UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO

Jorge Luiz Franco

A graph-based approach to Last Mile Delivery Drones

São Carlos

2024

Jorge Luiz Franco

A graph-based approach to Last Mile Delivery Drones

Monograph presented to the Undergraduate Program in Computer Science at the Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, as part of the requirements for obtaining a bachelor's degree.

Advisor: Prof. Filipe Alves Neto Verri, Ph.D.

São Carlos

2024

Franco, Jorge Luiz
S856m A graph-based approach to Last Mile Delivery Drones / Jorge Luiz Franco ; orientador Filipe Alves Neto Verri. – São Carlos, 2024.
43 p. : il. (algumas color.) ; 30 cm.

1. Optimization. 2. Last Mile Delivery. 3. Graphs I. Filipe Alves Neto Verri. II. Universidade de São Paulo. III. Instituto de Ciências Matemáticas e de Computação. IV. Last Mile Delivery Drones: A graph-based approach.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my family for helping me through this journey, specially my father, João Luiz Franco, who always has encouraged my studies. Second, I express my gratitude to my advisor, Filipe Alves Neto Verri, for guiding me this semester and introducing me to this problem. Third, I would like to thank professor Vitor Venceslau Curtis, since this work would not be possible without his help. Last but not least, I express my gratitude to my fiance, Isis, for all the patience and caring during this time, my friends for the support, and God who never abandoned me.

"Without education, we are in a horrible and deadly danger of taking educated people seriously."

G.K. Chesterton

ABSTRACT

FRANCO, J. L. **A graph-based approach to Last Mile Delivery Drones**. 2024. 43p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

The growing demand for efficient last-mile delivery solutions, driven by the rise of e-commerce and urbanization, necessitates innovative approaches to manage the complexities of urban logistics. This study investigates the use of drones for last-mile delivery through a graph-based approach, focusing on the Multi-Agent Pathfinding (MAPF) problem, which involves routing multiple drones to deliver packages efficiently. Most algorithms proposed for the Last Mile Delivery Drones (LMDD) problem tend to overlook the inherent complexities of the MAPF problem. This research adopts a graph-based representation of the delivery area, transforming the problem into a network flow optimization. The proposed graph-based heuristic method is evaluated against a purely Mixed Integer Linear Programming (MILP)-based solution. Experimental results demonstrate that the heuristic approach not only enhances computational efficiency but also maintains high solution quality. Specifically, the heuristic outperforms MILP in terms of runtime while achieving comparable accuracy in path optimization. Additionally, a hybrid implementation combining heuristic and MILP is proposed. The exact MILP model has exponential complexity time as the number of drones is increased, which is expected since the MAPF paradigm is NP-Complete. The complexity of the heuristic is bounded by $\mathcal{O}(N^3 K \max(\log N, \log K))$, where K is the number of drones and N is the dimension, if the grid is square. The study highlights the potential of centralized control systems for managing a fleet of delivery drones. The findings from extensive simulations indicate that the graph-based heuristic effectively balances computational efficiency and operational reliability, making it a viable solution for real-world last-mile delivery applications. This research contributes to the broader field of drone logistics by offering a scalable and robust method for optimizing drone delivery paths, thereby supporting the integration of drones into commercial delivery systems.

Keywords: Optimization. Graphs. Last Mile Delivery Drones.

RESUMO

FRANCO, J. L. **A graph-based approach to Last Mile Delivery Drones**. 2024. 43p. Monografia (Trabalho de Conclusão de Curso) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2024.

A crescente demanda por soluções eficientes de entrega de última milha, impulsionada pelo crescimento do comércio eletrônico e urbanização, exige abordagens inovadoras para gerenciar as complexidades da logística urbana. Este estudo investiga o uso de drones para entrega de última milha por meio de uma abordagem baseada em grafos, focando no problema de Multi-Agent Pathfinding (MAPF), que envolve o roteamento de múltiplos drones para entregar pacotes de forma eficiente. A maioria dos algoritmos propostos para o problema de Drones de Entrega de Última Milha (LMDD, em inglês) tendem a ignorar as complexidades inerentes do problema MAPF. Esta pesquisa adota uma representação baseada em grafos da área de entrega, transformando o problema em uma otimização de fluxo de rede. O método heurístico baseado em grafos proposto é avaliado em comparação com uma solução puramente baseada em Programação Linear Inteira Mista (MILP, em inglês). Os resultados experimentais demonstram que a abordagem heurística não apenas melhora a eficiência computacional, mas também mantém alta qualidade nas soluções. Especificamente, a heurística supera a MILP em termos de tempo de execução, enquanto alcança uma precisão comparável na otimização de caminhos. Além disso, é proposta uma implementação híbrida combinando heurística e MILP. O modelo MILP exato tem complexidade exponencial conforme o número de drones aumenta, o que é esperado, uma vez que o paradigma MAPF é NP-Completo. A complexidade da heurística é limitada por $\mathcal{O}(N^3 K \max(\log N, \log K))$, onde K é o número de drones e N é a dimensão, se a grade for quadrada. O estudo destaca o potencial de sistemas de controle centralizados para gerenciar uma frota de drones de entrega. Os resultados de simulações extensivas indicam que a heurística baseada em grafos equilibra a eficiência computacional e a confiabilidade operacional de forma eficaz, tornando-a uma solução viável para aplicações de entrega de última milha do mundo real. Esta pesquisa contribui para o campo mais amplo da logística de drones oferecendo um método escalável e robusto para otimizar os caminhos de entrega de drones, apoiando assim a integração de drones em sistemas de entrega comerciais. Este trabalho também sugere avanços com agentes inteligentes usando Aprendizado por Reforço (RL, em inglês) e Redes Neurais de Grafos (GNNs, em inglês) para melhorar o paradigma descentralizado, o qual nós provamos ser inferior à nossa metodologia centralizada, uma vez que utiliza uma otimização local em contraste com nossa otimização global.

Palavras-chave: Otimização. Grafos. Drones de Entrega de Última Milha.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

LMDD	Last Mile Delivery Drones
MAPF	Multi-Agent Pathfinding
BFS	Breadth First Search
MILP	Mixed Integer Linear Programming
MOEA	Multiobjective Evolutionary Algorithm
UAV	Unmanned aerial vehicle
NSGA-I	Non-dominated Sorting Genetic Algorithm
PLS	Pareto Local Search
MIP	Mixed Integer Programming
TSP	Traveling Salesman Problem
TS	Tabu Seach
GVNS	General Variable Neighborhood Search
ALNS	Adaptive Large Neighborhood Search
SA	Simulated Annealing
UTM	Unmanned Aircraft System Traffic Management
FAA	Federal Aviation Administration
BVLOS	Beyond Visual Line of Sight
NASA	National Aeronautics and Space Administration
RL	Reinforcement Learning
GNN	Graph Neural Network

CONTENTS

1 INTRODUCTION

Efficient last-mile delivery logistics is vital for accurate and timely delivery of goods, which influences customer satisfaction and overall business success (BOYSEN; FEDTKE; SCHWERDFEGER, 2021). The use of drones, also called Unmanned Aerial Vehicles (UAVs), in Last-Mile Delivery improves efficiency by overcoming traffic constraints, reducing delivery times, and lowering operational costs. For this reason, the literature on Delivery Drones increased significantly in the last few years, as analyzed in Dukkanci, Campbell and Kara (2023).

The literature related to Last-Mile Delivery Drones is highly heterogeneous, encompassing a diverse array of techniques and problem formulations. This diversity spans from studies involving both drones and trucks in delivery scenarios, linear integer modeling for logistic problem-solving, applications of fuzzy logic to address uncertainties, multi-objective optimization to consider multiple criteria simultaneously, to exclusive investigations focusing solely on drones without the presence of other vehicles, including decentralized models.

In Freitas, Penna and Toffolo (2023) it is studied the Truck-Drone Delivery Problem, which can be addressed as a variation of TSP (Traveling Salesman Problem) for two agents, using a Mixed Integer Programming (MIP) formulation and a heuristic based on a TSP solver and two metaheuristics: General Variable Neighborhood Search (GVNS) and Tabu Search (TS). At the same time, Moshref-Javadi, Hemmati and Winkenbach (2020) addresses a near problem, but with multiple drones, using Mixed Integer Linear Programming (MILP) and a heuristic based on Adaptive Large Neighborhood Search (ALNS) and Simulated Annealing (SA). A similar MILP modeling was used in Rinaldi *et al.* (2023) and two hybrid genetic algorithms were evaluated.

When addressing multiple criteria simultaneously in collaborative Truck-Drone Delivery, researchers have employed multi-objective optimization techniques. Luo *et al.* (2022) consider flexible time windows and diverse objectives, including minimizing the routing costs of drones and trucks and maximizing customer satisfaction. They improved NSGA-II(Non-dominated Sorting Genetic Algorithm) and used a posterior heuristic for Pareto Local Search(PLS). Complementing this, Zhang *et al.* (2022) introduces a novel multi-objective optimization model for drone delivery, optimizing economic and environmental goals concurrently. The model addresses dynamic drone flight endurance based on loading rates, using an extended NSGA-II algorithm that proves to be effective in generating high-quality solutions, marking advancements in this field.

In another branch, works such as Nur *et al.* (2020) introduce a systematic multi-

criterion, multipersonnel decision-making approach called interval-valued inferential fuzzy TOPSIS, handling fuzziness in decision-making and providing quality drone selection decisions. Another study Ahmadi, Wicaksono and Valilai (2021) considers sustainability in optimizing the vehicle routing problem with drones, utilizing sentiment analysis based on Twitter to determine customer sentiments on environmental protection. The net promoter score index calculated from the sentiment analysis is considered as a coefficient in the penalty function added to the base model, aiming to help logistics companies improve the sustainability of the supply chain. Addressing the aspect of energy efficiency in drone-based last mile delivery, another study Bruni, Khodaparasti and Perboli (2023) focuses on tactical decisions about the selection of shared fulfillment centers used as launch and recovery locations for drones, fleet size plans, and operational drone route decisions.

A new perspective in Last Mile Delivery Drones was introduced in Verri *et al.* (2020). The study addressed the last-mile delivery problem from a complex system viewpoint, where the collective performance of the drones is investigated. The delivery system incorporates a *tradable permit model* (AKAMATSU; WADA, 2017), based on a financial decentralized market-inspired approach, for airspace use, requiring drones to compete for airspace permits in a distributed manner. The simulation evaluates how different parameters, such as arrival rate and airspace dimensions, impact system behavior in terms of cost, time required for drones to acquire flight permits and airspace utilization. Although the simulation employs a simplified model with naïve agents and disregards drone flight dynamics, it captures interesting properties in the agents' collective behavior, demonstrating satisfactory system performance even under high traffic conditions. Simultaneously, Lee and Wong (2022) contributes a novel combinatorial double auction bi-objective winner determination problem for last-mile drone delivery.

