Research of the route planning algorithms on the example of a drone delivery system software development

Yevhen L. Turchyk[1], Milana V. Puzino[2], Olena H. Rybalchenko[1] and Svitlana V. Bilashenko[1]

[1]Kryvyi Rih National University, 11 Vitalii Matusevych Str., Kryvyi Rih, 50027, Ukraine
[2]Lviv Polytechnic National University, 12 Stepana Bandery Str., Lviv, 79000, Ukraine

Abstract

The paper analyzes the existing drone delivery systems all around the world. Different route building algorithms are analyzed for navigating the drones through the cities, advantages and disadvantages of all the approaches are highlighted. Requirements for the system are defined that must provide quick and convenient operation; the system was planned and developed. It was concluded that the designed system has a great potential for real usage and further development.

Keywords

drone delivery, UAV, path finding, route building, machine learning

1. Introduction

In the modern world, there is a noticeable acceleration of the pace of life in large cities. The efficiency of businesses and the quality of life for individuals depend directly on well-established logistics. This issue is particularly pronounced in "last-mile" delivery, where the transportation of goods is influenced by various external factors that impact its speed and effectiveness. First and foremost, human labor involved in product delivery is limited by physical and psychological aspects, and the increasing demand for speed may exceed the capabilities of personnel. The involvement of significant resources in the delivery process can result in increased delivery costs, subsequently raising the prices of goods and services for both businesses and end consumers.

Drone delivery can address these challenges. The use of drones in delivery can reduce the dependency on human labor. Unmanned aerial vehicles (UAVs) can operate around the clock without rest, providing fast and precise product delivery when using advanced navigation and route building algorithms [1].

CS&SE@SW 2023: 6th Workshop for Young Scientists in Computer Science & Software Engineering, February 2, 2024, Kryvyi Rih, Ukraine

itsjonny5757@gmail.com (Y. L. Turchyk); milana.puzino.mpzip.2022@lpnu.ua (M. V. Puzino); rybalchenko@knu.edu.ua (O. H. Rybalchenko); bilashenko.s@knu.edu.ua (S. V. Bilashenko)
http://mpz.knu.edu.ua/vikladachi/olena-rybalchenko (O. H. Rybalchenko); http://mpz.knu.edu.ua/vikladachi/Svetlana_Bilashenko (S. V. Bilashenko)
0009-0006-3254-9411 (Y. L. Turchyk); 0009-0007-9450-630X (M. V. Puzino); 0000-0001-8691-5401 (O. H. Rybalchenko); 0000-0002-4331-7425 (S. V. Bilashenko)

Therefore, the primary idea of this work is to study the path finding algorithms and develop a software for automated aerial drone delivery to address the issue of "same-day" delivery between different city branches and accelerate it. To achieve this, a comprehensive software solution is proposed to implement a similar service and explore the capabilities of route planning algorithms for the automatic operation of UAVs.

2. Review of the subject area and existing solutions analysis for the development of drone delivery system

2.1. Existing systems analysis

Currently, there are relatively few existing and fully operational drone delivery analogs on both the Ukrainian and global markets. Most of the available services are either in the testing or development stages or operate within limited geographical areas. Additionally, the majority of these services are oriented towards "last-mile" delivery, which restricts users from utilizing the service for non-commercial purposes. Let's examine some of the most well-known analogs within the mentioned category.

2.1.1. Amazon Prime Air

Amazon Prime Air is an aerial drone delivery project developed by Amazon since 2013, aiming to provide rapid delivery of packages to recipients within 30 minutes [2].

Currently, the system is in the testing phase, conducted in the city of Lockford, California, USA. It is expected that based on feedback from local residents using the delivery service, the most problematic aspects will be identified and addressed. Through testing with a wide variety of cargo sizes and weights, the reliability and durability of the project's technical equipment have been verified.

The following advantages of this service can be highlighted:

- According to reports from Amazon Prime Air project specialists, unique software for drones has been developed, allowing UAVs to safely detect and avoid potential obstacles, making the point-to-point flight process more reliable.
- Amazon Prime Air drones have a high payload capacity from the outset.

However, it is worth noting the drawbacks of such a system:

- The process of delivering cargo to the recipient involves dropping it from a specified height onto the backyard of a private house, limiting the potential user base to those with suitable delivery locations. This approach may not guarantee the safe delivery of potentially fragile cargo.
- This system is planned to be used exclusively for delivering goods from the Amazon store, which narrows down the pool of potential users.

It is expected that users will have the option to order drone delivery of selected items through the Amazon online store, and as such, this service will not have its own separate user software but will be integrated into the existing services of the company.

2.1.2. Starship Technologies

Starship Technologies is an Estonian startup (later becoming a company) initiated in 2014, addressing the "last-mile" delivery problem using ground-based drones [3].

This service operates in London, Tallinn, Düsseldorf, Hamburg, Bern, and in some cities in the United States, such as Washington, D.C., and Mountain View, California. Among the advantages of this startup, the following points are noteworthy:

- The drones are fully autonomous, allowing them to independently locate and load the required product into their cargo compartment.
- Delivery is secure for the recipient since receiving an order is only possible after entering a personal security code.
- In case of navigation issues with the drone, the system provides remote control by a human pilot.

