Vehicle Routing Problem with Drones

Ashwini Doke

Student ID: 20493897

School of Computer Science, University of Nottingham.

Submitted September 2023, in partial fulfilment of the conditions for the award of the degree of MSc Computer Science (Artificial Intelligence)

Table of Contents

Abstract

Drones have emerged as a revolutionary factor in today's logistics industry, offering potential advancements in the realm of last-mile deliveries. This dissertation presents a novel prototype for incorporating drones into the classic Vehicle Routing Problem (VRP) using innovative computational algorithms. The study begins using the nearest neighbour algorithm to visit all cities, then refines routes using the two-opt heuristic. Further complexities are addressed by combining Matheuristics and a Mixed Integer Linear Programming (MILP) approach. This method not only optimises the path but also ensures smooth coordination between drones and trucks, eventually aligning their respective trips for optimum performance. To ensure practicality, provisions for real-world limits, such as drone-free zones, are included. Through a systematic methodology that spans from data preparation to visual representation, the research sheds light on the significant advantages of drone integration in delivery systems, primarily emphasizing decreased truck tour durations and distances as well as the inherent environmental benefits through potential reductions in carbon emissions. The study's major finding is the increased benefits of using Matheuristics, particularly in extensive delivery circumstances. The framework proposed exhibits adaptability, accommodating changes in operational parameters efficiently. Conclusively, the prototype paints a promising future for logistics, emphasising a balanced synergy of drones and enhanced optimization for next-generation delivery systems, all while echoing the imperatives of environmental sustainability.

Keywords: Vehicle Routing Problem; Matheuristic; Two-Opt Heuristic; Mixed Integer Linear Programming; MTZ formulation; Drones.

Acknowledgements

First and foremost, I would like to express my profound gratitude to my supervisor, Dr. Andrew Parkes. His continuous guidance, unwavering support, invaluable insights, and constructive criticism have been crucial throughout this research journey. His feedback and challenges pushed me beyond my boundaries, guiding me towards achieving the objective of this work.

I would also like to extend a heartfelt thank you to my personal tutor Damla Kilic and all the professors who have been a part of my academic journey. Each one has contributed to my learning, imparting in me the knowledge and skills that have shaped my understanding and passion for the subject.

Furthermore, my sincere thanks go out to my family and friends. Their constant encouragement, patience, and belief in my capabilities have been the pillars upon which I leaned during the challenging moments of this process. Their support throughout this year has been immeasurable.

Lastly, a special mention to my loyal companion, my dog, who has been by my side, bringing comfort, and silent encouragement during the long hours of work. His presence served as a daily reminder of the simpler joys of life, making the journey more bearable.

1. Introduction

The logistical domain has always been a focal point for introducing innovation and efficiency, particularly within the fast-evolving backdrop of the 21st century. Today's global economy depends largely on seamless transportation, and as a result, there's a substantial emphasis on optimising delivery systems. Over the past few years, drones, with their agility and precision, have emerged as potential game-changers in the logistics and delivery sector. Amidst the bustling streets and divergent terrains, the amalgamation of drones into conventional delivery systems is a topic of avid interest. In the realm of computational logistics, finding the best route in the shortest amount of time is frequently the holy grail.

The challenges of efficient logistics and delivery systems have evolved, especially considering the growing concerns related to environmental sustainability. There are numerous real-world issues centred around the delivery of small products to customers, predominantly facilitated by delivery drivers in vehicles that may contribute significantly to carbon emissions. Drones have surfaced as a potential alternative, potentially marking a paradigm shift in last-mile deliveries.

Such integration is crucial in today's era, marked by the imperatives of tackling climate change and the overarching aim of reducing carbon emissions. This is not only a step forward in environment-friendly logistics but also offers a promise of efficiency and cost savings. Particularly for small businesses in remote locations, this could mean bridging the gap between accessibility and affordability.

This research aims to conceptualise and design a prototype that seamlessly integrates drones into the conventional vehicle routing conundrum. Understanding the complexity of this task, famously known as NP-hard in nature, a robust and systematic approach has been carried out. Several tried-and-tested algorithms are included in this prototype each of which contributes a unique set of strengths.

The two-opt heuristic is a simple local search method used for solving the travelling salesman problem and related routing problems (Manthey & Veenstra, 2013). The motive behind the two-opt heuristic is to take a given route and iteratively improve it by reversing the order of two cities in the route if it results in a shorter tour (Hougardy & Brodowsky, 2021). This process continuous until no further improvements are made. The heuristic is named "two-opt" because it involves removing two edges and reconnecting them differently.

Matheuristics are hybrid algorithms that combine the strengths of mathematical programming (or exact optimization) techniques and heuristics. These algorithms are designed to exploit the robustness of heuristic methods and the precision of mathematical programming to solve complex combinatorial optimization problems efficiently (Gunawan et al., 2021).

By harnessing the power of the two-opt heuristic (Manthey & Veenstra, 2013) and melding it with the insights of Matheuristic techniques (Gunawan et al., 2021), a foundation is crafted. The initial route was formed through the nearest neighbour method. This method involves selecting the closest city to the current one and repeating the process until all cities are covered. Although this approach can provide quick results, there may be better solutions available. Nevertheless, it's a reliable foundation to begin with (Josiah, Ojo et al., 2013).

However, to polish this initial route further, the prototype employs two-opt heuristics (Manthey & Veenstra, 2013). Originating from the world of combinatorial optimization, it's a fine-tuning process that irons out any inefficiencies in the original tour and yields a more streamlined path for the truck.

With this tour as the basis, the next phase is to pinpoint cities that are apt for drone deliveries. The identification rests on multiple criteria, ranging from potential travel time savings to the overall feasibility of drone missions, factoring in specific operational constraints inherent to the drones.

Once this framework is laid, the focus switches to seamlessly integrating the basic two-opt truck tour with the designated drone paths. This type of synthesis ensures that the drone and truck activities are beneficial, maximising efficiency. After utilizing the two-opt heuristic to develop the near-optimal tour, a specific segment of this route that stands out as having room for even further optimization is chosen for performing Matheuristic. Essentially, Matheuristics takes this targeted section of the two-opt route and, using the ability of the Mixed Integer Linear Programming (MILP) model, aims to enhance its efficiency even further. To navigate the complexities of this model, the Gurobi solver (Gurobi Optimization, 2023) is employed, ensuring precision in the Matheuristic optimisation (Gunawan et al., 2021). This allows for a deeper dive into potential efficiencies fine-tuning the route to near perfection. However, the model is also built with an awareness of real-world challenges.

In the upcoming chapters, this dissertation focuses on a comprehensive exploration of the integration of drones within the Vehicle Routing Problem (VRP). To start with the Literature Review section searches into previous academic endeavours, shedding insight into previous approaches and findings, leading to the design decisions, which outline the approaches examined and finalised. Following this, the Methodology elucidates the systematic techniques and processes employed in this research, offering a clear perspective into the project's foundation. The Experimental Setup describes the practical aspects and metrics used to evaluate the prototype. Following that, the Results and Evaluation section shows the collected data while assessing its significance. The Conclusion summarises the important findings capturing the essence of the research.

2. Literature review

The Vehicle Routing Problem (VRP), a renowned operational research challenge, involves optimally determining routes for a fleet of vehicles to service a given set of customers. The integration of modern technologies such as drones and Vehicles has introduced further complexity, as researchers look for efficient, extensible, and cost-effective solutions. The following review of literature outlines the development of research in this area, emphasizing the various constraints and innovations introduced over time.

At the head of this evolution is the study by Kuo et al., (2022) which addressed the Vehicle Routing Problem with Time Windows and Drones (VRPTWD). What set this work apart was the unique concept of trucks functioning as launchpads for drones. In a bid to cut travel costs, they introduced a Mixed-Integer Programming (MIP) model. Their research was deeply rooted in the VRP literature. They utilized a Mathematical formulation inspired by Sacramento et al., 2019 while also addressing the overlooked time window setups by Di Puglia Pugliese & Guerriero, 2017. Their approach extended the MIP framework discussed in Lan et al., (2020), showing their study's deep connections with past works.

Looking into the complexity of the Flying Sidekick Travelling Salesman Problem (FSTSP) and the Parallel Drone Scheduling Travelling Salesman Problem (PDSTSP) is the work by Murray & Chu, 2015. Their focus was on scenarios where distribution centres are adjacent to customers. Acknowledging the NP-hard nature of these problems, they presented both a MILP formulation and a heuristic solution. Their work resonated with the challenge presented by Lin, 2011, where light resources were transported by heavy vehicles.

Another aspect of modern-day challenges in VRP is energy consumption, a theme central to Kyriakakis et al., 2022. Their paper tackled the Electric Vehicle Routing Problem with Drones (EVRPD), to reduce energy use while efficiently managing drone payloads. By employing algorithms from the Ant Colony Optimization (ACO) frameworks and various Variable Neighbourhood Descent (VND) operators, the study aimed to provide optimal solutions. Their work is based on earlier studies like Conrad & Figliozzi, 2011's VRPD and Erdogan & Miller-

Hooks, 2012's Green Vehicle Routing Problem with Drones (VRPD). The practical usage and application of their model were demonstrated through scenarios in Chania, Greece, showcasing the integration of electric vehicles and Amazon's drone fleet.

Considering a renewed perspective on the parallel drone scheduling TSP (PDSTSP) is the work by Dell'Amico et al., 2020. Their target was to optimize how a truck could service a set of customers while drones tackled others within their operational range. The aim was to minimize the completion time for all deliveries. The authors advanced a simplified MILP model combined with a set of matheuristic algorithms. They not only accepted the MILP approach proposed by Murray & Chu, 2015 but also considered Ham, 2018's exploration of multi-drone, truck, and depot challenges.

Taking further the boundaries of optimization methodologies is the work by Montemanni & Dell'Amico, 2023, where they championed a Constraint Programming model for PDSTSP. This approach aimed to revamp drone scheduling for urban and rural deliveries, making sure of a reduction in total trip time.

VRP variants have been addressed using various solution techniques. For example, Kuo et al., 2022, and Murray & Chu, 2015, both used MIP models for their respective challenges, while Dell'Amico et al., 2020, combined MILP with matheuristic algorithms. Heuristic methods have also seen traction, such as the combination of variable neighbourhood search with hybrid cheapest insertion heuristics by Kytöjoki et al., 2007. The emerging metaheuristic strategies, like the ACO frameworks, were utilized by Simensen et al., 2022, who proposed a method named Hybrid Genetic Search with Ruin-and-Recreate (HGSRR) for the Capacitated Vehicle Routing Problem (CVRP). Their approach synergized the strengths of the hybrid genetic search algorithm with the Ruin-and-Recreate based Large Neighbourhood Search (LNS). The authors discuss the origins of metaheuristics (Glover, 1986) and CVRP (Dantzig & Ramser, 1959). The scale of the issue that can be solved to demonstrate optimality is also discussed as being limited by the NP-hardness of CVRP and its expansions (Lenstra & Kan, 2006).

In the pursuit of arriving at optimized solutions for the Vehicle Routing Problem (VRP) and its many variants, researchers have not only developed innovative techniques and algorithms but have also laid significant emphasis on robust comparative analyses and evaluations of their methodologies.

Surveys like that by Khoufi et al., 2019, which covered UAV integration in the VRP between 2014 and 2019, delve deep into the evolving landscape of Unmanned Aerial Vehicles (UAVs) and their integration within the VRP paradigm. With a specific focus on UAVs as companions in TSP and VRP, the survey serves as a comprehensive guide for researchers and practitioners alike. Factors like weather conditions, payload constraints, energy limitations, and collision avoidance are all analyzed in depth. By dissecting classical routing challenges and juxtaposing them against UAV constraints, the authors offer fresh insights into optimization solutions that account for a diverse range of scopes, parameters, and techniques. Their work culminates in a poignant assertion about the ample room for improvement in objective and case sets, emphasizing the importance of addressing the highlighted aspects in future UAV routing optimization challenges.