As we can see, the Last Mile Delivery Drone problem is very hard and complex. In Faiçal *et al.* (2023), the authors characterize the problem through a cyber-physical lens, highlighting challenges related to the physical aspects of airspace in the context of drone operations for last-mile delivery. Their analysis delves into considerations such as air traffic management systems, focusing on the evolving landscape of airspace control. This review underscores the need for optimal usage of airspace, then ensuring the need for collision avoidance. However, collision avoidance is a problem that has not been addressed by most of the cited papers until now. Verri *et al.* (2020) addresses this systematically within the decentralized approach. Indeed, Dukkanci, Campbell and Kara (2023) highlights collision avoidance and air traffic management as a crucial aspect that needs attention in Last-mile delivery drone problems, as much of the literature tends to overlook this when modeling the problem.

With this problem of collision avoidance and efficient airspace control in mind, the main idea of our work is solve the Last Mile Delivery Drone problem using a MAPF

(Multi-Agent Path Finding) approach, thus not ignoring spatial characteristics natural from the airspace. The problem addressed in this work can easily be seen as multi-objective drone path planning for search and rescue (HAYAT *et al.*, 2020), thus the Last Mile Delivery Drone is per se a MAPF. As stated in Yu and LaValle (2013) and proved in Nebel (2020) the MAPF problem is NP-hard. Then our approach is to use elements of prioritized planning (ČáP *et al.*, 2015) and conflict-based search (SHARON *et al.*, 2015) to face the computational complexity inherited from MAPF while satisfying the constraints of the reduced problem. Although our algorithm does not guarantee global optimum, it always converges in a finite and bounded polynomial time, which is an improvement compared to the previous algorithms described (MILPs, MOEAs, Metaheuristics).

Centralized control for airspace management, particularly under the Unmanned Aircraft System Traffic Management (UTM) framework developed by the Federal Aviation Administration (FAA) and NASA, exemplifies the necessity of organized, legislative-backed airspace control (PREVOT *et al.*,). The UTM facilitates multiple drone operations beyond visual line of sight (BVLOS) by enabling cooperative interaction between drone operators and the FAA, determining and communicating real-time airspace status. While decentralized models, such as those based on blockchain technology (VERRI *et al.*, 2020), offer novel approaches, they introduce complexities in scalability, regulatory compliance, and operational efficiency that can hinder real-time airspace control. The centralized UTM system, supported by regulatory compliance and legislative backing, ensures effective airspace management, safety, and reliability, fostering the integration of UAV deliveries within existing air traffic systems. The FAA's UTM Implementation Plan (LIEB; VOLKERT, 2020) and ongoing evaluations highlight the continual refinement of this centralized approach, emphasizing its critical role in future UAV operations.

In this study, we propose a novel approach to tackle the Last Mile Delivery Drone problem by employing a Multi-Agent Path Finding (MAPF) strategy. Leveraging elements of prioritized planning and conflict-based search, our algorithm aims to address the computational complexity inherent in the MAPF problem. Unlike previous algorithms such as MILPs and MOEAs, our heuristic guarantees convergence in finite and bounded polynomial time, representing a significant advancement. Our graph-based approach, employing Breadth-First Search (BFS) on temporal grids, enhances the efficiency of our solution. Also, we propose a hybrid method, that combines the heuristic and the MILP. We further present experimental results, comparing our approach with a MILP-based solution and evaluating its performance in the hybrid approach that combines the time found by the heuristic and the MILP solution. Finally, a qualitative comparison between our centralized approach and the decentralized, proposed in Verri *et al.* (2020), is presented. The use of a graph-based heuristic offers distinctive advantages, and our experiments shed light on its efficacy in solving the complex Last Mile Delivery Drones problem.

Therefore, in what follows, we describe the main contributions of this work:

- We develop a MILP model to solve the LMDD problem exactly. This model offers a novel method to adapt the MAPF solution to the LMDD context, allowing for drone take-offs, waiting in airspace, and correct landings.
- We propose a heuristic algorithm to address the LMDD problem. The primary advantage of our heuristic lies in its graph-based structure, which can be adapted to higher dimensions and weighted sets in the LMDD context. Additionally, our heuristic is polynomially bounded with a low computational cost, representing a novel contribution to the current literature on drone delivery solutions.

This monograph is organized as follows. In Section 2 we present the problem definition and the new property. The proposed solution approach is described in Section 3, while the results of the computational experiments are presented in Section 4. Finally, in Section 5 conclusions are drawn.

2 PROBLEM DESCRIPTION

In this section details about the formulation of the Last Mile Delivery Drones (LMDD) as a MAPF in a grid and a mathematical model of the problem are presented. At the core of MAPF lies the objective of crafting collision-free paths for a set of agents, each navigating from individual starting points to, respectively, goal destinations. As delineated in Surynek (2022), tackling this issue generally involves the use of compilation, a notable computing technique. This process is defined by the conversion of an input instance from its original formalism into a different, usually well-established formalism, thereby creating a reduced problem for which an efficient solver exists. This reduction is crucial, as it essentially simplifies the original problem into a more manageable form, optimizing the problem-solving process.

The Last Mile Delivery Drones problem can be reduced as a set of tasks for each agent(drone), where each task is described by two tuples $task_i\{(x_{\text{start}_i}, y_{\text{start}_i}), (x_{\text{goal}_i}, y_{\text{goal}_i})\}$, $\forall$ drone d_i, with $x, y \in \mathbb{Z}_{\geq 0}$, describing the start position of the drone and the final objective position. In our discretization we make the grid finite, where each axis has bounds, i.e. $x \leq X, y \leq Y$, with X and Y being the bounds of the grid. Our goal is to find paths $P_{d_i}\{(x_{\text{start}}, y_{\text{start}}, t_{\text{begin}}), (x_{\text{decision}_1}, y_{\text{decision}_1}, t_{\text{begin}} + 1), (x_{\text{decision}_2}, y_{\text{decision}_2}, t_{\text{begin}} + 2) \ldots, (x_{\text{goal}}, y_{\text{goal}}, t_{\text{begin}} + |P_{d_i}|)\}$ for each drone d_i making the path length as short as possible.

Navigational constraints dictate that each drone's movement is restricted to parallel advancements along the grid axes, thus excluding the possibility of diagonal traversals to streamline pathfinding complexity. Consequently, each drone is limited to four principal movements: upward $(x, y + 1)$, downward $(x, y - 1)$, rightward $(x + 1, y)$, and leftward $(x - 1, y)$ shifts. Formally, a position within the grid (x_1, y_1) is adjacent to $(x_2, y_2) \iff |x_2 - x_1| + |y_2 - y_1| = 1$.

It is clear how close this formulation is to the MAPF problem. In fact, the only difference is now we have a new decision variable, the time that each drone starts their flight(t_{begin}), i.e, the arrival time when the drone takeoff. The analogy with MAPF enhances the understanding of the problem and leverages the literature of the LMDD problem to a new paradigm.

There are currently approximates, compilation and AI search based solutions to the MAPF problem (MA *et al.*, 2019). But there is a lack of MILP based formulations for the problem (STERN *et al.*, 2019). Branch Cut and Price (LAM *et al.*, 2019) is a MILP solution that hybridizes these approaches, using a decomposition framework.

The cited solutions to MAPF are not quite intuitive and leave a hard work for

generalization. However, in Yu and LaValle (2013) it is shown the isomorphism(equivalence) between the MAPF and multi-commodity minimum cost maximum flow problem. This work let an excellent and easy visualization of MAPF problems. In fact, they can all be seem as a Network Flow problem. That being said, we propose a network-based formulation of the problem.

The goal of our LMDD formulation is: given an available time T, minimize the sum of distances of all drones, while allowing drones to wait in cells of the grid and choose when they enter at the airspace(the grid). Actually, these two additions are what differentiates our formulation from standardly MAPFs as Lam *et al.* (2019) and Yu and LaValle (2013). Also, the proposed LMDD problem has non-weighted distances, what make the problem easier then the MAPF itself. As evidenced in 3.1.1 this formulation could be easily addressed if modeled as a graph, where our goal is to find the paths that generates the minimum flow in the network/graph.

2.1 MAPF as a multi-commodity Network Flow Problem

As shown in Yu and LaValle (2012b), the Multi-Agent Pathfinding (MAPF) problem can be transformed into a network flow problem using a time-expanded network and multi-commodity flow approach. This transformation involves several key steps and concepts, illustrated by the figures below.

2.1.1 Time-Expanded Network

The first step is to construct a time-expanded network. In this representation, each node in the original grid is duplicated for each time step, creating a layered network where each layer corresponds to a specific time instance. Edges are added to represent possible moves between nodes over time, including waiting in place. This transformation allows us to model the time dimension explicitly within the network.

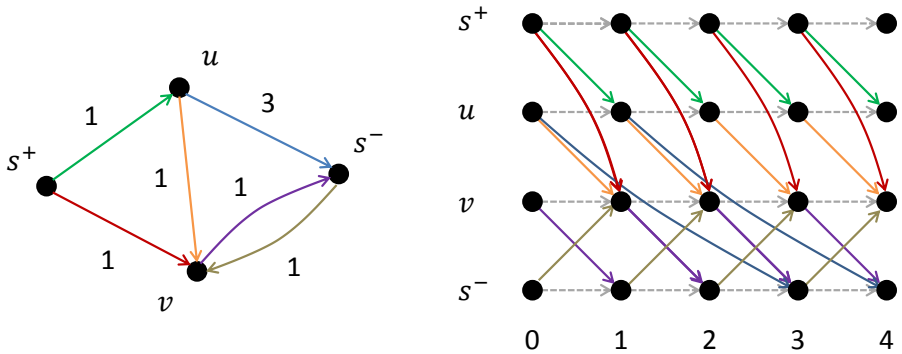

Figure 1 – A time-expanded network representation. Picture sourced from Yu and LaValle (2012a).

2.1.2 Multi-Commodity Flow Formulation

In this transformed network, the MAPF problem is equivalent to a multi-commodity flow problem. Each drone is considered a separate commodity that needs to flow from its start node to its goal node through the network. The key constraints include ensuring that each drone reaches its goal within a given time horizon T.

