13	1000	0.95	750	10
10	SA	1193.24	58.37	1000	0.95	1000	10
10	SA	1779.64	58.08	1000	0.95	1500	10
10	SA	4528.01	58.33	1000	0.99	750	10
10	SA	5995.72	57.67	1000	0.99	1000	10
10	SA	10664.06	57.67	1000	0.99	1500	10

Table A.1. Parameter selection for Simulated Annealing - extended.