(Laporte, 2007) also made a significant contribution by tracing back the origins of VRP to the foundational work of (Dantzig & Ramser, 1959). This narrative presents an overview of various methodologies employed to tackle conventional VRPs, discussing the merits and demerits of exact algorithms, classical heuristics, and metaheuristics. Of particular note is the author's emphasis on the dual goals of simplicity and flexibility in algorithmic design. This work also touches upon the often-overlooked aspect of user-controlled parameters in algorithmic design, suggesting a cautious approach to their over-integration.

Similarly, the study by (Gulczynski et al., 2011) offers an illuminating perspective on real-world applications of VRP, showcasing the practical implications of merging the Multi-Depot Vehicle Routing Problem (MDVRP) with the Split Delivery Vehicle Routing Problem (SDVRP), hence they propose the Multi-Depot Split Delivery Vehicle Routing Problem (MDSDVRP). They presented a two-phase integer programming-based heuristic solution for the MDSDVRP, taking vehicle capacity and time windows into account. Their focus on waste collection and inventory distribution challenges brings to the fore the tangible benefits of optimized routing solutions in everyday scenarios. Their work is also pivotal in amalgamating the rich histories of MDVRP and SDVRP, positioning them as the first to merge these two distinct streams.

The evolution of the Vehicle Routing Problem (VRP) into a complex tapestry integrating new technologies, such as drones, underscores the dynamic nature of logistical challenges in today's world. As depicted in the literature, while the foundational concepts of VRP have been long-established, the emergence of drone technology in the logistics domain requires novel approaches and solutions. These newer challenges interlace algorithmic complexity, real-world constraints (like drone-prohibited areas), and the pressing need for optimized routes. This exploration across the body of existing literature reveals that while some attempts have been made to bridge traditional VRP techniques with drone integrations, a holistic solution remains elusive.

The core of the research is the junction of heuristic methods, known for their intuitive quick fixes, with the precision of mathematical programming. Notably, the collaboration of the two-opt heuristics (Manthey & Veenstra, 2013) with Matheuristics, set against the backdrop of the nearest neighbour algorithm (Josiah, Ojo et al., 2013), offers a subtle approach not widely discussed in the literature. The unique interplay between the truck routes and drone paths, made even more complex by real-world restrictions, underlines the need for a systematic methodology – a recurring theme highlighted across various studies.

2.1 Factors influencing decision making

Formulating and refining the model process to come up with optimal delivery routes involves several iterations and considerations. This segment focuses on providing ideas of the decisions made and the underlying reasons for them.

The Traveling Salesman Problem (TSP) inherently presents challenges, one of which is the emergence of sub-tours. These sub-tours are segments of a route that begin and end at intermediate points rather than the designated start and endpoint. Such inclusions lead to inefficient solutions that do not encapsulate the true essence of the TSP: to identify the shortest possible route visiting each city exactly once before returning to the commencement city forming the Hamilton Cycle (Rahman & Kaykobad, 2005).

To prevent sub-tours, Subtour elimination (STE) was adopted. By using STE, the optimization process can ensure that sub-tours are identified and eliminated, making the solution both feasible and optimal (Gutekunst & Williamson, 2020). One effective mechanism for sub-tour elimination is the branch and cut method (Mitchell, 2011). This strategy comprises creating branches to explore various routes and cutting those that lead to sub-tours, ensuring that the solution remains efficient and accurate (Mitchell, 2011).

However, while the branch and cut mechanism is favourable, it's important to note that the potential of the branch and cut method, especially when combined with matheuristic approaches, is tempered by the intricacies of its integration. The necessity of constantly producing and appending cutting planes (constraints) to reduce the search space gives rise to the branch and cut approach's complexity. Understanding the problem deeply involves understanding its underlying mathematical principles. Additionally, an adequate framework is needed to effectively manage the resulting constraints. Consistent algorithm fine-tuning is crucial, especially considering how the addition of cutting planes impacts the linear programming solver's performance. Rigorous testing is also essential to ensure both the validity of the constraints and the improvement of solution quality (Vercesi, Eleonora et al., 2023). Given these complex issues, considering branch and cut method integration within the limited three-month time frame for this dissertation seemed overly ambitious.

MTZ formulation is carefully crafted to prevent the formation of sub-tours. Named after, Miller, Tucker, and Zemlin, the MTZ formulation introduces auxiliary variables that capture the sequence in which cities are visited. This strategy inherently suppresses sub-tours by leveraging these variables, guaranteeing a valid and efficient solution. (Miller et al., 1960).

Although with many strong formulations, there are challenges which can arise. The MTZ formulation (Miller et al., 1960), despite its strengths, requires significant computational time. But, when paired with matheuristic techniques utilizing Mixed Integer Linear Programming (MILP), this issue can be somewhat mitigated, allowing for further optimization. Therefore, given its robustness and synergy with the chosen optimization approach, it was decided to incorporate the MTZ formulation (Miller et al., 1960) for sub-tour elimination (Gutekunst & Williamson, 2020) and accurate tracking of the truck's tour in this investigation.

The initial strategy involved fixing the drone cities, dictating specifically which cities the drone would cater to. This mechanism bypassed the model's freedom to decide the optimal city for drone operation. Although it simplified the model to some extent, the rigidity of 'hard coding' the drone's destinations was quickly recognised as a potential limitation. To enhance the model's flexibility and allow for a more nuanced decision-making process, the focus shifted towards empowering the model to determine the drone city.

By not programming the drone's delivery city, the model had the autonomy to select the most efficient city for drone deliveries, while the truck took charge of the remaining cities. However, an important observation was the exponential rise in computation times as the number of cities increased. This escalation was majorly attributed to the MTZ formulation employed in the model. MTZ, while ensuring the formation of a Hamilton cycle and effectively eliminating the risk of sub-tours, simultaneously expands the computational burden, especially with a growing number of cities. Though the rise in computation times for the model choosing one drone city was evident, it implied that selecting multiple drone cities would not be feasible due to the anticipated drastic surge in computational demands (Miller et al., 1960) (Rahman & Kaykobad, 2005).

Confronted with escalating computation times, the two-opt heuristic (Manthey & Veenstra, 2013) emerged as a viable solution, particularly for larger city instances. Renowned for constructing near-optimal tours in reduced timeframes, its integration appeared promising. The strategy devised was to let the two-opt heuristics (Manthey & Veenstra, 2013) establish a preliminary tour, followed by employing Matheuristics within the MILP framework to refine this tour. By 'freezing' certain segments of the two-opt generated tour, Matheuristics could intensively focus on and optimise the remaining portions. This combination aimed to harness the best of both worlds: the rapidity of heuristics and the precision of MILP, ensuring a highly optimised tour with significantly decreased computation time.

In the realm of the two-opt heuristic, the quality of the initial tour supplied directly influences the optimality of the output. Various algorithms were evaluated for this purpose, including:

a. Nearest Neighbour Method
b. Nearest Insertion

c. Random Tour
d. Clarke-Wright Savings Algorithm (CWS)

Of all these algorithms, the Nearest Neighbour method was settled upon as the most suitable after rigorous experimentation and analysis. The objective was to start and conclude the tour at an identical city, culminating in the formation of a Hamilton cycle (Rahman & Kaykobad, 2005). The Random tour and the CWS method fell short of this requirement.

The CWS Algorithm (Doyuran, 2008), while effective in many applications, didn't fit seamlessly into this model's demands. Its savings approach didn't naturally lend itself to the Hamiltonian cycle requirement, and so was the problem with random tours (Basel et al., 2001). Nearest Insertion, on the other hand, while closer in approach to the Nearest Neighbour method, often yielded tours that weren't as optimal, taking more computational time in the subsequent two-opt phase. The Nearest Neighbour method emerged as the most promising, especially in providing consistency and a quality starting point for the two-opt to work upon (Josiah, Ojo et al., 2013).

To incorporate multiple drone cities into the model, the two-opt heuristic tour was analysed to identify potential drone routes. Once these routes were determined, adjustments were made to the two-opt tour to integrate these drone routes. These adapted drone routes, along with segments of the initial two-opt tour, were then subjected to Matheuristics for further refinement. The process of spotting potential drone routes involved pinpointing the two closest truck cities to any given drone city. These cities acted as the start and end points for the drone, ensuring that the drone's journey aligned with the truck's route.

The initial method of identifying drone tour opportunities, grounded in proximate cities, didn't yield identification of savings. A more nuanced savings calculation was necessitated. Thus, a time matrix was instituted, with the formula:

$time = \frac{distance}{speed}$

Incorporating speeds of 35 mph for trucks and 50 mph for drones.

The savings were then discerned using the formula:

savings = truck effective time – drone tour time.

Here, the truck's effective time encapsulated its journey time between cities, excluding the drone's city, and the drone tour time was the duration taken by the drone from one city, through the drone city, to another.

A positive savings value meant that the truck's route was more time-consuming than the drones for that segment, and vice versa for a negative value. High savings, while optimal, posed logistical challenges, such as drones potentially having to wait for trucks. To negate this, constraints were introduced: Drones shouldn't idle for over 10 minutes awaiting trucks. The drone's waiting time should be in the positive, preventing trucks from arriving prematurely. The drone's endurance was fixed at 30 minutes. Additionally, the drone's operational radius was capped at 20 miles.

For smaller instances, there was a congruence between the outputs of the two-opt heuristic and the Matheuristics, whether the latter was supplied with a segment or the entirety of the problem. However, with medium-sized instances, the Matheuristics demonstrated a clear edge over the two-opt heuristic, especially when tasked with the complete problem. As the city instances swelled, the efficacy and precision of the Matheuristics MILP model became increasingly evident, showcasing its computational robustness and optimisation ability.

Initially, the selection of segments for Matheuristics was manual, with other parts of the tour being fixed or "frozen". An automated approach was later adopted, where the start and end points of a desired segment could be specified. The system then identified the cities lying between these points, determining the segment based on these cities and their corresponding indexes. The segments outside this chosen path, which extend from the tour's start city to the segment's beginning and from the segment's end to the tour's closing point (also its starting point), were recognised as the remaining segments.

Opting for only one segment in Matheuristics seemed to restrict the scope of optimisation. To address this, the inclusion of multiple segment selection was introduced, allowing for a wider range of possibilities and deeper fine-tuning of the tour.

While multi-segment selection expanded the model's flexibility, inputs for each segment still had to be provided individually. This required a thorough manual analysis of the truck's tour to ensure the correct sequence of segment start and endpoints. To streamline this process, a system was developed to accept region-based inputs by simply specifying the minimum and maximum x and y coordinates for a desired region. The segments within that region, needing optimisation via Matheuristics, were then automatically determined.

The methodology of region-based input selection for Matheuristics inspired a similar approach for defining drone-prohibited regions. The system was further enhanced to consider multiple drone-restricted areas. Inputs for each prohibited region's coordinates could be specified, and cities within these boundaries were automatically identified and appended to the list of *DRONE_PROHIBITED_CITIES*. This facilitated a dynamic and flexible way to manage drone activities according to specific requirements.

A shift from matplotlib to Plotly for plotting results was initiated due to Plotly's capacity to render interactive and visually appealing graphs. Distinct visual markers were introduced: trucks were represented with solid black lines and drones with dashed black lines. Cities where drone activity was restricted were highlighted with red crosses, while standard drone-permitted cities were denoted with black dots.

Plotly's interactive features provide a dynamic experience. Specific tours can be isolated and examined by simply clicking on the corresponding legend. Enhanced zoom and drag features further support detailed analysis, allowing for panning and zooming adjustments, and clarifications.

Seaborn was chosen for visually representing and analysing the results. Its varied colour palettes and diverse features made it a perfect tool for crafting detailed and informative bar plots.

2.2 Technology Review

Gurobi is a leading optimization solver that has gained recognition in the optimization circle. It is eligible to tackle complex optimization problems which has proven it to be a preferred choice for many researchers and professionals. For this review context, Gurobi was involved in solving a vehicle routing problem involving drones, formulated as a mixed-integer linear programming (MILP) model.

2.2.1 Features and Capabilities

Performance: Gurobi is known for its remarkable speed and efficiency. When compared to other solvers, Gurobi has been involved in the delivery of optimal solutions in reduced time, making it invaluable for complex problems (Gurobi Optimization, 2023).