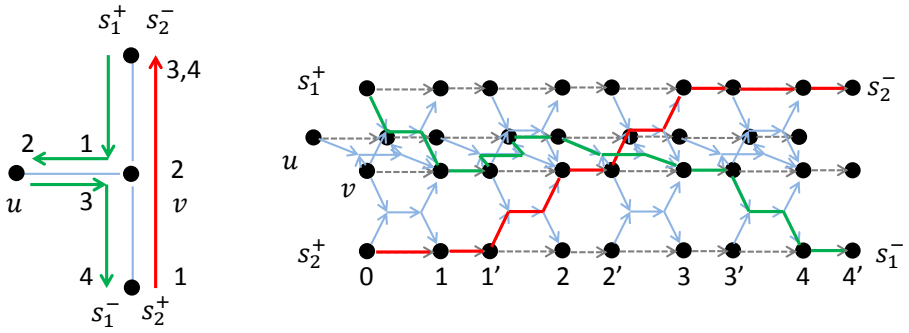

Figure 2 – Equivalence of MAPF to multi-commodity network flow (sourced from Yu and LaValle (2012a))

The equivalence between MAPF and multi-commodity network flow is fundamental because it allows the application of well-established network flow algorithms to solve MAPF problems. By leveraging this equivalence, it is possible to find paths for all drones in a computationally efficient manner, especially when combined with integer linear programming (ILP) techniques.

This transformation not only provides a clearer visualization of the problem but also enhances the ability to apply optimization techniques to find solutions. Consequently, it supports the development of more efficient algorithms for last-mile delivery drones and other applications requiring coordinated multi-agent pathfinding.

3 MATERIALS AND METHODS

In this chapter, we present in the first two sections the methodologies of both methods we used to solve the LMDD problem. Further, we present the hybrid method, that unifies both approaches.

3.1 Mathematical Model

We propose a Mixed-Integer Linear Programming (MILP) model to solve the problem. First, in Section 3.1.1, we illustrate in detail how to construct a graph representing the spatio-temporal permits shared by the drones.

3.1.1 Graph Modeling

Following the flow network problem paradigm, we model our spatio-temporal permits as a directed graph (digraph) $G = (V, A)$, where:

- V is the set of nodes representing the airspace and virtual drone locations.
- A is the set of directed arcs representing the permitted transitions between nodes.

The temporal component is omitted here as we are primarily interested in representing the problem's spatial topology.

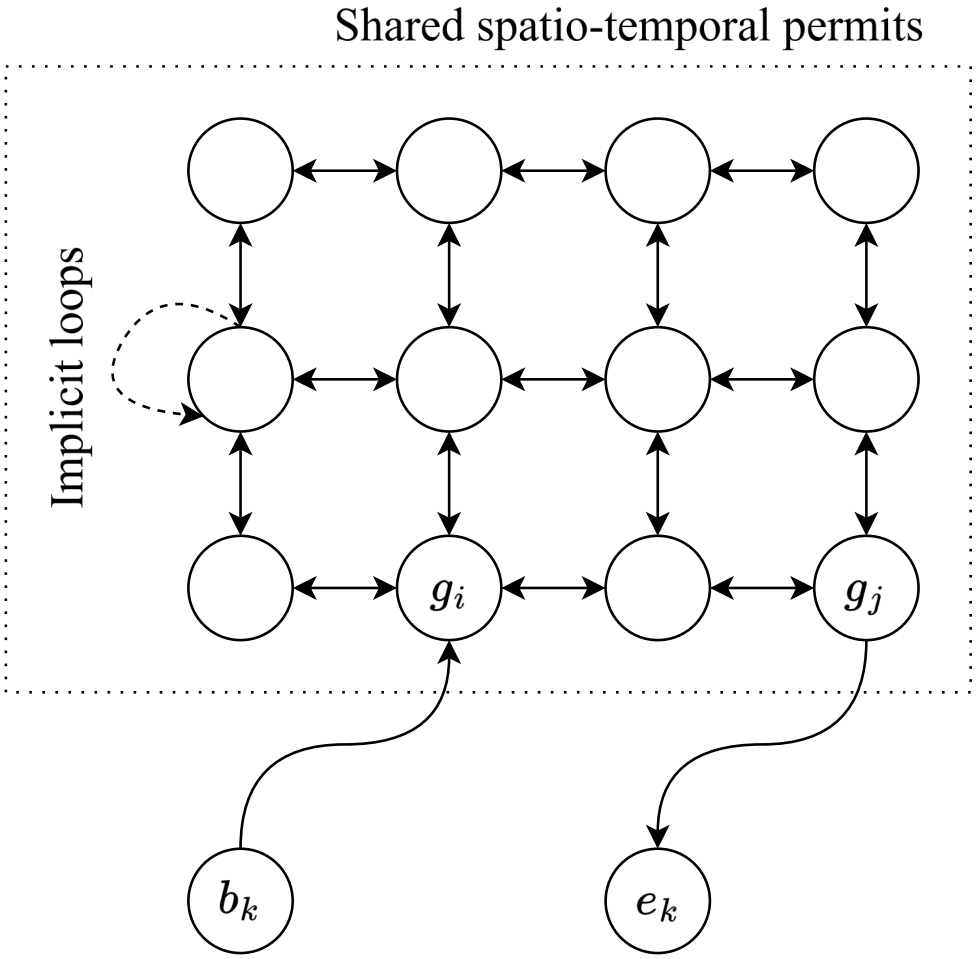

Figure 3 – Graph modelling. Source: The authors.

Figure 3 depicts a simplified 2D grid-shaped example of a shared airspace (without altitude) for illustrative purposes. The dotted rectangular area represents the physical 2D airspace.

3.1.1.1 Virtual Nodes for Drones

In addition to the shared airspace, we introduce two virtual nodes for each drone k:

- A source node b_k: This represents the starting point of drone k's mission.
- A sink node e_k: This represents the ending point of drone k's mission.

Each drone k must initiate its mission at b_k and conclude it at e_k. These virtual nodes streamline the model by circumventing the need to explicitly handle individual drone start and end positions.

Here's a breakdown of the virtual node connections:

- Drone k can only remain in b_k or exit it to the shared airspace.
- Once exited, no drone (including k itself) can re-enter b_k.
- A priori, any drone k can enter its designated sink node e_k. However, constraints will be formulated in the mathematical programming section to ensure only drone k enters its corresponding e_k.

3.1.1.2 Drone Loitering

Since we're dealing with rotary-wing vehicles, the model allows drones to remain (loiter) at a node for a certain time. To accommodate this, we include a loop arc for every node in the graph, encompassing both physical and virtual source/sink nodes. For clarity, these loops are implicitly considered in the example of Figure 3 and not explicitly depicted.

Including loops simplifies the mathematical programming model by avoiding the need to address potential exceptions.

3.1.2 Parameters

Below are listed the parameters that are constants of the problem:

- T: maximum time allowed for the mission (limits drone movement);
- b_k: initial virtual vertex representing the initial position (begin) of drone k;
- e_k: final virtual vertex representing the final position (end) of drone k;
- $\mathcal{R}$: set of drones;
- $\mathcal{G}$: digraph $(\mathcal{V}, \mathcal{A})$ representing the airspace;
- $\mathcal{V}$: set of vertices of $\mathcal{G}$ representing each cell of the airspace or virtual vertex;
- $\mathcal{B} \subset \mathcal{V}$: set of initial virtual vertices b_k;
- $\mathcal{E} \subset \mathcal{V}$: set of final virtual vertices e_k;
- $\mathcal{S}$: set $\mathcal{V} \setminus (\mathcal{B} \cup \mathcal{E})$ of vertices exclusively representing the shared airspace, i.e., eliminating virtual vertices;
- $\mathcal{A}$: set of arcs $(i, j) \in \mathcal{A}$ of $\mathcal{G}$ connecting airspace cells to each other or a virtual vertex to the airspace.

3.1.3 Indices

In the equations, we use the following iteration indices:

- k : drone $\implies k \in \mathcal{R}$;
- t : time $\implies 1 \leq t \leq T$;
- i, j, l : vertices $\implies i, j, l \in \mathcal{V}$.

3.1.4 Decision Variables

Below are listed the decision variables of the model:

- $x^k_{i,j,t} = 1 \iff$ drone k jumps from i to j at time t.

3.1.5 Objective Function

We choose an objective function that minimizes the total sum of the number of drone movements,

$$\min \sum_{k \in \mathcal{R}} \sum_{t=1}^{T} \sum_{(i,j) \in \mathcal{A}:\ j \notin (\mathcal{E} \cup \mathcal{B})} x^k_{i,j,t}, \tag{3.1}$$

counting $n-1$ jumps for each drone that performs n jumps, since the jump to the final virtual vertex cannot be counted as a vertex with cost for the drone mission. Loops at initial virtual vertices are also disregarded.

3.1.6 Constraints

The set of constraints are described in the following.

3.1.6.1 Movement Start

We require that each drone starts its mission,

$$\sum_{t=1}^{T} \sum_{j \in \mathcal{S}} x^k_{b_k,j,t} = 1, \quad \forall k \in \mathcal{R} \tag{3.2}$$

, by jumping from its virtual initial node b_k to the shared airspace.

3.1.6.2 Flow Conservation

The flow conservation is also addressed,

$$\sum_{j \in \mathcal{V}} x^k_{i,j,t-1} = \sum_{l \in \mathcal{V}} x^k_{j,l,t}, \quad \forall j \in \mathcal{V}, \forall k \in \mathcal{R}, \forall t \in \{2, \ldots, T\}\ , \tag{3.3}$$

ensuring the consistency of allowed movements defined by the graph $\mathcal{G}$

3.1.6.3 Border Condition

Initially, under the border condition at time $t = 0$, we consider there is a loop flow of drones at the virtual initial vertices. This flow is redirected into the problem by the flow conservation constraint.

$$x^k_{i,j,0} = \begin{cases} 1, & \text{if } i = b_k \wedge j = b_k, \\ 0, & \text{otherwise.} \end{cases} \quad \forall k \in \mathcal{R}, \forall (i,j) \in A \ . \tag{3.4}$$

For all other variables at time $t = 0$, we consider a flow equal to zero.

3.1.6.4 Mutual exclusion of vertex occupation

We ensure that no drones occupy the same position,

$$\sum_{k \in \mathcal{R}} \sum_{j \in \mathcal{V}} x^k_{i,j,t} \leq 1, \quad \forall j \in \mathcal{V}, \forall t \in \{1, \ldots, T\} \ , \tag{3.5}$$

that is, each vertex $j \in \mathcal{V}$ is occupied by at most one drone.

3.1.6.5 Mission Accomplishment

Every drone k needs to complete its mission,

$$\sum_{t=1}^{T} \sum_{i \in \mathcal{S}} x^k_{i,e_k,t} \geq 1, \quad \forall k \in \mathcal{R} \ , \tag{3.6}$$

i.e., reaching its final virtual vertex e_k is required.

3.2 Heuristic

In this section, we present our heuristic approach for tackling the problem, focusing on the utilization of a distance measure as a heuristic metric to organize drones in ascending order(prioritized planning) based on their start and end points. This prioritized order is then employed in an iterative Breadth-First Search (BFS) on the temporal graph. The BFS process dynamically updates the constraints of occupied positions(conflict-based search), creating a sequential refinement of the temporal graph with each newly generated minimum path. By combining heuristic sorting and iterative BFS, our approach aims to efficiently navigate the solution space, providing a balance between effective path planning and adaptability to evolving constraints in the context of drone delivery.

MAPF algorithms are recognized for their versatility with various distance metrics (WEISE; MOSTAGHIM, 2023). In our specific scenario, we chose the Euclidean Distance as our metric of choice. This decision stems from its efficacy as a tiebreaker for drones that traverse the same number of cells in the minimum path, as observed with the Manhattan Distance. Notably, drones may share the same Manhattan Distance but diverge

in Euclidean Distance, particularly when paths involve changes in direction. For instance, drones following a straight-line trajectory in the minimum path encounter fewer potential movement positions, while paths with changes in axis may introduce additional possibilities, making Euclidean Distance a valuable discriminant in such cases. This strategic use of the Euclidean Distance enhances the overall performance and adaptability of our algorithm in navigating complex drone delivery scenarios, as exemplified in 3.2.1, especially in scenarios with high traffic congestion where having more possibilities is crucial for finding local optimal solutions.

While conflict-based search has exponential complexity (GORDON; FILMUS; SALZMAN, 2021), our algorithm has polynomial complexity in function of size of the grid and number of drones. However, we cannot guarantee optimality, and this is expected since, even in the 2D grid case, MAPF where each agent has three or more possible directions of movement is also NP-hard(GEFT, 2023).

Table 1 – Notation used in the Algorithm.

Notation	Definition
$\mathcal{V}$	Set of vertices in the graph : (i, j, t)
$\mathcal{E}$	Set of edges
$\mathcal{D}$	Set of drones
$\mathcal{S}$	Set of already scheduled vertices
$\mathcal{P}_d \subseteq \mathcal{V}$	Path of drone d
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	Temporal Graph

Formally, using the notation in Table 1, we can create a temporal graph $\mathcal{G}$ that naturally represents our problem. Since the set of vertex $\mathcal{V} := (i, j, t)$ represents every possible position (i, j) at any given time t in our problem. This modeling approach streamlines the search for optimal solutions in an efficient manner. The representation allows us to capture the possibility of a drone waiting at its position at a specific time t_{wait} through the graph edges. We achieve this by defining the set of outgoing edges for a vertex $v^* = (i^*, j^*, t^*$ as $\mathcal{E}_{v^*} = \{(i^* + 1, j^*, t^* + 1), (i^*, j^* + 1, t^* + 1), (i^* - 1, j^*, t^* + 1), (i^*, j^* - 1, t^* + 1), (i^*, j^*, t^* + 1)\}$. The edge $(i^*, j^*, t^* + 1)$ means that the drone chooses a path in which it decides to wait a time step. We define our algorithm sequentially as follows:

1. **Drones Sorting**: We begin by ascending sorting the drones using the Euclidean Distance as a heuristic, given by $\sqrt{(x_{\text{begin}} - x_{\text{end}})^2 + (y_{\text{begin}} - y_{\text{end}})^2}$,

$$\mathcal{D}_{\text{sorted}} = \text{sort}(\mathcal{D}, \text{heuristic}) \ . \tag{3.7}$$

2. **Path for Each Drone**: For each drone d in the sorted order, we compute the path P_d using Breadth-First Search (BFS) on the graph $\mathcal{G}$. The BFS is performed with

the constraints imposed by the set $\mathcal{S}$,

$$\forall d \in \mathcal{D}_{\text{sorted}} : \quad \mathcal{P}_d = \text{BFS}(\mathcal{G}, d) \ . \tag{3.8}$$

3. **Constraints Update**: After determining paths for the respective drone, we update the set of already scheduled vertices $\mathcal{S}$ immediately following the execution of each BFS sequentially. This update involves incorporating the vertices covered by the path into the existing set of constraints (previously scheduled paths),

$$\mathcal{S} = \mathcal{S} \cup \bigcup_{d \in \mathcal{D}_{\text{sorted}}} \mathcal{P}_d \ . \tag{3.9}$$

Indeed, the **constraints** ensure that no two drones occupy the same vertex at the same time,

$$\forall d, d' \in \mathcal{D}, d \neq d', \forall (i, j, t) \in \mathcal{P}_d, (i, j, t) \notin \mathcal{P}_{d'} \ . \tag{3.10}$$

In the computational aspect, our implementation follows the standard implementation of BFS in 3D grids, since the time t can be simply thought as a third dimension. The main difference of common implementations is the addition of a *map*, that is a Red Black Tree, to manage the set of scheduled positions. This addition adds $\mathcal{O}(\log n)$ in the complexity of our algorithm. The pseudo algorithm of the heuristic is described in

Algorithm 1.

Algoritmo 1: Path Planning for Drones using BFS

Data: Grid dimensions N (rows) and M (columns), Number of drones K, List of drones *drones*

Result: Paths for each drone considering constraints

```
foreach drone in drones do
    Initialize data structures for BFS: bfs_queue, parent, visited
    Set flight_time of drone to -1
    while No valid path found for drone do
        Increment flight_time of drone
        Enqueue drone_begin with flight_time into bfs_queue
        while bfs_queue is not empty do
            Dequeue a position and time
            if Position is the destination of drone then
                Reconstruct and schedule path for drone
                return path
            end
            if Position is scheduled then
                Continue
            end
            for Neighbor positions do
                if Position is valid and not visited then
                    if Position is scheduled and no schedule discovered in neighbors
                     yet then
                        Enqueue current position and flight_time + 1
                        Continue
                    end
                    Mark position as visited
                    Store parent information
                    Enqueue the neighbor position into bfs_queue
                end
            end
        end
    end
end
```

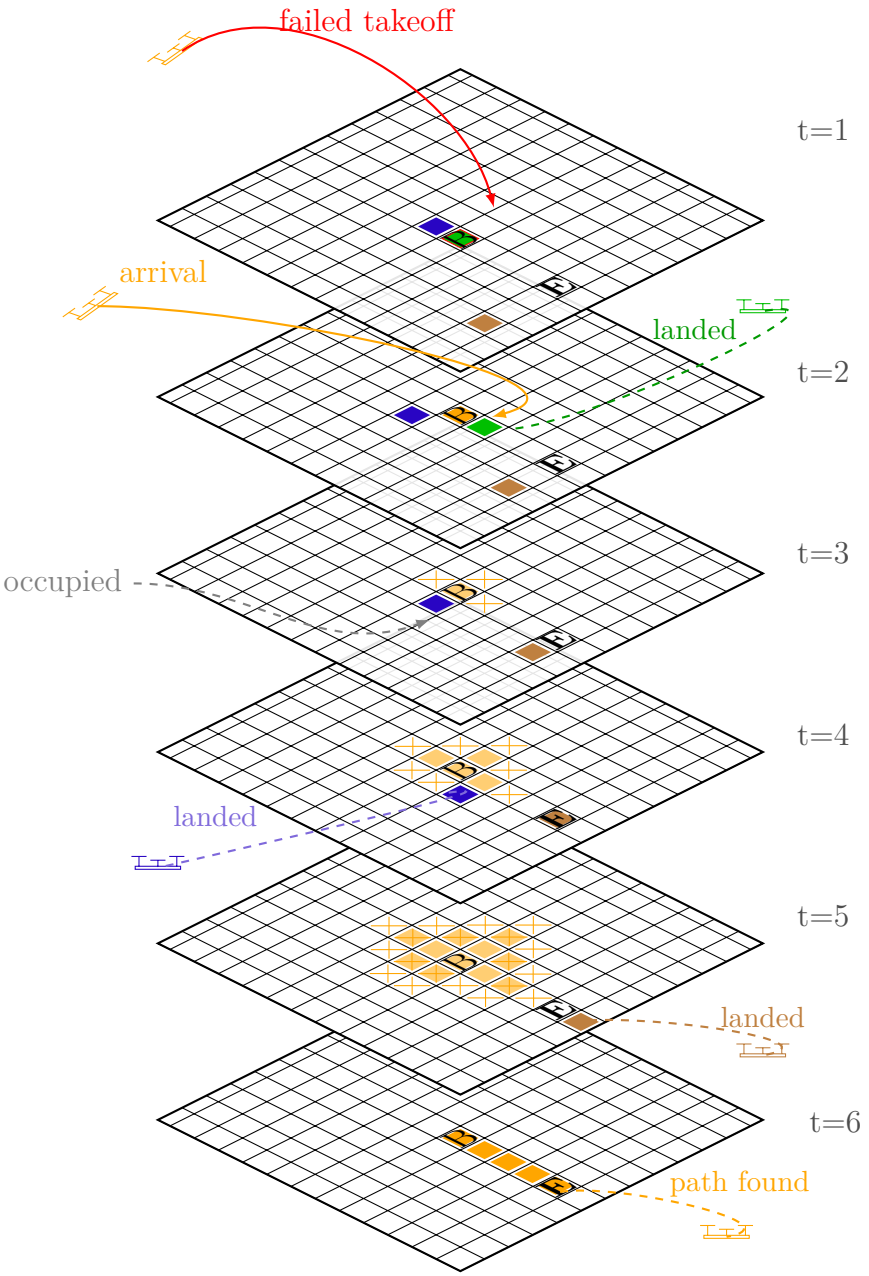

Figure 4 – Algorithm visualization. Source: The authors.

3.2.1 Example

In Figure 4, the algorithm visualization shows its ability to address corner cases, including take-off in occupied positions and collision avoidance.

The depicted scenario involves determining the route for the orange drone within a grid where blue, brown, and green drones have already established their scheduled routes. These pre-existing paths serve as constraints in the optimization problem.