Flexibility: The adaptability of Gurobi is exceptional. Along with its work for MILP, it is also proficient in handling various optimization problems, from linear to quadratic programming (Gurobi Optimization, 2023)

Integration with Python: The *gurobipy* library makes the Integration of Gurobi with Python easy. This integration makes instinctive model formulation, modification, and solution extraction (Gurobi Optimization, 2023).

2.2.2 Application in Dissertation

The focus was on a vehicle routing problem where drones were given the task of package delivery to various locations, optimizing routes for reducing costs and time of delivery. The problem was expertly formulated as a MILP model, using binary decision variables to represent drone routes and continuous variables for delivery times.

Using the *gurobipy* library, the model was implemented in Python. The library's functions and methods were involved in defining limitations, setting objectives, and deriving solutions.

2.2.3 Experience

Ease of Use: Gurobi's user interface is intuitive, and its documentation is thorough, aiding users in navigating the software (Gurobi Optimization, 2023). Examples presented in the documentation have proved beneficial.

Support and Community: The Gurobi community has always provided active support. An abundance of forums and resources are available, offering solutions to common.

Licensing: An academic license was used, which Gurobi generously offers to students and researchers, further exhibiting its commitment to the academic community.

2.2.4 Comparison with Other Solvers

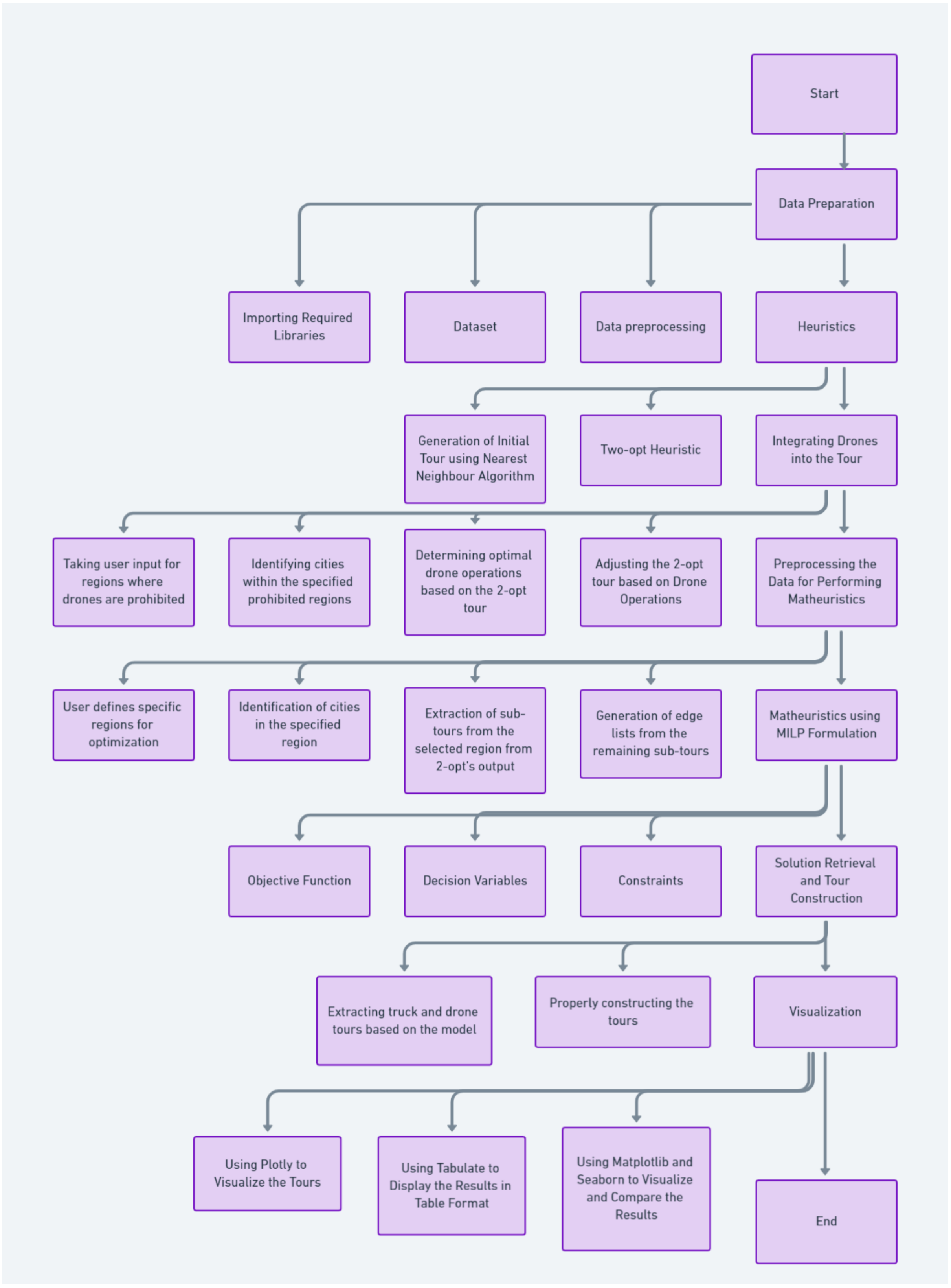
While Gurobi was the primary solver which was used, it should be noted that it often performs better than its competitors in terms of speed and solution quality. However, the choice of solver can be specific to the problem, and benchmarking solvers for specific applications is always recommended.

3. Methodology

The methodology portrays the systematic approach undertaken to address the vehicle routing problem with drones. It encloses data preparation, modelling considerations, and the analytical techniques employed. Figure 3.1 depicts the structured procedure used when addressing the vehicle routing problem with drone integration. Each stage, as well as its sub-processes, is defined to provide clarity on the strategy adopted.

Considering the computational demands of MILP and the Gurobi solver, it's essential to emphasize the computational framework used for this research. For this research, a powerful computational environment was crucial to handle the complex modelling and optimization tasks.

The research was led on a system boasting an 11th Gen Intel(R) Core (TM) i5-1135G7 processor, clocking at 2.40GHz 2.42 GHz. Paired with this was an 8.00 GB RAM, ensuring seamless multitasking and efficient processing. Furthermore, the system was equipped with a 64-bit operating system and an x64-based processor, optimizing software compatibility and performance. These specifications resulted in timely and efficient processing, especially during intensive optimisation runs.



Start
Data Preparation
Importing Required Libraries
Dataset
Data preprocessing
Heuristics
Generation of Initial Tour using Nearest Neighbour Algorithm
Two-opt Heuristic
Integrating Drones into the Tour
Taking user input for regions where drones are prohibited
Identifying cities within the specified prohibited regions
Determining optimal drone operations based on the 2-opt tour
Adjusting the 2-opt tour based on Drone Operations
Preprocessing the Data for Performing Matheuristics
User defines specific regions for optimization
Identification of cities in the specified region
Extraction of sub-tours from the selected region from 2-opt's output
Generation of edge lists from the remaining sub-tours
Matheuristics using MILP Formulation
Objective Function
Decision Variables
Constraints
Solution Retrieval and Tour Construction
Extracting truck and drone tours based on the model
Properly constructing the tours
Visualization
Using Plotly to Visualize the Tours
Using Tabulate to Display the Results in Table Format
Using Matplotlib and Seaborn to Visualize and Compare the Results
End

Figure 3.1: Flowchart illustrating the comprehensive methodology employed

3.1 Data Preparation

The uprightness of any quantitative analysis is upheld by the careful preparation of its data, considering its accuracy, comprehensiveness, and relevance to the research questions at hand.

This section makes clear the initial steps considered to condition the data for the subsequent optimisation analyses.

3.1.1 Importing Required Libraries

To build, evaluate, and optimise the model associated with the vehicle routing problem with drones, some specific Python libraries were selected.

- *numpy*: At the core of spatial computations, *numpy* was crucial for the creation and manipulation of data structures.
- *gurobipy*: Route optimisation, being a complex computational task, required the use of Gurobi's optimiser. With version 10.0.2.0 of *gurobipy*, advanced linear and mixed-integer programming capabilities were taken into consideration to devise the most optimal routes for the drones, ensuring minimal distance and time expenditure.
- *plotly*: Visual representation assists in both the comprehension and presentation of data. Using *plotly.graph_objects*, it's possible to generate interactive and detailed plots, providing insightful visuals of the drone routes and highlighting the optimisation results.
- tabulate: In research, clarity in presenting data is of high importance. Tabulate impressively serves this purpose by rendering lists of data into well-organized tables.
- seaborn & matplotlib: Visualization is not just restricted to interactivity alone. Seaborn, built on top of matplotlib, boosts the aesthetics and versatility of data plots. While matplotlib lays the foundation with comprehensive plotting tools, seaborn enhances it with attractive themes and colour palettes. Together, they were contributing to crafting bar graphs for method comparisons, thus offering a deeper dive into the data's insights.

These libraries, with their different functionalities, collaboratively enabled an all-inclusive approach to the research, ensuring accuracy, efficiency, and clarity in each step of the model's development and analysis.

3.1.2 Dataset

The vehicle routing problem, being essentially spatial, makes it mandatory to precise geographical coordinates of cities or destinations. An array, facilitated by *numpy*, was carefully structured to store these coordinates. Each datum in this array represents a city, denoted by its X (longitude) and Y (latitude) coordinates. This methodical approach not only augments forthcoming computations but ensures an unambiguous representation of each city, reducing any potential for data redundancy or overlap.

3.1.3 Data Preprocessing

3.1.3.1 Computing the Distance Matrix Between Cities

Central to route optimisation is the intricate understanding of the proximities between distinct nodes or cities. Following the aggregation of city coordinates, the immediate progression was the computation of a comprehensive distance matrix.

To execute this, the Euclidean distance formula, supported by *numpy*'s mathematical functions, was employed:

Distance = $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

Where:

- x_1, y_1 are the coordinates of the maiden city.
- x_2, y_2 correspond to the coordinates of the ensuing city.

This resultant matrix, infused with the distances between all conceivable city combinations, is the solid foundation upon which the routing optimisation is based. The Euclidean method was chosen, given its intuitive nature, unambiguous approach, and its esteemed recognition in spatial evaluations.

3.1.3.2 Calculating Time Matrices for Drones and Trucks

Separate time matrices were computed for both drones and trucks to increase the accuracy and applicability of the model. The reason for this differentiation is because of the distinct speed capabilities of each mode of transport:

- Trucks: Travelling at a constant speed of 35 mph.
- Drones: Operating at a swifter pace of 50 mph.

Utilising the simple kinematic formula, time was calculated:

Time = Distance/Speed

Thus, for each pair of cities in the distance matrix, two corresponding time values were computed, representing the time taken by a truck and a drone, respectively. This dual-matrix approach provides a comprehensive framework, allowing for a more intricate and nuanced optimisation process considering both vehicles.

3.2 Heuristics

Solving the Vehicle Routing Problem (VRP), particularly for extensive instances, demands significant computational resources. The Two-opt Heuristic (Manthey & Veenstra, 2013) to optimize the VRP effectively, requires an initial tour as a starting point. Recognizing this criterion, this methodology employs the Nearest Neighbour algorithm (Josiah, Ojo et al., 2013). This algorithm is chosen for its efficiency in quickly generating a feasible initial tour. Once this foundation is established, the Two-opt Heuristic steps in to further refine and enhance the solution, ensuring a more optimized route.

3.2.1 Generation of Initial Tour using Nearest Neighbour Algorithm

The Nearest Neighbour (NN) algorithm offers a straightforward yet effective way of generating an initial tour by sequentially selecting the *nearest* unvisited city based on Euclidean distance. The following are the primary steps involved in its implementation:

Initialization: The list of cities, provided as *city_list*, is considered. The algorithm begins with a chosen starting city, and the set of unvisited cities is populated with all available cities.

Iterative Selection: While there are still unvisited cities left:

- The current city is removed from the set of unvisited cities.
- The nearest city is identified using the *euclidean_distance* function. Specifically, it calculates the Euclidean distance between the current city and every unvisited city, selecting the one with the minimum distance.