Upon closer inspection of Figure 4, it is evident that both the orange and brown drones cover an equal number of positions in the grid. However, an important factor influencing the algorithm's decision-making is a heuristic chosen for prioritization. In this

case, the heuristic involves considering the shorter Euclidean distance covered by the brown drone. Consequently, during the sorting process, the algorithm prioritizes the brown drone based on this chosen heuristic, irrespective of the equality in the number of positions traversed.

3.2.2 Complexity Analysis and Boundedness

Given K the number of drones and N, M the grid sizes, the number of rows, and the number of columns, respectively. We can analyze our algorithm regarding these inputs using asymptotic computational complexity.

The complexity of our algorithm is the worst case of a BFS in the temporal graph, considering the $\mathcal{O}(\log S)$ from the scheduled structure, where $S = NMT$ is the size of the search space. It can be seen that this would be $\mathcal{O}(NMT)\mathcal{O}(\log NMT)$, where T is the maximum time a drone lands. However, T is surely bounded by $\mathcal{O}(KNM)$. In addition, it can be shown that T is bounded by $\mathcal{O}((N+M)K)$. An intuitive view of this is shown in 5, where we have three drones on a 3×4 grid with the same initial position(s_i) and final position(g_i), where $i = 1 \dots 3$. In this case, the shorter path for each drone is $n+m$,$n = 3; m = 4$, and in the worst case, we will wait for each drone to complete its path, thus $k(n+m)$ is an upper bound for our heuristic since our heuristic always chooses a shorter path than $k(n+m)$ because we do not wait for each drone to complete the path. Then the complexity is bounded by

$$\mathcal{O}((N+M)KNM\log((N+M)KNM)) = \mathcal{O}(\max(N,M)NMK\log(\max(N,M)NMK))$$

$$\approx \mathcal{O}(N^3K\log(N^3K) = \mathcal{O}(N^3K\max(\log N, \log K)), \text{ if the grid is square.}$$

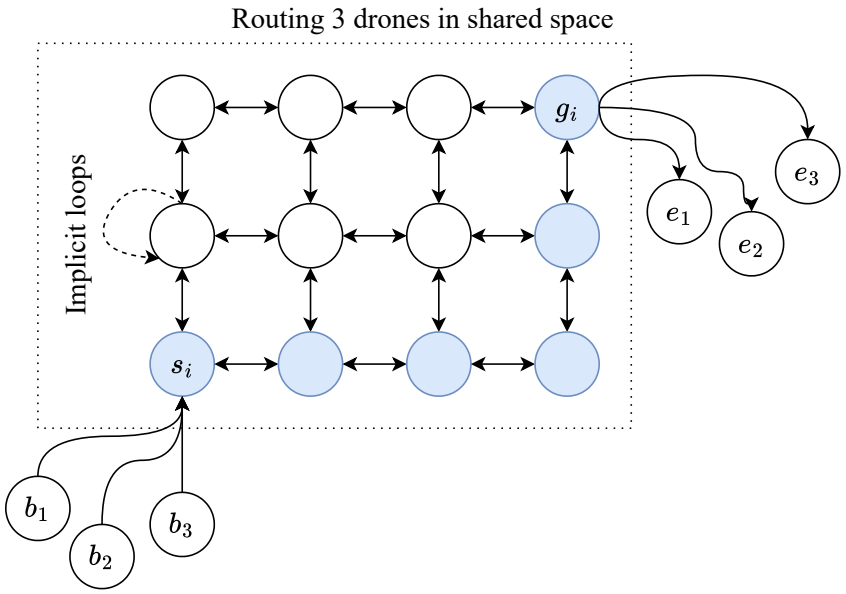

Figure 5 – Worst case path.

3.2.3 Adaptability for 3D

A notable difference from our method to others in MAPF is that our algorithm remains consistent across any grid dimension. That is, the algorithm for a 2D grid obtains solutions in the same way as for a 3D or 4D grid.

This fact becomes clear when we see that both our methods, the heuristic and the exact model, are based solely on the topology of the graph. There is no spatial dependency. When we increase the dimension to 3D, we are simply increasing the neighborhood of each node. In some sense, each vertex v in 3D space will have two additional edges in its neighborhood (up and down).

Again, using the notation in Table 1, we can create a temporal graph $\mathcal{G}$ that naturally represents our problem in three dimensional space. The set of vertices $\mathcal{V} := (i, j, k, t)$ represents every possible position (i, j, k) at any given time t in our problem. Our definition is almost the same, the difference is we add one more index k and consequently the possibility of the drone going up$(k + 1)$ and down$(k - 1)$. We achieve this by defining the set of outgoing edges for a vertex $v^* = (i^*, j^*, k^*, t^*)$ as $\mathcal{E}_{v^*} = \{(i^*+1, j^*, k^*, t^*+1), (i^*, j^*+1, k^*, t^*+1), (i^*-1, j^*, k^*, t^*+1), (i^*, j^*-1, k^*, t^*+1), (i^*, j^*, k^*+1, t^*+1), (i^*, j^*, k^*-1, t^*+1), (i^*, j^*, k^*, t^*+1)\}$.

3.3 Hybrid Methodology

In this section, we propose a hybrid approach to solve the problem. Our hybrid methodology combines the strengths of both the heuristic approach presented in Section 3.2 and the MILP model outlined in Section 3.1. By leveraging the efficiency of the heuristic to quickly generate a feasible solution and then refining it using the exact model, we aim to achieve a balance between computational speed and solution optimality.

3.3.1 Algorithm Overview

We can outline the hybrid method sequentially. The methodology proceeds through the following steps:

3.3.1.1 Heuristic Solution Generation

The first step involves employing the heuristic algorithm to generate an initial feasible solution. The heuristic algorithm determines the time horizon $T_{\text{heuristic}}$ and computes paths for drones based on their initial positions and problem parameters.

3.3.1.2 MILP Model Refinement

Once the heuristic completes its execution, we utilize the time horizon $T_{\text{heuristic}}$ along with the initial drone positions and grid size parameters as input to the MILP model.

Also, we modify the initial model variables to be the initial feasible point found by the heuristic, that is, the solver will already start the search in a local optimum point. The MILP model refines the solution within the time frame $T_{\text{heuristic}}$, ensuring optimality while maintaining computational efficiency.

3.3.2 Advantages and Considerations

The hybrid methodology offers several advantages:

- **Computational Efficiency**: Leveraging the heuristic for initial solution generation reduces computational time compared to relying solely on the MILP model. Since we would have no warm start and would need to start with a guessed T, that in general would be high.
- **Global Optimality**: Refinement using the MILP model ensures the final solution achieves global optimality.
- **Flexibility and Adaptability**: Dynamic integration of the heuristic and MILP model allows for handling diverse problem instances and adapting to varying computational resources.

The main reason, that makes the hybrid method an useful one, is that we can skip the part where we need to determine T. Since, using just the MILP model we would need to test lots of T_s to determine which one to use, for each T we would need to solve an other instance of the problem and when finding a infeasible solution return $(T + 1)$. A simple algorithm would be: `while(T--){ if(is_infeasible(T) return T+1;}`, for a high guess of T, that is, in the worst case we would solve $\mathcal{O}(T)$ instances of the mathematical model, which is expensive. When we get the T from the heuristic we guarantee the MILP solver search to be in a plausible T close to the optimal.

However, potential trade-offs include computational complexity and the need for careful validation and testing. The success of the hybrid methodology relies on the synergy between the heuristic and MILP model components. Although in this work we do not experiment the hybrid method, in Section 4.2.3 we show that using $T_{\text{heuristic}}$ as a guess T for the exact model is reasonable, since T is linear in the number of drones.

4 RESULTS

In this chapter, we describe the experiments conducted, beginning with the experimental setup and technology settings used (Section 4.1). We then compare the heuristic with the exact model using a fixed T (obtained by the heuristic) (Section 4.2), and finally, we present the phase transition in the heuristic, normalizing by individual drone paths (Section 4.2.3). This approach allows us to illustrate the advantages and limitations of our method.

All experiments were performed on a 64-bit operating system with an Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz and 16GB of RAM.

4.1 Experimental Setup and Technologies

All experiments were conducted using the following software and tools:

4.1.1 Heuristic Implementation in C++

The heuristic approach was implemented in C++, chosen for its efficiency and control over system resources. Key components include:

- **Standard Library Containers**: The heuristic utilized standard C++ library containers such as `vector` and `map` for efficient data management and manipulation.
- **Version and Compiler**: The implementation made use of C++20, and the code was compiled using the GNU GCC compiler.

4.1.2 Exact Model Implementation in Julia

The exact model was implemented in Julia, chosen for its high-performance capabilities and ease of use in mathematical and scientific computing. Key aspects include:

- **JuMP Package**: The exact model's optimization tasks were carried out using the `JuMP` package (LUBIN *et al.*, 2023), a domain-specific modeling language for mathematical optimization embedded in Julia.
- **Gurobi Solver**: Gurobi was used as the Mixed-Integer Linear Programming (MILP) solver, interfaced through Julia, to solve the optimization problems with high precision and performance.

4.1.3 Experimentation and Analysis

The comparison, experiments, and plots were conducted in Julia. The workflow included the following steps:

- **Calling C++ Heuristic**: The compiled C++ heuristic code was called externally from Julia. Input data was passed as files to the C++ program.
- **Data Handling**: The output from the C++ heuristic was read back into Julia for further analysis.
- **Visualization**: Results were plotted using the `Plots.jl` package in Julia.

4.1.4 Drone Generation

The experimental setup involved generating random samples for the drones' initial and final positions, varying the number of drones and the size of the grid. For each drone $k \in \{1, \ldots, K\}$, where K is the current number of drones, the starting position $(x_k^{\text{start}}, y_k^{\text{start}})$ and ending position $(x_k^{\text{end}}, y_k^{\text{end}})$ were randomly selected within a grid of size $N \times M$, where N and M are the dimensions of the grid.

Formally, for each number of drones $K \in \{1, \ldots, R\}$, where R is the maximum number of drones, and for each sample/trial $s \in \{1, \ldots, S\}$, the following procedure was applied:

- **Drone Generation**: For each drone $k \in \{1, \ldots, K\}$:
  - Randomly generate $(x_k^{\text{start}}, y_k^{\text{start}}) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$
  - Randomly generate $(x_k^{\text{end}}, y_k^{\text{end}}) \in \{1, \ldots, N\} \times \{1, \ldots, M\}$, ensuring $(x_k^{\text{end}}, y_k^{\text{end}}) \neq (x_k^{\text{start}}, y_k^{\text{start}})$

4.2 Models Comparison

In this section, we compare the two models in terms of optimality(objective performance) and computational efficiency(running time). The x-axis in each plot represents the number of drones used to generate 20 samples per number of drones in the box plots, conducted in a 5x5 grid. The input T for the exact model in each sample is the one returned by the heuristic, that is, $T_{\text{exact\_model}} = T_{\text{heuristic}}$.

4.2.1 Objective Function Performance

We first evaluate the performance of both models in terms of their objective function outcomes. The objective function (y-axis) represents the optimization goal of minimizing path lengths for drones while avoiding collisions. Figure 6 and Figure 7 depict box plots

comparing the objective function values achieved by the exact model and the heuristic model, respectively.

Figure 6 presents the distribution of objective function values obtained by the exact model, illustrating the spread and central tendency of the outcomes across different scenarios. Conversely, Figure 7 showcases the corresponding results for the heuristic model. By comparing these plots, we can gain valuable insights into the performance of each model in optimizing the objective function.

In the scenario where $n_{\text{drones}} = 1$, the heuristic model performs as optimally as the exact model since the BFS guarantees optimality in this simple case. However, as the number of drones increases, the performance of the heuristic model diminishes. This decline in performance is anticipated due to the increased traffic within a fixed grid size, leading to more complex interactions and potential conflicts between the drones.

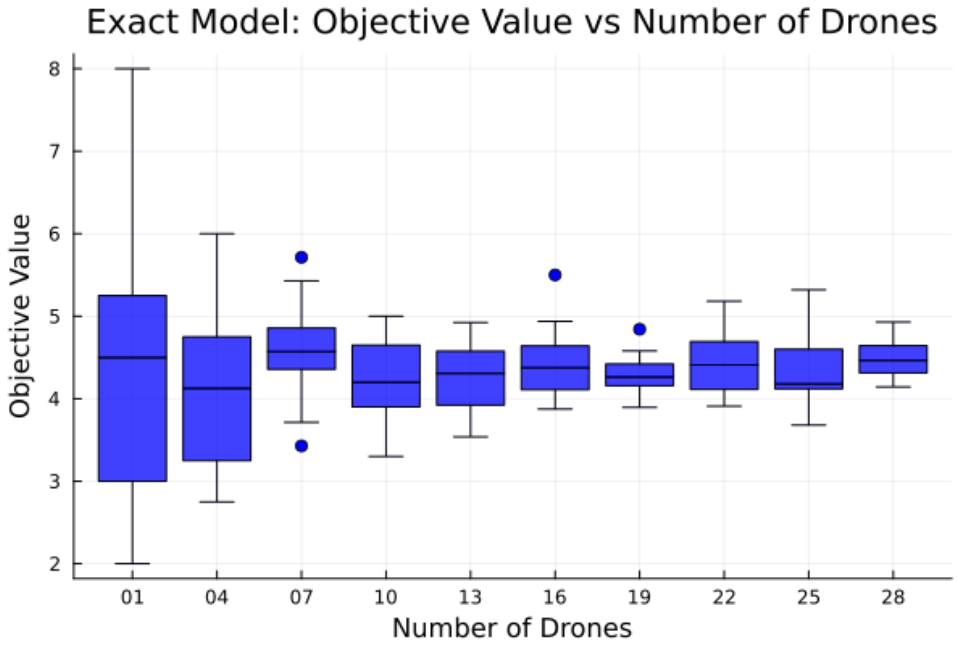

Figure 6 – Exact Model Objective. Source: The authors.

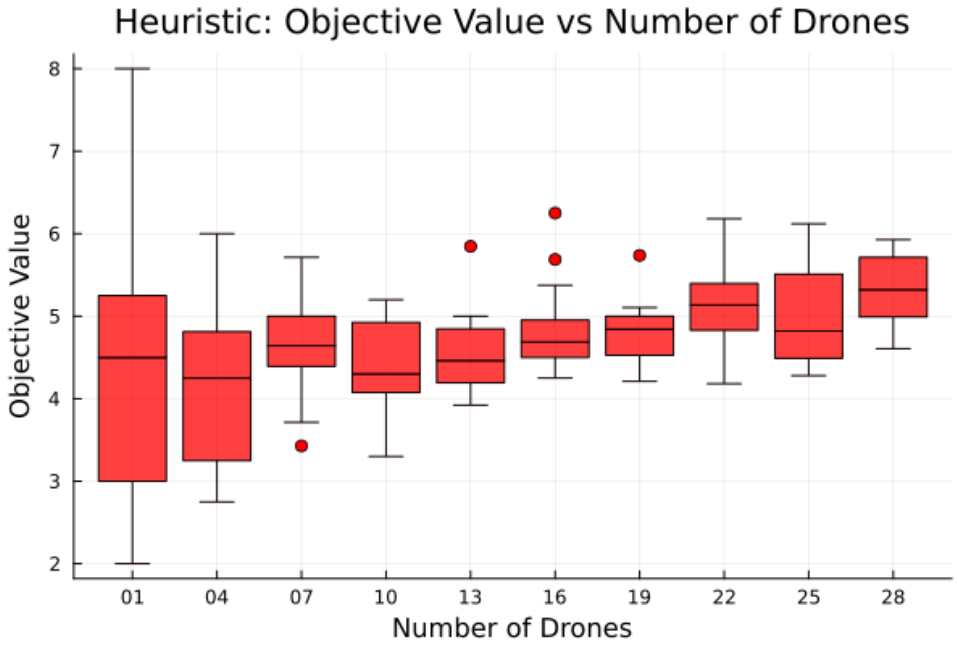

Figure 7 – Heuristic Objective. Source: The authors.

4.2.2 Computational Efficiency

Next, we analyze the computational efficiency of the two models, focusing on the time required (y-axis) to compute the optimal paths for drones. Figures 8 and 9 display box plots representing the computational time taken by the exact model and the heuristic model, respectively.

Figure 8 illustrates the distribution of computational times for the exact model. This plot provides an understanding of the computational overhead associated with solving the optimization problem precisely. Conversely, Figure 9 presents the computational times required by the heuristic model. By comparing these plots, we assess the trade-off between computational efficiency and solution optimality offered by each model.

As illustrated in Figure 8, the heuristic consistently exhibits a lower running time compared to the exact model. This disparity in running times is not merely a constant difference but reflects the inherent differences in computational complexity between the two approaches. The heuristic model demonstrates an almost constant running time, attributed to its sub-linear performance relative to the number of drones, as discussed in Section 3.2.2. In contrast, the exact model, being NP-complete, has constraints that cause its performance to depend exponentially on both the number of drones and the specified T. Consequently, the running time of the exact model increases exponentially with the number of drones, as evidenced by the plot in Figure 8.

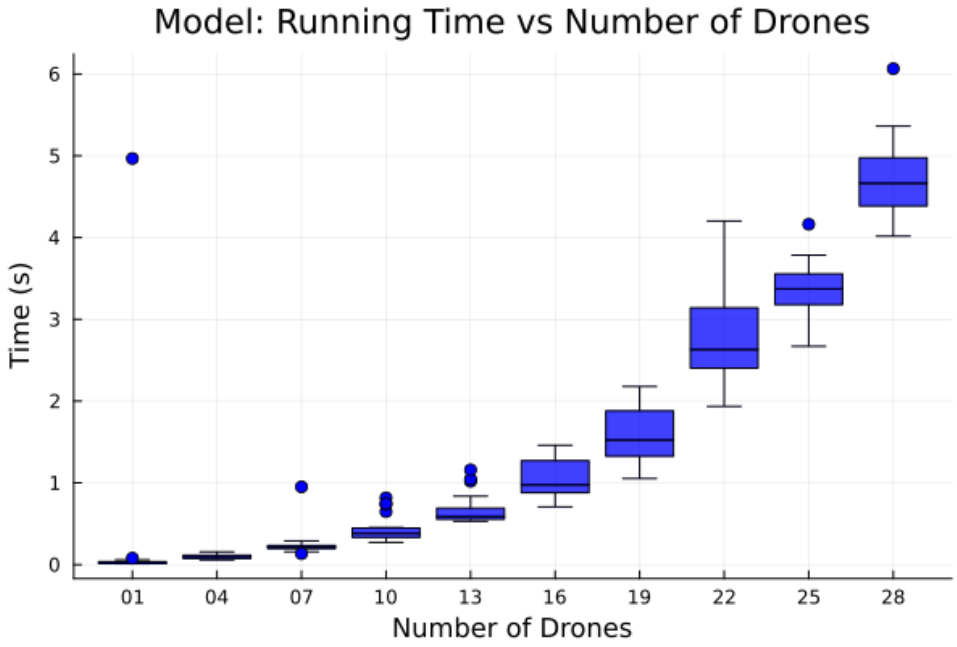

Figure 8 – Exact Model time. Source: The authors.

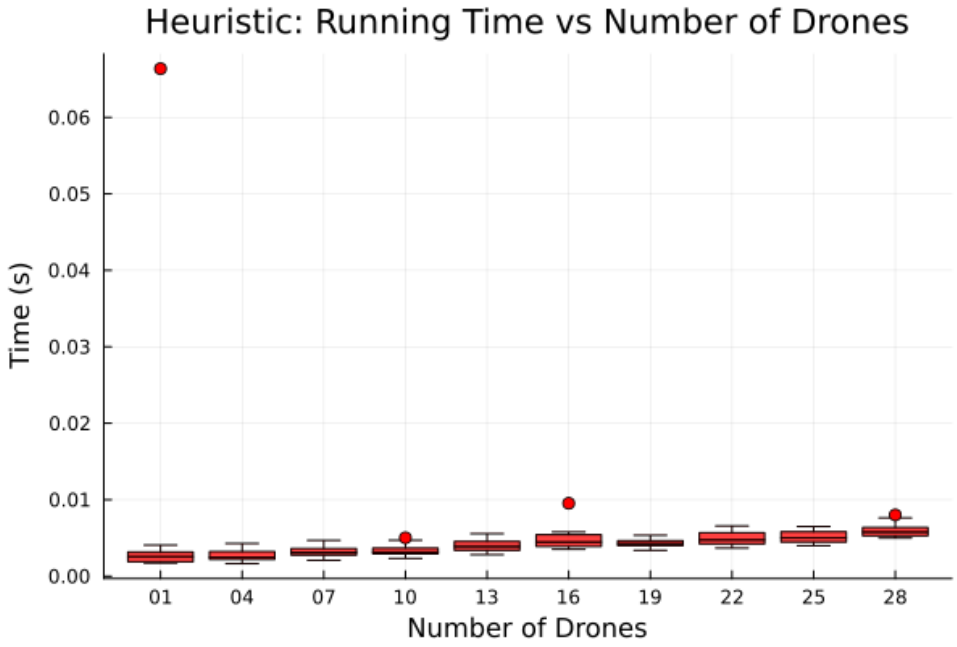

Figure 9 – Heuristic time. Source: The authors.