- The nearest city is then appended to the ongoing tour, and the process iterates until all cities have been visited.

Completion of Tour: Finally, to close the loop of the tour, the starting city is added to the end.

The primary rationale behind the Nearest Neighbour algorithm is its simplicity and speed in providing a feasible solution. It capitalizes on local proximity, assuming that visiting nearby cities can often yield a reasonably efficient route, making it an excellent choice for generating initial solutions.

3.2.2 Two-opt Heuristic.

Building upon the initial solution provided by the NN algorithm, the Two-opt Heuristic serves to enhance the quality of the route. The heuristic operates by iteratively reversing sub-tours within the given route, aiming to identify and rectify any sub-optimal segments.

Algorithmic Framework:

Initialization: Begin with a starting tour, preferably from an algorithm such as the Nearest Neighbour.

Iterative Optimization: Employ a loop mechanism to perpetually search for and implement tour improvements until no further improvements can be identified.

Pseudo-code for the Two-opt Heuristic (Saturn Cloud, 2023):

```
Function two_opt_heuristic
INPUT: Number of cities (n), Distance matrix (dist_matrix), Initial tour (initial_tour)
OUTPUT: Optimized tour (best_tour)
START:
1. ASSIGN initial_tour TO best_tour
2. SET improved TO True
3. WHILE improved IS True DO:
   a. SET improved TO False
   b. FOR i FROM 1 TO n-2 DO:
      FOR k FROM i+1 TO n-1 DO:
         i. FORM a segment by REVERSING the section from i TO k IN best_tour
         ii. CREATE new_tour by REPLACING the section from i TO k IN best_tour WITH the reversed segment
         iii. IF total_distance_of (new_tour, dist_matrix) IS LESS THAN total_distance_of (best_tour, dist_matrix)
THEN:
            - ASSIGN new_tour TO best_tour
            - SET improved TO True
4. RETURN best_tour
END
```

The *initial_tour* is initially set as the *best_tour*. The algorithm employs a cyclic mechanism, continuously seeking tour refinements. This iterative process remains active as long as improvements in the tour length are identified. During each cycle, the algorithm meticulously examines every possible pair of cities, represented as (*i, k*). For each of these city pairs, it assesses the impact of reversing the segment of the tour between them. Should this reversal yield a shorter overall tour, the *best_tour* is promptly updated. This iterative refinement proceeds until the algorithm can no longer find any further enhancements to the tour.

3.3 Integrating Drones into the Tour

This section focuses on seamlessly incorporating drone operations into the established two-opt tour. The methodology involves addressing prohibited regions for drone operations, identifying suitable cities for drone activity, and making subsequent adjustments to the tour.

3.3.1 Taking user input for regions where drones are prohibited.

For enhanced user interaction, the system prompts the user to define specific regions where drones aren't permitted to operate. This step is essential as it ensures that drones won't unintentionally trespass into restricted zones. Here's the approach:

- The user is continuously prompted to input the bounds of prohibited regions using x and y coordinates until they decide to stop.
- Each city within the specified prohibited region is identified and added to a list of *DRONE_PROHIBITED_CITIES*.

3.3.2 Identifying cities within the specified prohibited regions.

Post user input, the system cross-references the cities in the tour with the restricted regions. As a result, any city located within these regions is marked as prohibited for drone operations.

3.3.3 Determining optimal drone operations based on the 2-opt tour.

The algorithm aims to determine the optimal drone operations for a given tour of cities based on a 2-opt tour approach. while adhering to certain constraints regarding drone operations. The constraints are:

- A drone shouldn't wait for the truck for more than 10 minutes.
- A drone's flight endurance is capped at 30 minutes.
- A drone can't travel more than 20 miles in total for any single operation.

For each city in the tour, except the last two, the algorithm checks if it's suitable for drone operations. If it is, the algorithm then looks two places ahead till the end of the tour to find a potential end city. Between this start and end city, the algorithm searches for a drone city (an intermediate city) where the drone can be dispatched. Certain conditions are checked at this stage:

- The city should not be prohibited from drones.
- The city hasn't already been included in a prior drone operation.
- The current drone segment shouldn't overlap with any existing drone operation.
- The drone's travel conditions are met, including distance constraints, time limits, wait times, and savings.

If the savings achieved by sending the drone exceed the previous maximum savings, the best starting, ending, and drone cities, along with the maximum savings and least wait time, are recorded.

Once all possible combinations of cities have been assessed, the algorithm returns the best combination of starting, ending, and drone cities that lead to the maximum savings in time.

Pseudo-code for finding drone operations:

```
FUNCTION find_drone_operations
INPUT: tour, drone_prohibited_cities, drone_covered_cities, drone_operations, dist_matrix,
drone_time_matrix, truck_time_matrix
OUTPUT: best_start, best_end, best_drone_city, max_savings, least_wait_time

START:
1. INITIALIZE max_savings, least_wait_time TO negative infinity
2. INITIALIZE best_start, best_end, best_drone_city TO None

3. FOR each city in tour EXCEPT the last two DO:
   a. ASSIGN current city TO start_city
   b. IF start_city is in drone_prohibited_cities OR drone_covered_cities THEN continue to next city

   c. FOR each subsequent city in tour STARTING from two cities after current TO end DO:
      i. ASSIGN current subsequent city TO end_city
      ii. IF end_city is in drone_prohibited_cities OR drone_covered_cities THEN continue to next subsequent city

      d. FOR each city between the current and subsequent city DO:
         i. ASSIGN current city TO drone_city
         ii. IF drone_city is in drone_prohibited_cities OR drone_covered_cities THEN continue to next city
         iii. IF drone_city segment with start_city and end_city overlaps with any existing drone operations THEN continue to the next  city

         iv. CALCULATE truck_effective_time WITHOUT considering drone_city
         v. CALCULATE drone_time for drone_city
         vi. CALCULATE drone_tour_length for drone_city
         vii. CALCULATE drone_wait_time

         e. CHECK constraints for drone:
            i. IF drone_tour_length is greater than DRONE_TRAVEL_RADIUS OR drone_time exceeds DRONE_ENDURANCE OR drone_wait_time exceeds WAIT_LIMIT OR drone_wait_time is negative THEN continue to next city
            ii. CALCULATE savings for using drones for this segment
            iii. IF savings is greater than max_savings THEN UPDATE max_savings, best_start, best_end, best_drone_city, least_wait_time with current values

4. RETURN best_start, best_end, best_drone_city, max_savings, least_wait_time
END
```

The main execution involves running this operation iteratively. After each iteration, if a suitable drone operation is identified, it's added to a list of drone operations. The loop continues until the maximum allowable number of drone operations is reached or no more suitable drone operations can be found. To ensure uniqueness:

- A set, named *drone_covered_cities*, retains a record of cities already involved in drone operations.

- Post each iteration, if an optimal drone operation is discerned, it's appended to the *drone_operations* list and the cities involved are added to the *drone_covered_cities* set.

In the end, if no suitable drone operations are identified throughout the entire process, the algorithm informs the user. Otherwise, it provides the total number of drone tours found.

3.3.4 Adjusting the 2-opt tour based on Drone Operations.

With the optimal drone routes identified, the tour undergoes further adjustments to account for these drone routes. If a drone operates between two cities, the truck might not need to pass through them directly. The goal is to synchronize the truck's route with the drones to maximize efficiency.

The method starts by extracting all cities that have drone operations. A fresh *adjusted tour* is then initialized, beginning with the first city from the provided tour. The function then iterates over each subsequent city in the tour. If the city is not part of the drone operations, it is included in the adjusted tour. This way, cities serviced by drones are excluded from the truck's route, preventing unnecessary stops, and thereby achieving greater efficiency.

With this method, the modified 2-opt tour ensures that the truck only visits cities that cannot be reached by drone deliveries, simplifying and optimising the entire process.

3.4 Preprocessing the Data for Performing Matheuristics

Preprocessing the data is essential before applying the Matheuristics optimisation technique. This makes certain that the required input is properly segmented and formatted, to feed into the mathematical computations. The following are the steps:

3.4.1 User defines specific regions for optimization.

The user needs to specify the region they are interested in because the matheuristic technique is location-specific. To designate the optimisation region, the user gives x and y coordinates.

3.4.2 Identification of cities in the specified region.

The user-provided coordinates are used to identify the cities that are located there. This involves going through the full city list, filtering out the cities, and saving the cities in the *selected_cities* list that fall inside the specified x and y coordinate boundaries.

3.4.3 Extraction of sub-tours from the selected region from 2-opt's output.

From the post-processing of the 2-opt algorithm's output, the continuous sub-tours that contain the cities identified within the region of interest are extracted. This involves traversing the entire 2-opt tour sequence. As the traversal progresses, each city encountered that matches the list of *selected_cities* (from the desired region) is appended to a *current_subtour* However, if a city outside this selection appears in sequence, it signals the end of the *current_subtour*, prompting its storage and the initiation of a new one. Repeating this process iteratively produces a curated list of sub-tours, each exclusively composed of cities from the designated region.

3.4.4 Generation of edge lists from the remaining sub-tours.

After extracting the required sub-tours, the remaining part of the 2-opt's output is considered, which does not include the identified cities. This segment, referred to as *remaining_segments*, is necessary to preserve the integrity of the entire tour. This segment is transformed into a list of edges, which is the desired format for feeding into the matheuristic algorithm.

The selected sub-tours are initially organized by their sequential appearance within the primary 2-opt tour. Following this arrangement, segments devoid of cities from the *selected_cities* are then discerned and classified as *remaining_segments*. For each of these identified segments, an iterative process translates the segment into a collection of edges, with each edge being represented in the [*i*, *j*] format. All these transformed lists of edges are then collated and stored within *all_edge_lists*.

3.5 Matheuristics using MILP Formulation.

3.5.1 Objective Function

The primary objective of the model is to minimize the total distance travelled by both the truck and the drone. This is captured by the summation of distances between each pair of cities *i* and *j*, multiplied by the respective decision variables x_{ij}(for the truck) and y_{ij} (for the drone).

Minimize: $\sum_{i=1}^{n}\sum_{j=1}^{n} dist_{ij}.\left(x_{ij}+y_{ij}\right)$, Where is $dist_{ij}$ the distance between city *i* and city *j*.

3.5.2 Decision Variables

Decision Variables:

t_i : {1 if city *i* is visited by the truck, 0 otherwise}

d_i : {1 if city *i* is visited by the drone, 0 otherwise}

x_{ij}: {1 if the truck travels from city *i* to city *j*, 0 otherwise}

y_{ij}: {1 if the drone travels from city *i* to city *j*, 0 otherwise}

u_i : Position of city *i* in the truck's tour

The binary decision variable t_i indicates whether city *i* is visited by the truck or not. A value of 1 suggests the city is visited by the truck, whereas 0 means it's not. Analogous to t_i, the binary decision variable d_i defines the drone's operation. If the drone visits city *i*, d_i stands at 1; otherwise, it remains 0. The variables x_{ij} determine the truck's trajectory between cities. If the truck moves from city *i* to city *j*, x_{ij} equals 1. If no such movement occurs, it stands at 0. The binary decision y_{ij} variable serving a purpose similar to x_{ij}, this variable is solely for drone operations. A value of 1 implies the drone's trajectory from city *i* to city *j*, and 0 otherwise. u_i is an auxiliary variable used to represent the position of city *i* in the truck's tour, which aids in eliminating subtour formations.

3.5.3 Constraints

Sets:

n = Number of cities

Constraints:

$$x_{ij} = 1 \ \forall\ (i,j) \in \textit{given edge list.} \qquad (1)$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = t_i\ ,\forall i \in n \qquad (2)$$

$\sum_{j=1,j\neq i}^{n} x_{ji} = t_{i=t}, \forall i \in n$ (3)

$x_{ij} + y_{ij} \leq 1, \forall \{i,j\} \in n, i \neq j$ (4)

$x_{ji} + y_{ij} \leq 1, \forall \{i,j\} \in n, i \neq j$ (5)

$x_{ij} + y_{ji} \leq 1, \forall \{i,j\} \in n, i \neq j$ (6)

$x_{ji} + y_{ji} \leq 1, \forall \{i,j\} \in n, i \neq j$ (7)

$t_{drone\_city} = 0 \; \forall \; drone\_city.$ (8)

$y_{start\_city\ drone\_city} = 1, \forall \; operation \in \; drone\_operations$ (9)

$y_{drone\_city\ end\_city} = 1, \forall \; operation \in \; drone\_operations$ (10)

$d_{drone\_city} = 1, \forall \; operation \in \; drone\_operations$ (11)

$\sum_{i=[start\_city,drone\_city,end\_city]} d_i = 3, \forall \; operation \in \; drone\_operations$ (12)

$u_1 + (n-1)x_{ij} \leq u_j + (n-2), \forall \; i,j \in 1 \leq i,j \leq n, i \neq j$ (13)

$$\sum_{i=1,i\neq drone\_city}^{n} t_i = n - len(drone\_operations) \quad (14)$$

Constraint (1) ensures that the given edges in the edge lists are fixed, i.e., the truck is mandated to traverse these edges. In constraints (2) and (3), the model ensures that for each city *i*, the number of incoming and outgoing edges for the truck is equivalent to whether the city is visited by the truck (t_i). Constraint (4) to (7) guarantees that the paths of the truck and drone don't overlap. This means that for any two cities *i* and *j*, only one of them (either the truck or the drone) can travel between them. Constraint (8) ensures that cities designated to be visited by drones are strictly off-limits for the truck. Constraints (9) to (12) lays down specifics about the drone's operations. It mandates the drone's start and end points for every operation and further dictates that only three distinct cities (*start_city*, *drone_city*, and *end_city*) are covered in a drone operation.

The Miller-Tucker-Zemlin (MTZ) (Miller et al., 1960). constraints (13) are integrated to prevent subtours within the truck's route. This ensures the truck's tour is one continuous loop without any isolated cycles. It also ensures that the solution forms a Hamiltonian cycle (Rahman & Kaykobad, 2005), thus visiting all cities once and only once starting from city 0.

Finally, constraint (14) makes certain that all cities, excluding those explicitly designated for drone visits, are visited by the truck. This upholds the principle that each city is visited either by the truck or the drone but not left out.

3.6 Solution Retrieval and Tour Construction

After obtaining the results from the MILP model, the next vital step is to translate this solution into tangible truck and drone tours that can be easily visualized and executed. This section provides a detailed procedure followed to retrieve and construct these tours based on the solution matrix.

3.6.1 Extracting truck and drone tours based on the model

Using the solution matrices obtained from the MILP, one can delineate distinct tours for both the truck and the drone. The function *get_tour_from_solution_matrix*, which, depending on the solution matrix and starting city fed into it, extracts the respective tour for either the truck or the drone.

3.6.1.1 Truck Tour Construction

To initiate the truck tour, the starting city, *start_city*, is set. Using the solution matrix x (representing the truck's routes), the *get_tour_from_solution_matrix* function is called to procure the truck's route. Beginning at *start_city*, the tour moves in a sequence, picking the next city from the matrix until the route loops back to the starting point or no subsequent cities are evident. This sequence of cities, which comprises the truck's route, is captured in the *truck_tour_updated* list, which concludes by revisiting the *start_city*, thereby completing the circuit.

1.6.1.2. Drone Tour Construction

The drone tour extraction utilizes a similar iterative approach, but it is extracted from the solution matrix y, which pertains to the drone's path.

Looping over the unique set of cities in the matrix y, the function ensures not to revisit cities already included in a drone tour. The function *get_tour_from_solution_matrix* is once again leveraged to fetch a tour for the drone, initiating from a given city *i*. If the resulting tour consists of more than a single city, it's appended to the all*_drone_tours* list, which accumulates all the drone tours.

3.6.2 Properly constructing the tours

To ensure the tours are logically sound and executable:

a. The algorithm begins with the designated *start_city* and endeavours to find the subsequent city in the sequence.
b. Once a city is identified and confirmed to be part of the tour, it is appended to the ongoing tour list.
c. The search for the next potential city in the tour is based on two critical criteria:
d. The city should have a direct connection with the current city in the solution matrix.
e. It shouldn't have been visited already in the tour.
f. As the list of potential cities is perused, any city that is connected to the current city in the solution matrix is promptly chosen as the next city in the tour.
g. The iteration ceases if the algorithm encounters the starting city again or if no plausible subsequent city is found.
h. At the end of one cycle, the algorithm resets its parameters, specifically the *next_city* variable, and proceeds to the upcoming iteration.

This process ensures that the tours derived from the MILP solution are continuous, logical, and actionable, making the logistics process streamlined and efficient.

3.7 Visualization

It is essential to visualise these results for easier understanding and practical takeaways after obtaining the answer from the MILP model and building the appropriate tours. This section explores the methods used to present these results.

3.7.1 Using Plotly to Visualize the Tours

Plotly is renowned for its ability to craft interactive plots, rendering it apt for this task. Here's a breakdown of how Plotly is used to bring the tours to life:

Cities Representation: All cities in the tour are depicted as black solid dots using a scatter plot, excluding the drone-prohibited cities.

Drone-Prohibited Cities: These cities are portrayed with a red cross to make them distinguishable and signify that drones cannot navigate through them.

Truck Tour: The truck's path between the cities is visualized using black lines, which help in understanding the sequence and connection of cities in the truck's tour.

Drone Tour: Distinctively, the drone's tour paths are depicted using dashed black lines. If there are multiple drone tours, they are all drawn on the same plot. The first drone tour gets its legend entry, while the subsequent ones are clubbed under the same legend category to prevent redundancy.

City Labels: Every city is labelled with its corresponding number, set slightly apart from the city dot to maintain clarity, and provide easy identification.

The above plotting elements combined yield an interactive visualization where the user can hover over parts of the plot to get more details.

3.7.2 Using Tabulate to Display the Results in Table Format

To present an overview of results, tabulate is employed. It's an excellent tool for rendering lists of dictionaries or other tabular data types into a neat table. The code prints tables with headers, distinctly showcasing results from the 'Two-Opt Heuristics' and 'Matheuristics' methods.

3.7.3 Using Matplotlib and Seaborn to Visualize and Compare the Results

While Plotly is instrumental in visualizing the tours, Matplotlib and Seaborn offer broader capabilities to compare results across different methods:

Bar Graphs: Using Seaborn's barplot, bar graphs are crafted to depict a comparison based on the total distances and durations by different methods. This helps in discerning which method is more efficient with respect to distance or time.

Styling and Appearance: The code also employs seaborn styles and custom colour palettes to enhance the appearance of these plots, ensuring they are aesthetically pleasing while delivering insights.

The amalgamation of Plotly, Tabulate, Seaborn, and Matplotlib guarantees a holistic representation of the results.

4. Experimental Setup

The experimental setup plays an important role in the elucidation of a model's efficacy and robustness. Through meticulous planning, the aim is to establish a comprehensive insight into how the model performs under varying scenarios, thereby accentuating its strengths, and identifying potential areas of improvement.

4.1 Grid Layout

Before diving into city instances, it's imperative to recognize that cities are spread out across different grid sizes to better reflect variations in spatial distributions and complexities. The grid layouts selected for the experiments are:

a. 5x5 grid for 10 cities
b. 10x10 grid for 20 cities
c. 15x15 grid for both 50 and 100 cities
d. 20x20 grid for 200 cities

4.2 City Instances

To understand how the model behaves under varying scales, experiments will be executed across different instances distinguished by the number of cities. It is also imperative to recognize that cities are spread out across different grid sizes to better reflect variations in spatial distributions and complexities The chosen instances and grid sizes for the experiments are:

a. 10 cities: Within a 5x5 grid, this serves as an elementary test, highlighting the model's foundational proficiency.
b. 20 cities: Set within a 10x10 grid, this tests the model's efficiency in handling moderately complex scenarios.
c. 50 cities: Positioned within a 15x15 grid, the complexity increases, testing the model's resilience against heightened computational needs.
d. 100 cities: Still within a 15x15 grid, the model will be pushed to optimize amidst an increasingly dense network.
e. 200 cities: With a 20x20 grid, this challenges the model's scalability and its ability to produce efficient solutions under intensive conditions.

4.3 Comparative Analysis

The primary objective of these experiments is not only to analyse how the model manages across different scales but also to compare the efficacy of vehicle routing in scenarios both with and without the inclusion of drones. The key metrics for comparison will be:

Computation time: Determining the model's efficiency in terms of how quickly it can produce optimal or near-optimal solutions.

Improvements: Quantifying any enhancements in route efficiency achieved by the integration of drones in the vehicle routing process.

Time duration and distance: Evaluating the model's capability to minimize both the time taken and the distance travelled, which are critical in real-world logistics scenarios.

4.4 Drone-Prohibited Regions Testing

Different drone-prohibited areas will be integrated into the test scenarios. This is to understand how specific regions, where drones can’t operate, impact the overall efficiency and routing outcomes of the model.

4.5 Matheuristic Testing Regions

In addition to the standard experiments, different regions will be selected for deploying and testing Matheuristics. In a specialized subset of tests:

a. Different regions will be chosen for the deployment of Matheuristics, providing insights into how geographic variances (like city density and configuration) affect model performance.
b. For a select few instances, the MILP, utilized for Matheuristics, will be presented with the entirety of the problem. This is to put beside its solutions with those arising when only a segmented portion of the problem was supplied.

In orchestrating this rigorous experimental structure, the intention is to furnish a detailed understanding of the model's functionality. Such a methodical approach not only highlights its potential but also underscores areas warranting further refinement or exploration.

5. Results and Evaluation

In the following section, a comprehensive analysis of the performance metrics for drone and truck tours across various grid sizes and prohibited regions is presented. Table 5.1 provides a detailed breakdown of the tour distances, durations, and computation times under different scenarios and optimization techniques.

Table 5.1: Comparative Analysis of Drone and Truck Tour Metrics Across Different Grid Sizes and Prohibited Regions

Instances	10	20	20	50	50	50
Grid size	5x5	10x10	10x10	15x15	15x15	15x15
Drone prohibited region(s)	0,4,0,2.5	7,10,0,3	7,10,0,4	5,9,6,10	5,9,6,10	12,14,0,2 and 0,6,13,15
Region coordinates for performing Matheuristic: min x, max x, min y, max y	0,4,0,8	6,10,0,10	0,10,0,10	0,8,0,15	8,15,8,15	0,15,0,15
Two-opt heuristics truck tour distance [before adding the drone]	18.52	36.37	36.37	91.38	91.38	91.38
Two-opt heuristics truck tour duration [before adding the drone]	0.53	1.04	1.04	2.61	2.61	2.61
Two-opt heuristics truck tour distance [after adding the drone]	17.7	35.75	35.75	86.54	86.54	83.44
Two-opt heuristics truck tour duration [after adding the drone]	0.51	1.02	1.02	2.47	2.47	2.38
Sum of two-opt heuristics truck and drone tours distance	23.86	56.69	56.69	129.43	129.43	128.62
Sum of two-opt heuristics truck and drone tours duration	0.63	1.44	1.44	3.33	3.33	3.29
Matheuristics truck tour distance	17.7	35.75	35.75	86.31	86.54	79.89
Matheuristics truck tour duration	0.51	1.02	1.02	2.47	2.47	2.28
Sum of Matheuristics truck and drone tours distance	23.86	56.69	56.69	129.2	129.43	125.07
Sum of Matheuristics truck and drone tours duration	0.63	1.44	1.44	3.32	3.33	3.19
Drone tour computation time	0.0s	0.0s	0.0s	1.0s	1.0s	0.7s
Two-opt heuristics computation time	0.0s	0.4s	0.4s	0.8s	0.8s	0.8s
Matheuristics computation time	0.1s	0.2s	0.7s	4.4s	1.0s	3.6s