4.2.3 Heuristic Behavior and Transition

In this section, we propose an experimental setup to analyze the behavior of the heuristic in high-traffic configurations. We compare the heuristic's performance with the optimal distance for each drone in a collision-free environment, where the optimal path for each drone does not consider other drones in the grid (Manhattan distance).

Primarily, we define the optimal distance per drone as the Manhattan distance between its start and end points. For each drone $k \in \mathcal{R}$, with starting position $(x_k^{\text{start}}, y_k^{\text{start}})$

and ending position $(x_k^{\text{end}}, y_k^{\text{end}})$, the Manhattan distance D_k is given by:

$$D_k = |x_k^{\text{start}} - x_k^{\text{end}}| + |y_k^{\text{start}} - y_k^{\text{end}}|.$$

We then calculate the actual path length P_k for each drone k based on the heuristic path, and compute the optimality ratio R_k as follows:

$$R_k = \frac{P_k}{D_k}.$$

The optimality ratio provides a measure of how efficiently the heuristic performs compared to the ideal (collision-free) path.

The experiments were conducted using a 6×6 grid with the number of drones ranging from 1 to 99. For each number of drones, 30 points were sampled.

The visualization of the heuristic performance can be measured in several ways. For our purpose, we chose two metrics:

- **Normalized Distance**: This metric shows the actual path length taken by each drone normalized by the Manhattan distance. It illustrates how the heuristic manages to find paths in high-traffic scenarios compared to the optimal path lengths.

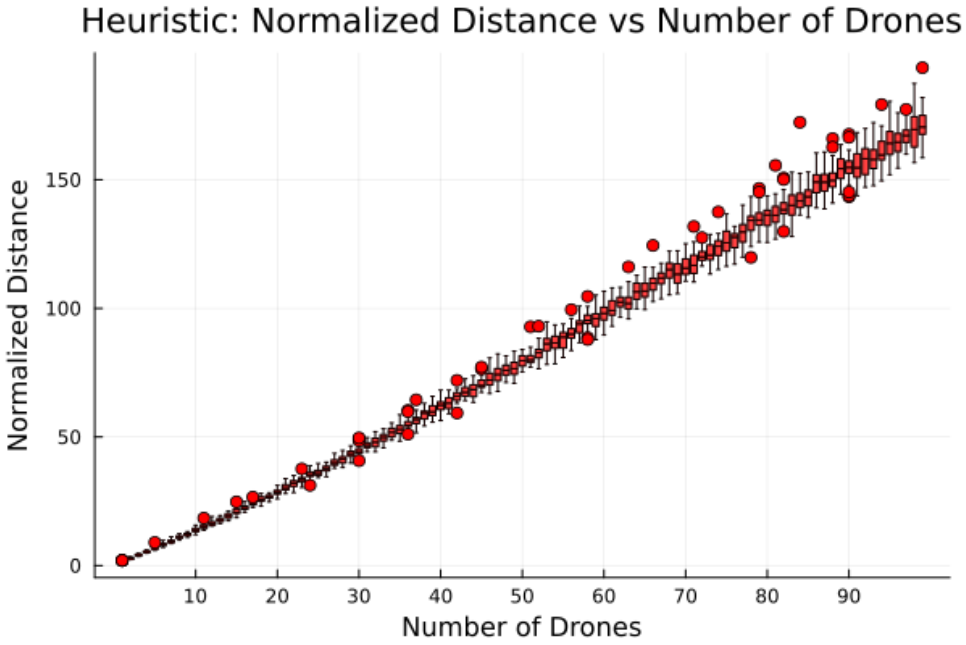

Figure 10 – Heuristic normalized distance per drone. Source: The authors.

Figure 10 displays the distribution of normalized distances for the heuristic, highlighting how the path lengths vary with an increasing number of drones. The box plot provides insights into the spread and central tendency of the heuristic's performance.

- **Routing Time (T)**: This metric shows the time taken for the heuristic to compute the paths for all drones. It provides an understanding of the computational efficiency of the heuristic as the traffic increases.

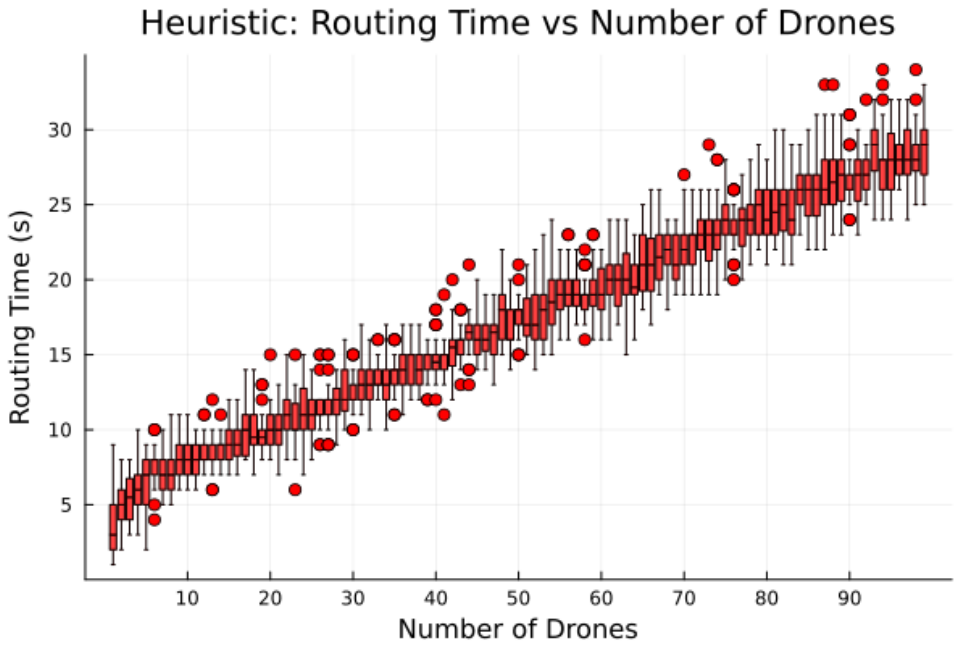

Figure 11 – Heuristic Routing Time (T) corresponding to Figure 10. Source: The authors.

Figure 11 illustrates the routing times required by the heuristic as the number of drones increases. The box plot shows the variability in routing times and helps identify any trends or outliers. This result evinces the reasonability of the hybrid method in 3.3, since we have a T that increases just linearly in the number of drones, even in a high traffic condition(36 cells available cells and 99 drones).

Together, these figures provide a comprehensive view of the heuristic's behavior under varying traffic conditions, demonstrating both its effectiveness in finding feasible paths and its computational efficiency.

5 CONCLUSION

This chapter summarizes the findings and contributions of this research, highlights the relationship between the undergraduate course and the project, discusses the implications and limitations of the study, and suggests directions for future research.

5.1 Summary of the Project and Objectives

The primary objective of this research was to develop and analyze an exact model and a heuristic for optimizing drone routing in centralized last-mile delivery scenarios. This involved creating a heuristic algorithm implemented in C++ and comparing its performance against the exact model implemented in Julia. The study focused on evaluating the heuristic's efficiency and effectiveness under high-traffic conditions and comparing its results to the optimal paths determined by the exact model.

5.2 Main Findings and Results

The experiments demonstrated that the heuristic algorithm, while not always achieving the optimal solution, provided satisfactory results with significantly lower computational time compared to the exact model. Key findings include:

- The heuristic algorithm achieved optimal results for scenarios with a single drone, as BFS guarantees optimality in such cases.
- As the number of drones increased, the heuristic's performance decreased, which is expected due to higher traffic in the fixed grid size.
- The heuristic maintained sub-linear performance concerning the number of drones, indicating its scalability for larger problems.
- The exact model's computational time increased exponentially with the number of drones, consistent with its NP-complete nature.
- The use of a hybrid methodology, as described in 3.3, would be a good choice.

5.3 Contributions to the Field and Implications

This research contributes to the field of drone logistics and optimization by:

- Providing a practical and efficient heuristic for drone routing that can be applied in real-world scenarios with multiple drones.

- Highlighting the trade-offs between computational efficiency and optimality in heuristic algorithms.
- Demonstrating the applicability of heuristic methods in high-traffic conditions, offering a scalable solution for last-mile delivery challenges.

The implications of these findings suggest that heuristic algorithms can effectively complement exact models in scenarios where computational resources are limited or quick solutions are required.

5.4 Decentralized vs Centralized

Decentralized systems, like the one proposed in (VERRI *et al.*, 2020), operate as dynamic processes where individual naive agents make localized decisions. However, this decentralized decision-making can limit global optimization, leading to inefficiencies in airspace management and route planning.

In contrast, our centralized approach demonstrated superior efficiency through global optimization, enabling more effective airspace management and route planning. By leveraging centralized control, we achieved better performance in optimizing drone routes and minimizing delivery times.

This comparison is easily observed when visualizing the experiments in Verri *et al.* (2020) versus the ones developed here. It is evident that the increase in the number of drones (arrival time in Verri *et al.* (2020)) results in significantly longer paths and wait times compared to the exact model and heuristic proposed in our study.

While decentralized systems offer flexibility, they may struggle to achieve the level of optimization and coordination provided by centralized approaches. Future research should focus on integrating the strengths of both approaches to develop hybrid models that combine the agility of decentralized decision-making with the efficiency of centralized optimization. This integration could involve incorporating intelligence into the agents in the decentralized approach, since **decentralized is the future**.

5.5 Limitations and Future Research Directions

This study has several limitations that need to be addressed in future research:

5.5.1 Limitations

- The experiments conducted in this study were confined to fixed and relatively small grid sizes. To comprehensively evaluate the efficacy of the heuristic and exact model, future studies should extend their analyses to larger grid sizes to capture the complexities and challenges inherent in more extensive environments.

- The utilization of the exact model solely with the T value obtained from the heuristic represents a simplification in the experimental methodology. To enhance the accuracy and robustness of the approach, future investigations could employ advanced techniques, such as binary search in $\mathcal{O}(\log(T))$, to efficiently determine the optimal T value while ensuring the model's feasibility.