Instances	100	100	100	200	200	200	200

Grid size	15x15	15x15	15x15	20x20	20x20	20x20	20x20
Drone prohibited region(s)	0,6,12,15	0,6,12,15	0,6,12,15	0,6,0,6	0,6,0,6	0,6,0,6	0,6,0,6 and 16,20,17,20
Region coordinates for performing Matheuristic: min x, max x, min y, max y	8,15,0,15	6,15,0,6	0,8,0,12	12,20,0,20	13,20,13,20	0,13,14,20	0,13,0,10
Two-opt heuristics truck tour distance [before adding the drone]	125.98	125.98	125.98	235.36	235.36	235.36	235.36
Two-opt heuristics truck tour duration [before adding the drone]	3.6	3.6	3.6	6.72	6.72	6.72	6.72
Two-opt heuristics truck tour distance [after adding the drone]	121.37	121.37	121.37	222.91	222.91	222.91	223.53
Two-opt heuristics truck tour duration [after adding the drone]	3.47	3.47	3.47	6.37	6.37	6.37	6.39
Sum of two-opt heuristics truck and drone tours distance	189.52	189.52	189.52	357.2	357.2	357.2	354.12
Sum of two-opt heuristics truck and drone tours duration	4.83	4.83	4.83	9.05	9.05	9.05	9
Matheuristics truck tour distance	116.48	119.3	117.37	214.64	221.82	222.91	212.97
Matheuristics truck tour duration	3.33	3.41	3.35	6.13	6.34	6.37	6.08
Sum of Matheuristics truck and drone tours distance	184.56	187.38	185.45	348.92	356.11	357.2	343.56
Sum of Matheuristics truck and drone tours duration	4.69	4.77	4.72	8.82	9.02	9.05	8.7
Drone tour computation time	7.9s	7.9s	7.9s	3m 34.8s	3m 34.8s	3m 34.8s	6m 37.4s
Two-opt heuristics computation time	1.6s	1.6s	1.6s	9.1s	9.1s	9.1s	22.0s
Matheuristics computation time	2.4s	1.3s	1.8s	16.5s	5.7s	5.2s	25.1s

5.1 Efficacy of Drone Integration

The primary goal of this section is to understand the impact of incorporating drones into the logistical framework. This is achieved by observing the percentage reduction in both truck tour distances and durations upon the inclusion of drones.

For instance, in a 5x5 grid with 10 cities, the truck's tour distance decreased from 18.52 units to 17.7 units upon drone incorporation, marking a decrease of approximately 4.4%. Similarly, the duration of the truck's tour decreased from 0.53 units to 0.51 units, implying a reduction of nearly 3.8%. The trend continued in larger grid sizes and greater city counts. In a 10x10 grid with 20 cities, the distance was reduced by approximately 1.7%, and the duration showed a decrease of around 1.9%.

It was further observed that as the number of cities increased, the reduction percentages also exhibited fluctuations. For a 15x15 grid with 50 cities, it can be observed in Figures 5.1 and 5.2, that reductions in distance ranged between 5% to nearly 7.5%. Correspondingly, the duration also showcased similar reductions that can be seen in Figure 5.2.

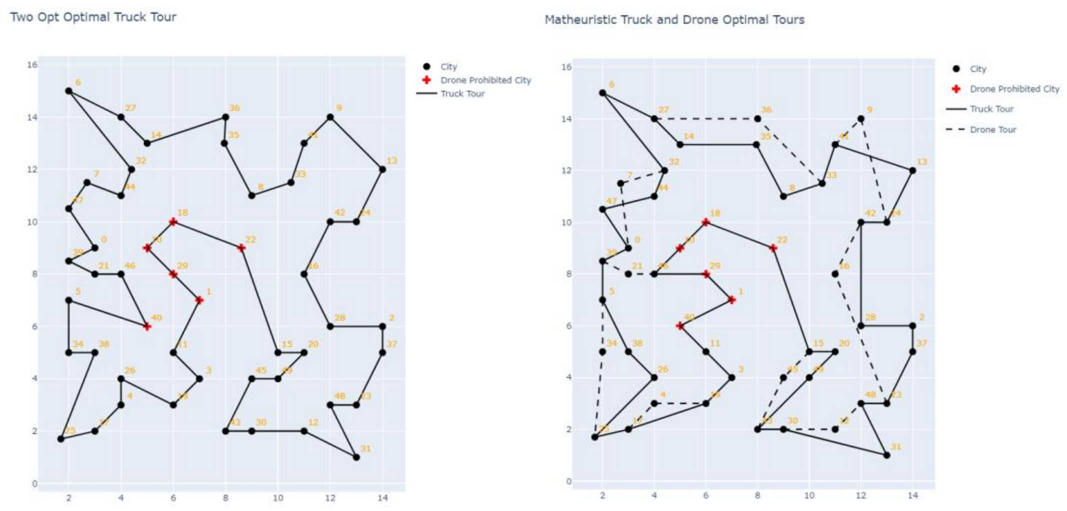


Figure 5.1 Depicting the Truck and Drone Tours for 50 cities instance with drone prohibited region "5,9,610", graph on left is two-opt tour before adding drones and the one on right is after adding drones and performing Matheuristic on region "0,8,0,15".

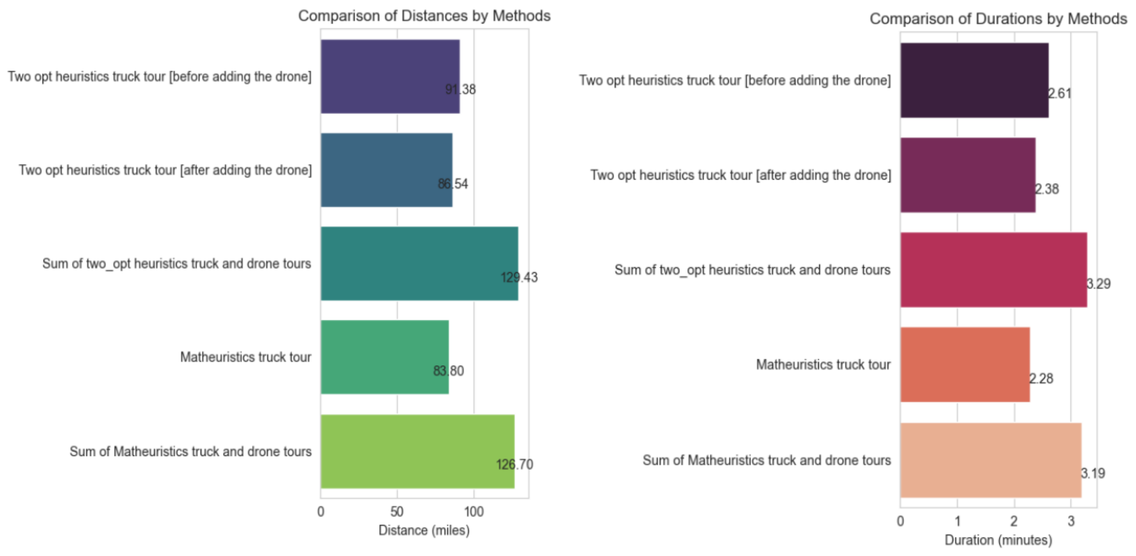


Figure 5.2 Depicting the Truck and Drone Tour Distances and Durations comparison across methods for 50 cities instance with drone prohibited region "5,9,610" and Matheuristic region "0,8,0,15".

Moreover, in the largest grid size of 20x20 with 200 cities, the efficacy of drone integration remained evident. The distance reduction varied from 5.3% to almost 9.3%, depending on drone-prohibited regions. Duration reductions also ranged from 5.2% to 9.5% in similar instances.

In conclusion, the introduction of drones in the delivery system consistently led to reductions in both truck tour distances and durations across various grid sizes and city counts. This is indicative proof of the operational efficacy of drone integration, suggesting the immense potential of drones in optimizing traditional delivery systems.

5.2 Improvements from Matheuristic MILP Model Optimization

After the utilization of two-opt heuristics to derive a near-optimal tour, further optimization was endeavoured through the application of the matheuristic MILP model. This additional layer of optimization is intended to fine-tune the results from the two-opt heuristic method.

Taking a closer look at the data, in the 5x5 grid with 10 cities, the truck tour distances post-drone integration remained constant at 17.7 units, whether using the two-opt heuristic or matheuristic MILP. However, as the grid size and city count increased, some discrepancies began to emerge. For instance, in a 10x10 grid with 20 cities, both methodologies returned identical results in one scenario, but when the matheuristic was applied to the entire problem, the computation time increased, although without any improvement in tour distances or durations.

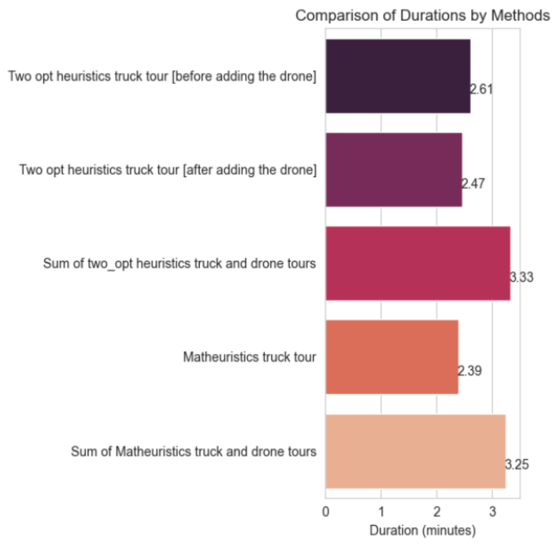


Figure 5.3 Depicting the Truck and Drone Tour Duration comparisons for 50 cities instance with drone prohibited region "12,14,0,2 and 0,6,13,15" and the whole problem passed for Matheuristics.

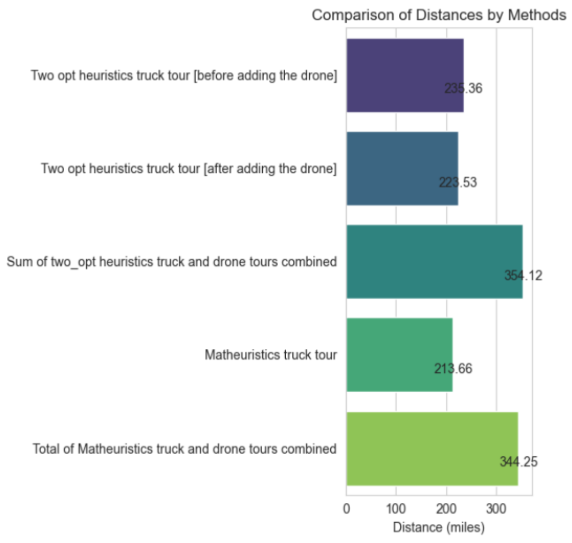


Figure 5.4 Depicting the Truck and Drone Tour Distances comparisons for 200 cities instance with drone prohibited regions "0,6,0,6 and 16,20,17,20", and region "0,13,0,10" passed for Matheuristics.

The potential of the matheuristic MILP model became more pronounced in larger scenarios. In a 15x15 grid with 50 cities, where two drone-prohibited regions were identified, the distance after using the two-opt heuristic was 83.44 units. But, with the incorporation of matheuristic across the entire grid, there was an improvement leading to a reduced distance of 79.89 units—a tangible reduction of 4.26%. Moreover, the duration underwent a decrease from 2.38 to 2.28 units, a 4.2% improvement seen in Figure 5.3.

The pattern continued in larger instances, such as in the 20x20 grid with 200 cities. In one scenario with two drone-prohibited regions, the distance reduced from 223.53 units (two-opt heuristic) to 212.97 units (matheuristic MILP) — a significant 4.46% improvement. Furthermore, in terms of duration, a decrease from 6.39 to 6.08 units represented a 4.85% reduction. Refer to Figure 5.4.

Across these instances, the application of the Matheuristic MILP model on the two-opt heuristic solutions exhibits varying levels of improvement. In smaller grids with fewer instances, the optimizations are marginal or negligible. However, as the complexity of the instance increases, either in terms of grid size or drone-prohibited zones, the benefits of the Matheuristic model become more pronounced.

These results suggest that for more intricate logistical challenges, investing in further optimization through the Matheuristic model can yield valuable efficiencies in both tour distance and duration.

5.3 Analysis of Convergence and Robustness of Two-Opt Heuristics and Matheuristics

Numerous variables can be used to assess an algorithm's effectiveness, but convergence time and resilience are crucial. Two-opt heuristics and Matheuristics were compared in various domains as part of the study.

5.3.1 Convergence Time for the Two-Opt Heuristics and Matheuristics

When examining the efficiency of algorithms, a primary consideration often revolves around the time taken for convergence. In the analysis, the two-opt heuristics stood out due to its relative speed in deriving near-optimal tours. Matheuristics, given its more complex nature, naturally consumed a lengthier duration to converge. However, it’s important to factor in the potential benefits derived from each method. By initially utilizing the two-opt heuristic to establish a foundation with a near-optimal route and subsequently directing specific regions to Matheuristics for further optimization, the process manages to capitalize on the strengths of both algorithms. This complex approach potentially reduces the convergence time for the matheuristic method, ensuring a swift yet efficient optimization. Therefore, the combined use of two-opt heuristics followed by region-specific Matheuristics can offer time-saving benefits, without compromising the quality of the solution.

5.3.2 Robustness Check by Introducing Variance in Input Data

a. **Variations in Grid Sizes**: The flexibility of an algorithm is best evaluated by its response to changes in the input dataset. Different grid sizes were inevitably going to produce distinct results. As grid size increased, the complexity of the problem often grew, potentially lengthening the convergence time for Matheuristics. While the two-opt heuristic consistently provided a swift near-optimal solution irrespective of grid size, Matheuristics took longer but often achieved more refined solutions, especially in larger grids.
b. **Drone-Prohibited City Variations**: Introducing different or multiple regions as drone-prohibited cities indeed brought about variations in the outcome, Refer to Figures 5.2 and 5.3. In some scenarios, the exclusion of certain cities from the drone route led to longer truck tour distances but shorter durations. This points towards the adaptive nature of both algorithms, wherein they modify routes based on constraints, prioritizing either distance or time depending on the prohibition parameters.
c. **Different Regions Passed to Matheuristics**: The size and location of regions provided to Matheuristics for further optimization also showcased varied results. Smaller regions often took less time to converge and yielded marginal improvements, while larger regions, though taking longer, sometimes led to significant enhancements in the solution.

In conclusion, both the heuristic and matheuristic models display adaptability across different scenarios and input changes. Their combined usage, especially when calibrated for specific problem instances, can lead to optimal and efficient solutions. However, caution must be exercised to ensure that the increased computational time aligns with tangible improvements, making the process both effective and efficient.

5.4 Operational Consistency

When reviewing the intricacies of drone-truck operations in the context of logistics, it's essential to analyse the consistency and reliability of the model's performance, especially in varying operational scenarios.

5.4.1 Variability in Results for the Same Instance but Different Drone-Prohibited Regions or Matheuristic Regions

Impact of Drone-Prohibited Regions on Tour Durations and Distances:

Observing the data, the introduction of drone-prohibited regions has led to the adaptation of truck tours. While intuitively, more prohibited regions might seem to elongate the truck's tour, the model exhibits resilience in most scenarios, maintaining a relatively consistent distance and duration. For example, in the 50-city instance with a grid size of 15x15, even with multiple drone-prohibited regions, the Two-opt heuristic truck tour distance after drone integration and the Matheuristics truck tour distance remains relatively stable. This indicates that the model can reroute efficiently, accommodating these drone limitations.

Efficiency of Matheuristic Regions:

Matheuristics, by design, are implemented to fine-tune solutions and extract further optimization. Regardless of the drone-prohibited regions in play, the Matheuristic approach consistently fine-tunes the tour's overall distance and duration. Interestingly, this optimization doesn't drastically change the computational time in many cases, showcasing the efficiency of the approach. For instance, when observing the 100-city instance with a grid size of 15x15, despite variations in Matheuristics regions, there's a consistent reduction in overall distance and duration when comparing the Matheuristic results to the initial two-opt heuristic results.

Operational Consistency across Varying Instances:

The model maintains a remarkable consistency across different city instances and grid sizes. Larger instances, such as the 200 cities with a grid size of 20x20, reflect the intricacies of managing more drone-prohibited zones and larger Matheuristic regions. Even with multiple drone-prohibited areas, the truck and drone tours' overall distance and duration are modified only marginally, underscoring the model's ability to consistently adapt to operational changes.

5.5 Matheuristics Efficacy

Matheuristics blends the rigidity of mathematical modelling with the flexibility of heuristics to generate efficient solutions. Its performance can be dependent on factors like the number of cities and grid size. Let's delve deeper into the provided results to understand its behaviour better.

5.5.1 Comparison Between Whole and Part Application of Matheuristics

a. **For a 10-city, 5x5 grid**: Matheuristics was applied to a specific region and the whole problem. In both cases, the truck and drone tour distances and durations remained identical at 23.86 units and 0.63 minutes respectively.

b. **For 20-city, 10x10 grid**: Both the specific region and whole problem applications resulted in a consistent truck and drone tour distance and duration of 56.69 units and 1.44 minutes respectively.

c. **For 50-city, 15x15 grid**: The most varied results were noticed. The Matheuristics applied to the entire grid reduced the truck and drone tour distance to 125.07 units from 128.62 units, showing a significant improvement. However, when applied to smaller regions, results varied slightly, suggesting the sensitivity of outcomes to the chosen region. Refer to Figures 5.2 and 5.3.

d. **For 100-city, 15x15 grid**: Distinct improvements were observed as the region for Matheuristics varied. Application to a larger region ("8,15,0,15") Figures 5.5 and 5.6 showed a drop in total tour distance to 184.56 units from 189.52 units. In contrast, smaller regions ("6,15,0,6" and "0,8,0,12") had tour distances of 187.38 units and 185.45 units respectively, indicating a non-linear relationship between region size and outcome.

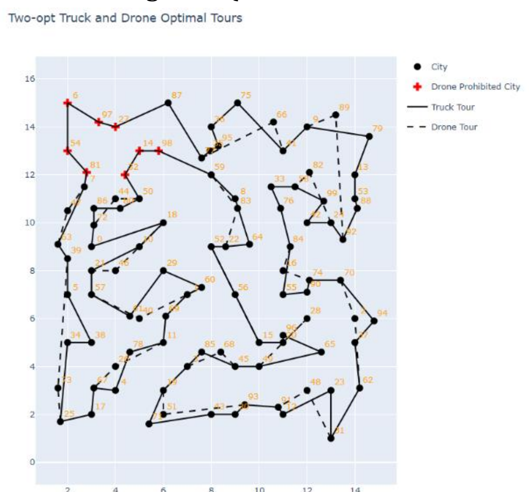


Figure 5.5 Depicting the two-opts Truck and Drone Tours for 100 cities instance with drone prohibited region "0,6,12,15".

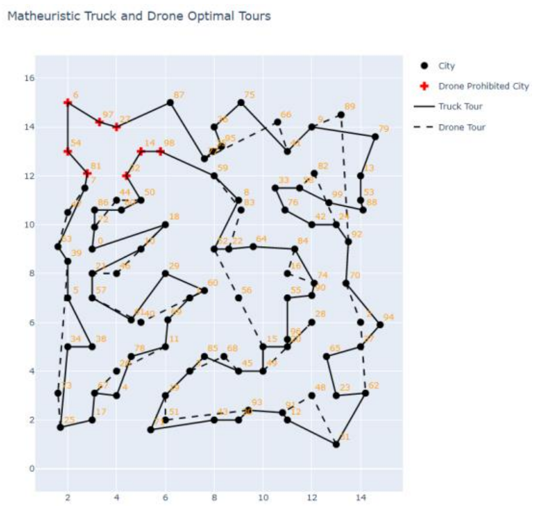


Figure 5.6 Depicting the Truck and Drone Tours for 100 cities instance with drone prohibited region "0,6,12,15" and Matheuristic region "8,15,0,15".

e. **For 200-city, 20x20 grid**: A diverse set of results was observed based on the selected region. The whole problem application was not always optimal. For instance, a specific region application ("12,20,0,20") improved the total tour distance from 357.2 units to 348.92 units, but when further restricted to "13,20,13,20", the distance was marginally reduced to 356.11 units. This showcases the balance between generalization and localization in optimization.

While applying Matheuristics to the entire problem does yield good results, it's not always superior. Depending on grid size and city density, localized applications can sometimes outperform whole-grid optimizations.

5.5.2 Relationship Between Matheuristics Region Size and Improvement Achieved

The relationship isn't linear. In certain grids like the 15x15 grid for 100 cities, larger regions resulted in better outcomes. However, for the 20x20 grid with 200 cities, certain smaller regions produced more optimal results than the entire grid.

The drone-prohibited regions play a crucial role in this relationship. In grids where multiple drone-prohibited regions exist, like in the 50-city, 15x15 scenario, the choice of Matheuristics region can vary results significantly.

Computation time can also be indicative of the complexity and efficiency of the applied method. While smaller grids had almost negligible computation times, the 200-city grid had noticeable time differences, with certain regions being more time-efficient than others.

5.6 Evaluating Drone Operations

5.6.1 Instance and Grid Relationship with Computation Time

It's evident from the data that as the number of instances (cities) increases, so does the computation time for determining drone operations. For instance, with 10 cities in a 5x5 grid, the computation time is mere milliseconds. However, when the instances increase to 200 cities in a

20x20 grid, the computation time can reach upwards of 3 minutes, indicating a direct correlation between the number of cities and the time required, The density of the city network can be compared using Figures 5.1 and 5.7 and Table.

5.6.2 *Impact of Drone-Prohibited Regions on Computation Time*

Analysing the drone-prohibited regions shows mixed results concerning computation time. In some cases, having multiple drone-prohibited regions leads to longer computation times, as seen with the 200 cities in a 20x20 grid where two prohibited areas resulted in a computation time of 6m 37.4s. However, in other scenarios, like another 200 cities test case in the same grid size but with different prohibited regions, Figure 5.7, the time taken was consistently around 3m 34.8s. This suggests that while the presence of drone-prohibited regions can increase computation time, the exact layout and distribution of these regions play a significant role.

5.6.3 *Comparative Analysis*

When comparing the drone computation time with the two-opt heuristics computation time and the Matheuristics computation time, a varied range is observed. While in smaller grids and instances the drone computation time is minimal, in larger scenarios, it becomes substantial. However, it often remains less than or comparable to the Matheuristics computation time, suggesting that while drones add a computational layer, the complexity of Matheuristics often overshadows it.

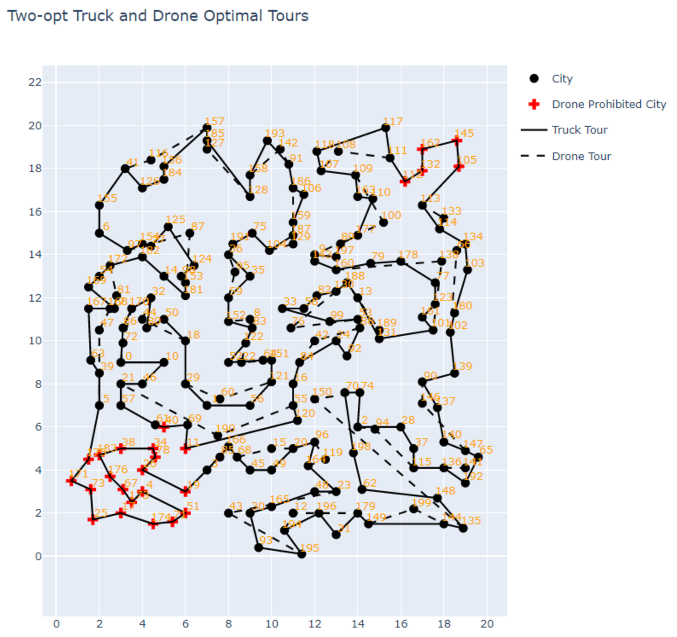


Figure 5.7 Depicting the Truck and Drone Tours produced by two-opt heuristics after adding drones in 200 cities instance with drone prohibited regions "0,6,0,6 and 16,20,17,20", and region "0,13,0,10" passed for Matheuristics.