- Both the heuristic and exact model proposed in this study operate under simplified assumptions regarding airspace conditions, neglecting factors such as drone speed variations and environmental conditions. Future research should strive to incorporate these real-world complexities into the modeling framework to enhance the accuracy and applicability of the solutions.

- The comparative analysis in this study focused exclusively on contrasting the heuristic against the exact model, overlooking the potential benefits of exploring alternative hybrid, heuristic, and metaheuristic approaches.

5.5.2 Future Research

Here we focus on advancements for the decentralized approach.

- Advancing research on the decentralized model by exploring techniques from Dynamical/Stochastic Processes and Sequential Decision Optimization. These approaches can enhance the adaptability and robustness of decentralized systems in dynamic environments.

- Investigating reinforcement learning (RL) techniques tailored for decentralized drone logistics. RL algorithms can learn optimal policies for drone routing and airspace management, improving efficiency and adaptability.

- Leveraging Graph Neural Networks (GNNs) and Complex Networks approaches to develop policies for decentralized systems (XIAO *et al.*, 2023). By incorporating insights from centrality measures and influence maximization, RL agents can prioritize actions that have the greatest impact on system dynamics.

- Enhancing exploration efficiency in decentralized systems by leveraging knowledge of the network's topology. By understanding node importance and connectivity, RL agents can navigate the environment more effectively, leading to improved learning and optimization of strategies.

5.6 Project contributions to the student

The project contributed to me because it was an excellent way to reinforce part of the knowledge passed on in the undergraduate course as algorithms and Graph applications.

Furthermore, it was a great way to start studying *optimization* and become aware of its techniques and methodologies because I had the opportunity to do lots of experiments and coding.

5.7 Relation between the undergraduate course and the project

The undergraduate course is heavily connected with the developed project. In particular, one can highlight the subjects of Algorithms and Data Structures, Discrete Mathematics, Computational Modeling in Graphs, Mathematical Programming, Theory of Computation and Formal Languages and Advanced Algorithms and Applications. These subjects present a wide range of knowledge and learning related to data structures, mixed integer linear programming, algorithm complexity and development. Overall, the undergraduate course provides a solid theoretical foundation in Computer Science.

Another fundamental point of the course was the opportunities for research development, which improved my ability to communicate ideas and solutions. In addition, the presence of study and extension groups further complements the contents of the bachelor's degree.

REFERENCES

AHMADI, E.; WICAKSONO, H.; VALILAI, O. F. Extending the last mile delivery routing problem for enhancing sustainability by drones using a sentiment analysis approach. *In*: **2021 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)**. [*S.l.: s.n.*], 2021. p. 207–212.

AKAMATSU, T.; WADA, K. Tradable network permits: A new scheme for the most efficient use of network capacity. **Transportation Research Part C: Emerging Technologies**, v. 79, p. 178–195, 2017. ISSN 0968-090X. Available at: https://www.sciencedirect.com/science/article/pii/S0968090X17300839.

BOYSEN, N.; FEDTKE, S.; SCHWERDFEGER, S. Last-mile delivery concepts: a survey from an operational research perspective. **OR Spectrum**, v. 43, p. 1–58, 2021. Available at: https://doi.org/10.1007/s00291-020-00607-8.

BRUNI, M. E.; KHODAPARASTI, S.; PERBOLI, G. Energy efficient uav-based last-mile delivery: A tactical-operational model with shared depots and non-linear energy consumption. **IEEE Access**, v. 11, p. 18560–18570, 2023.

DUKKANCI, O.; CAMPBELL, J. F.; KARA, B. Y. Facility location decisions for drone delivery: A literature review. **European Journal of Operational Research**, 2023. ISSN 0377-2217. Available at: https://www.sciencedirect.com/science/article/pii/S0377221723008123.

FAIçAL, B. S. *et al.* A cyber-physical system's roadmap to last-mile delivery drones. **IEEE Aerospace and Electronic Systems Magazine**, v. 38, n. 5, p. 6–16, 2023.

FREITAS, J. C.; PENNA, P. H. V.; TOFFOLO, T. A. Exact and heuristic approaches to truck–drone delivery problems. **EURO Journal on Transportation and Logistics**, v. 12, p. 100094, 2023. ISSN 2192-4376. Available at: https://www.sciencedirect.com/science/article/pii/S219243762200019X.

GEFT, T. Fine-grained complexity analysis of multi-agent path finding on 2d grids. **Proceedings of the Sixteenth International Symposium on Combinatorial Search**, v. 16, n. 1, 2023. Available at: https://doi.org/10.1609/socs.v16i1.27279.

GORDON, O.; FILMUS, Y.; SALZMAN, O. Revisiting the complexity analysis of conflict-based search: New computational techniques and improved bounds. *In*: MA, H.; SERINA, I. (ed.). **Proceedings of the Fourteenth International Symposium on Combinatorial Search, SOCS 2021, Virtual Conference [Jinan, China], July 26-30, 2021**. AAAI Press, 2021. p. 64–72. Available at: https://doi.org/10.1609/socs.v12i1.18552.

HAYAT, S. *et al.* Multi-objective drone path planning for search and rescue with quality-of-service requirements. **Autonomous Robots**, Springer, v. 44, p. 1183–1198, 2020.

LAM, E. *et al.* Branch-and-cut-and-price for multi-agent pathfinding. *In*: **Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence,**

IJCAI-19. International Joint Conferences on Artificial Intelligence Organization, 2019. p. 1289–1296. Available at: https://doi.org/10.24963/ijcai.2019/179.

LEE, C.-W.; WONG, W.-P. Last-mile drone delivery combinatorial double auction model using multi-objective evolutionary algorithms. **Soft Computing**, Springer, v. 26, p. 12355–12384, 2022. Available at: https://doi.org/10.1007/s00500-022-07094-9.

LIEB, J.; VOLKERT, A. Unmanned aircraft systems traffic management: A comparsion on the faa utm and the european corus conops based on u-space. *In*: **2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)**. [*S.l.: s.n.*], 2020. p. 1–6.

LUBIN, M. *et al.* JuMP 1.0: Recent improvements to a modeling language for mathematical optimization. **Mathematical Programming Computation**, 2023.

LUO, Q. *et al.* Hybrid multi-objective optimization approach with pareto local search for collaborative truck-drone routing problems considering flexible time windows. **IEEE Transactions on Intelligent Transportation Systems**, v. 23, n. 8, p. 13011–13025, 2022.

MA, H. *et al.* Searching with consistent prioritization for multi-agent path finding. *In*: **Proceedings of the AAAI conference on artificial intelligence**. [*S.l.: s.n.*], 2019. v. 33, n. 01, p. 7643–7650.

MOSHREF-JAVADI, M.; HEMMATI, A.; WINKENBACH, M. A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. **Applied Mathematical Modelling**, v. 80, p. 290–318, 2020. ISSN 0307-904X. Available at: https://www.sciencedirect.com/science/article/pii/S0307904X19306936.

NEBEL, B. On the computational complexity of multi-agent pathfinding on directed graphs. **Proceedings of the International Conference on Automated Planning and Scheduling**, v. 30, n. 1, p. 212–216, Jun. 2020. Available at: https://ojs.aaai.org/index.php/ICAPS/article/view/6663.

NUR, F. *et al.* Last mile delivery drone selection and evaluation using the interval-valued inferential fuzzy topsis. **Journal of Computational Design and Engineering**, v. 7, n. 4, p. 397–411, August 2020.

PREVOT, T. *et al.* Uas traffic management (utm) concept of operations to safely enable low altitude flight operations. *In*: _____. **16th AIAA Aviation Technology, Integration, and Operations Conference**. [*S.l.: s.n.*]. Available at: https://arc.aiaa.org/doi/abs/10.2514/6.2016-3292.

RINALDI, M. *et al.* Development of heuristic approaches for last-mile delivery tsp with a truck and multiple drones. **Drones**, v. 7, n. 7, 2023. ISSN 2504-446X. Available at: https://www.mdpi.com/2504-446X/7/7/407.

SHARON, G. *et al.* Conflict-based search for optimal multi-agent pathfinding. **Artificial Intelligence**, v. 219, p. 40–66, 2015. ISSN 0004-3702. Available at: https://www.sciencedirect.com/science/article/pii/S0004370214001386.

STERN, R. *et al.* Multi-agent pathfinding: Definitions, variants, and benchmarks. *In*: **Proceedings of the International Symposium on Combinatorial Search**. [*S.l.: s.n.*], 2019. v. 10, n. 1, p. 151–158.

SURYNEK, P. Problem compilation for multi-agent path finding: a survey. *In*: RAEDT, L. D. (ed.). **Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22**. International Joint Conferences on Artificial Intelligence Organization, 2022. p. 5615–5622. Survey Track. Available at: https://doi.org/10.24963/ijcai.2022/783.

VERRI, F. A. N. *et al.* An analysis on tradable permit models for last-mile delivery drones. **IEEE Access**, v. 8, p. 186279–186290, 2020.

WEISE, J.; MOSTAGHIM, S. A comparison of distance metrics for the multi-objective pathfinding problem. **Natural Computing**, v. 22, p. 315–328, 2023.

XIAO, J. *et al.* A graph neural network based deep reinforcement learning algorithm for multi-agent leader-follower flocking. **Information Sciences**, v. 641, p. 119074, 2023. ISSN 0020-0255. Available at: https://www.sciencedirect.com/science/article/pii/S002002552300659X.

YU, J.; LAVALLE. **Time-Optimal Multi-Agent Path Finding**. 2012. Available at: https://movingai.com/mapf/slides/time-optimal.pdf.

YU, J.; LAVALLE, S. Time optimal multi-agent path planning on graphs. *In*: . [*S.l.: s.n.*], 2012. p. 58–59. 2012 AAAI Workshop ; Conference date: 22-07-2012 Through 22-07-2012.

YU, J.; LAVALLE, S. M. Multi-agent path planning and network flow. *In*: FRAZZOLI, E. *et al.* (ed.). **Algorithmic Foundations of Robotics X**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 157–173. ISBN 978-3-642-36279-8.

ZHANG, S. *et al.* A novel multi-objective optimization model for the vehicle routing problem with drone delivery and dynamic flight endurance. **Computers & Industrial Engineering**, v. 173, p. 108679, 2022. ISSN 0360-8352. Available at: https://www.sciencedirect.com/science/article/pii/S0360835222006672.

ČáP, M. *et al.* Prioritized planning algorithms for trajectory coordination of multiple mobile robots. **IEEE Transactions on Automation Science and Engineering**, v. 12, n. 3, p. 835–849, 2015.