In conclusion, while drone operations introduce additional computational challenges, particularly in larger scenarios with multiple prohibited regions, their integration provides a more efficient solution when combined with matheuristic methods.

6 Conclusion

The integration of drones into the logistical framework presents a great opportunity to revolutionize traditional delivery systems. The consistent reduction in truck tour distances and durations underscores the potential of this approach. However, to extract the maximum benefit, efficient optimization strategies like two-opt heuristics and Matheuristics are crucial.

A key success in this research was the effective use of the MTZ formulation. It not only simplified the process of avoiding sub-tour elimination but also seamlessly integrated with the Matheuristics approach, proving its value in this framework.

The study indicates that as the complexity of the delivery problem increases, the benefits of advanced optimization strategies like Matheuristics become more pronounced. This insight is particularly valuable for logistics operations dealing with large delivery networks. The combined use of two-opt heuristics and Matheuristics seems to offer a balanced solution, while two-opt heuristics quickly provides a near-optimal solution and Matheuristics fine-tuning it, especially in larger and more complex scenarios.

A remarkable feature of the integrated model is its adaptability to operational changes. Whether there's a change in the drone-prohibited regions or the overall size of the delivery network, the model efficiently recalibrates to offer consistent performance. The flexibility of drone operations can be observed when changing grid sizes and introducing drone-prohibited regions. The algorithm modifies the drone paths based on the given constraints, ensuring that it achieves the best possible results under the circumstances.

Acknowledging that drones may not be universally viable across all terrains and regions, provisions are made to input areas where drone operations might be inadvisable or prohibited. This mechanism ensures the plotted drone routes sidestep areas that could be potentially unsafe or unsuitable for them.

While the research has primarily focused on a theoretical framework, the practical implications are immense. Delivery businesses can leverage these insights to streamline their operations, reduce operational costs, and enhance customer satisfaction.

In conclusion, the integration of drones into delivery systems, when combined with efficient optimization strategies, holds the promise of a more efficient, cost-effective, and sustainable future for logistics and delivery operations. The study provides a comprehensive blueprint that can guide the evolution of next-generation delivery networks.

7 Future Scope of Work

The drone delivery system presented offers a promising approach to optimizing delivery routes and efficiency. In envisioning a broader horizon, there's an intriguing prospect of extending this model to allow drones to cater to multiple cities within a single tour. By aligning this with the drone's weight capacity, one can explore ways for drones to deliver multiple packages within the constraints of a single journey, potentially redefining delivery efficiency.

Furthermore, with the potential increase in scale of operations, integrating multiple trucks along with drones into the model is a natural progression. This addition will require the model to ensure that a city is not served twice, guaranteeing optimal resource allocation and time efficiency. The challenge here lies in synchronizing the operations of multiple vehicles manned or unmanned without overlap, but the resultant increase in efficiency and potential scalability will be invaluable.

Introducing time windows for deliveries represents another significant enhancement. Integrating delivery time constraints can cater to customer preferences and enhance user experience. By incorporating these time window constraints into our existing MILP model, deliveries can be scheduled during the most desirable and convenient periods for customers. This adjustment might necessitate exploring advanced MILP techniques or alternative constraint programming methods to ensure efficient adaptation and resolution of the model.

Considering energy management is another exciting frontier. By analysing factors like wind resistance and energy consumption patterns, there's room to augment the model's delivery efficiency further. Additionally, real-time mapping and routing is a promising avenue to consider. In situations of unforeseen challenges, such as sudden bad weather affecting drone operations or unexpected road closures for trucks, a real-time mapping system can be invaluable. By integrating real-time data, the model can instantly adjust routes and delivery mechanisms. This not only ensures safety and adherence to regulatory standards but also significantly boosts customer satisfaction by maintaining delivery punctuality.

With these avenues in mind, it's evident that while the current model provides a robust framework, the domain of drone deliveries has numerous facets left to explore, ensuring the continuous evolution and optimization of such systems.

References

Basel, John, and Thomas R Willemain. “Random Tours in the Traveling Salesman Problem: Analysis and Application.” Computational optimization and applications 20.2 (2001): 211–211. Web.

Bo Lan, “Traveling salesman problem with time windows and drones (TSPTWD)”, (2020, https://dr.lib.iastate.edu/server/api/core/bitstreams/5f8fdefe-c538-4b03-b34f-51281fcb390b/content

Brodowsky, Ulrich A, and Stefan Hougardy. “The Approximation Ratio of the 2-Opt Heuristic for the Euclidean Traveling Salesman Problem.” arXiv.org (2021): n. pag. Print.

Conrad, Ryan & Figliozzi, Miguel. (2011). “The Recharging Vehicle Routing Problem”. Proc. of the 61st Annual IIE Conference".

Dantzig, G. B., and J. H. Ramser. “The Truck Dispatching Problem.” Management Science, vol. 6, no. 1, 1959, pp. 80–91. JSTOR, http://www.jstor.org/stable/2627477. Accessed 15 May 2023.

Dell’Amico, Mauro, Roberto Montemanni, and Stefano Novellani. “Matheuristic Algorithms for the Parallel Drone Scheduling Traveling Salesman Problem.” Annals of operations research 289.2 (2020): 211–226. Web.

Di Puglia Pugliese, Luigi & Guerriero, Francesca. (2017). Last-Mile Deliveries by Using Drones and Classical Vehicles. 557-565. 10.1007/978-3-319-67308-0_56.

Doyuran, Tamer. “Enhancements of Clarke-Wright savings heuristics for the capacitated vehicle routing problem.” (2008).

Erdogan, Sevgi, and Elise Miller-Hooks. “A Green Vehicle Routing Problem.” Transportation research. Part E, Logistics and transportation review 48.1 (2012): 100–114. Web.

Glover, Fred. “Future Paths for Integer Programming and Links to Artificial Intelligence.” Computers & operations research, https://doi.org/10.1016/0305-0548(86)90048-1.

Gulczynski, Damon, Bruce Golden, and Edward Wasil. “The Multi-Depot Split Delivery Vehicle Routing Problem: An Integer Programming-Based Heuristic, New Test Problems, and Computational Results.” Computers & industrial engineering 61.3 (2011): 794–804. Web.

Gunawan, Aldy et al. “Vehicle Routing Problem with Forward and Reverse Cross-Docking: Formulation and Matheuristic Approach.” 2021 IEEE 17TH INTERNATIONAL CONFERENCE ON AUTOMATION SCIENCE AND ENGINEERING (CASE). Vol. 2021. NEW YORK: IEEE, 2021. 1467–1472. Web.

Gurobi Optimization (2022) *Gurobi Optimizer Reference Manual*, *Gurobi Optimization*. Available at: https://www.gurobi.com/documentation/ (Accessed: 29 June 2023).

Gutekunst, Samuel C., and David P. Williamson. “Subtour Elimination Constraints Imply a Matrix-Tree Theorem SDP Constraint for the TSP.” Operations research letters 48.3 (2020): 245–248. Web.

Ham, Andy M. “Integrated Scheduling of m-Truck, m-Drone, and m-Depot Constrained by Time-Window, Drop-Pickup, and m-Visit Using Constraint Programming.” Transportation research. Part C, Emerging technologies 91 (2018): 1–14. Web.

Josiah, Ojo et al. “IMPLEMENTATION OF HEURISTICS FOR SOLVING TRAVELLING SALESMAN PROBLEM USING NEAREST NEIGHBOUR AND NEAREST INSERTION APPROACHES.” (2013).

Khoufi, Ines, Anis Laouiti, and Cedric Adjih. “A Survey of Recent Extended Variants of the Traveling Salesman and Vehicle Routing Problems for Unmanned Aerial Vehicles.” Drones (Basel) 3.3 (2019): 1–30. Web.

Kuo, R.J. et al. “Vehicle Routing Problem with Drones Considering Time Windows.” Expert systems with applications 191 (2022): 116264. Web.

Kytöjoki, Jari et al. “An Efficient Variable Neighborhood Search Heuristic for Very Large Scale Vehicle Routing Problems.” Computers & operations research 34.9 (2007): 2743–2757. Web.

Laporte, Gilbert. “What You Should Know About the Vehicle Routing Problem.” Naval research logistics 54.8 (2007): 811–819. Web.

Lenstra, Jan & Kan, A. (2006). “Complexity of vehicle routing and scheduling problems. Networks.”, https://doi.org/10.1002/net.3230110211.

Lin, C.K.Y. “A Vehicle Routing Problem with Pickup and Delivery Time Windows, and Coordination of Transportable Resources.” Computers & operations research 38.11 (2011): 1596–1609. Web.

Manthey, B. and Veenstra, R. (2013) ‘Smoothed analysis of the 2-opt heuristic for the TSP: Polynomial bounds for gaussian noise’, *Algorithms and Computation*, pp. 579–589. doi:10.1007/978-3-642-45030-3_54.

Martin Simensen & Geir Hasle & Magnus Stålhane, 2022. "Combining hybrid genetic search with ruin-and-recreate for solving the capacitated vehicle routing problem," Journal of Heuristics, Springer, vol. 28(5), pages 653-697, December. 10.1007/s10732-022-09500-

Mbiadou Saleu, Raïssa G. et al. “An Iterative Two-step Heuristic for the Parallel Drone Scheduling Traveling Salesman Problem.” Networks 72.4 (2018): 459–474. Web.

Mbiadou Saleu, Raïssa G. et al. “The Parallel Drone Scheduling Problem with Multiple Drones and Vehicles.” European journal of operational research 300.2 (2022): 571–589. Web.

Miller, C, A Tucker, and R Zemlin. “Integer Programming Formulation of Traveling Salesman Problems.” Journal of the ACM 7.4, (1960): 326–329. Web.

Mitchell, J.E. ‘Branch and cut’, *Wiley Encyclopedia of Operations Research and Management Science* [Preprint]. (2011) doi:10.1002/9780470400531.eorms0117.

Montemanni, Roberto, and Mauro Dell'Amico. “Solving the Parallel Drone Scheduling Traveling Salesman Problem via Constraint Programming.” Algorithms 16.1 (2023): 40. Web.

Murray, Chase C., and Amanda G. Chu. “The Flying Sidekick Traveling Salesman Problem: Optimization of Drone-Assisted Parcel Delivery.” Transportation research. Part C, Emerging technologies, 54 (2015): 86–109. Web.

Nikolaos A. Kyriakakis, Themistoklis Stamadianos, Magdalene Marinaki, Yannis Marinakis, The electric vehicle routing problem with drones: An energy minimization approach for aerial deliveries, Cleaner Logistics and Supply Chain, Volume 4, 2022, 100041, ISSN 2772-3909, https://doi.org/10.1016/j.clscn.2022.100041.

Rahman, M. Sohel, and M. Kaykobad. “On Hamiltonian Cycles and Hamiltonian Paths.” Information processing letters 94.1 (2005): 37–41. Web.

Sacramento, David, David Pisinger, and Stefan Ropke. “An Adaptive Large Neighborhood Search Metaheuristic for the Vehicle Routing Problem with Drones.” Transportation research. Part C, Emerging technologies 102 (2019): 289–315. Web.

Saturn Cloud (2023) *2-opt algorithm: Solving the travelling salesman problem in Python*, *Saturn Cloud Blog*. Available at: https://saturncloud.io/blog/2opt-algorithm-solving-the-travelling-salesman-problem-in-python/ (Accessed: 24 July 2023).

Vercesi, Eleonora et al. “On the Generation of Metric TSP Instances with a Large Integrality Gap by Branch-and-Cut.” Mathematical programming computation 15.2 (2023): 389–416. Web.

Appendix A: Supplementary Materials I

Included in the supplementary materials of this document is a compressed file named **COMP4026_MScProjects_Dissertation_20493897_Ashwini_Doke.zip**. This ZIP file encapsulates all the necessary code and files for the project's prototype.

For a detailed understanding of the structure, installation, and execution of the project, please refer to the **README.md** file within the **COMP4026_MScProjects_Dissertation_20493897_Ashwini_Doke.zip** archive.

The README provides comprehensive guidelines, from establishing prerequisites to executing the main prototype.