However, there are some limitations to the startup:

- Deliveries are only made within a 5-kilometer radius.
- The maximum drone speed is determined by the quality of the terrain and does not exceed 6.5 km/h, which is nearly equivalent to human walking speed.
- The project is exclusively oriented towards delivering food weighing up to 9 kg.

This described service has a dedicated user application through which the ordering, payment, package tracking, and other processes are conducted.

2.1.3. Zipline

Zipline is an American project involved in the manufacturing and delivery of air drones/aircraft [4]. The main concept of the company is to address the issue of delivering cargo to hard-to-reach locations.

Currently, the Zipline service is available in Rwanda, Ghana, Nigeria, Japan, and the United States. Additionally, it is expected that the company's services will soon become available in Côte d'Ivoire and Kenya. Furthermore, the Ministry of Health of Ukraine has announced negotiations regarding a potential partnership with the company.

Notable advantages of the Zipline service include:

- Ensuring high delivery speed, even for long distances, thanks to the mobility of the drone-aircraft (UAV speed reaches 101 km/h).
- Autonomous drone flights are possible under normal weather conditions.
- Drones have the capability to move between pre-established airstrips, where pilots can manually replace batteries or cargo for delivery, thus ensuring improved logistics and the range of package dispatch.

Among the drawbacks of the project, the following points should be noted:

- The cargo capacity of the drone is limited to 1.8 kg.

- Delivery to the destination occurs by dropping the package from an elevated position (20-35 m), and the pre-packed package descends slowly using a paper parachute. Consequently, the distance from the actual landing point to the anticipated one may vary up to 5 meters.
- The service primarily deals with the delivery of medicines or related medical items. The list of possible non-medical types of packages is limited to restaurant or grocery products, and the like.

It is also worth adding that the company has developed the next generation of drones that deliver cargo via a tether instead of deploying a parachute. However, this type of delivery significantly reduces the range and speed.

Users of the service can place orders on the company's website and monitor the delivery process through a dedicated mobile application.

2.2. Analysis of the latest research for drone delivery systems

The delivery by drones involves the management of a large number of Unmanned Aerial Vehicles (UAVs) simultaneously. The logistics challenges of such operations have been extensively discussed in [5], where a multi-physics model at the system level is presented for optimal control of multi-engine UAVs. This model can be utilized in the development and evaluation of control strategies. The authors demonstrate the capability of using this model for basic maneuvers and lay the groundwork for planning more complex maneuvers and complete missions. For instance, drones may employ different control strategies for achieving maximum energy efficiency under high and low battery levels.

The proposed system offers the following advantages:

- The multi-physics model enables a more comprehensive and accurate representation of the dynamics and interaction between different physical systems of multi-engine UAVs.
- An optimal control strategy is developed to minimize a cost function that considers time and energy consumption.
- The proposed methods are flexible and adaptable to various types of UAVs or other aerial systems, making them valuable for a wide range of applications in transportation, surveillance, mapping, etc.

During drone flight, a significant amount of computation is required to adjust the process and mission specifics of UAVs. The transfer of computational load from the drone to a cloud structure is discussed in [6]. The described framework features a client-server architecture, positioning the drone as a client and the cloud as a scalable server. Overall, it has potential applications in various fields requiring efficient drone management, especially in the delivery sector.

The proposed system has the following advantages:

- Scalability: The client-server architecture of the framework ensures effective communication between multiple drones and the cloud server, enabling real-time control of a large number of drones.
- Efficiency: By offloading certain tasks to the cloud server, the workload on individual drones is reduced, allowing them to operate more efficiently.

- Open source: The framework is open source, allowing developers to freely use and modify it.
- Versatility: The framework has potential applications in various industrial sectors.

A crucial aspect of a drone's mission during delivery is the route planning from the collection point to the delivery point. The optimization of the sequence of these processes is discussed in [7]. The delivery problem involves a group of couriers (drones) timely delivering orders to clients. The goal of the algorithm is to increase profit over a specific time interval and reduce the overall delivery time. The authors propose a Markov decision process model for the "courier" assignment task, using deep learning algorithms to address the problem in a dynamic environment. Successful implementation of this algorithm could significantly impact the delivery industry, enhancing its speed and increasing company profits.

An important task for optimizing the algorithm in a drone delivery system is to consider the drone's battery usage. Aiello et al. [8] presents a model of energy consumption for a similar urban logistics infrastructure. This methodology allows considering various factors affecting drone battery consumption, such as cargo weight, the size of the serviced urban area, population density, flight range, built-in battery capacity, etc. The model was developed to help researchers better understand the energy needs of delivery systems using UAVs and identify ways to optimize their performance.

2.3. Review of common approaches and algorithms for drone delivery route planning

The main stage in any type of cargo transportation is the process of route planning, for which there are currently numerous algorithms aimed at solving transportation problems efficiently and quickly. These algorithms are a crucial component of logistics and transportation infrastructure.

Such problems arise when it is necessary to determine the optimal delivery route from the point of origin to the destination, taking into account various external factors such as distance, cost, time constraints, and resources. As a result, this algorithmic process can become quite complex, especially when dealing with a large number of delivery points in complex urban conditions.

Navigating the UAV through the city is a tough task involving many safety preconditions, so an optimal way would be to deliver the packages to the delivery offices all around the city. Such an approach would allow to manually build the safe routes between many adjacent departments, thus creating a graph with nodes and branches of given cost (routes length), where we need to find a path to navigate.

Let's consider the most common approaches to solving such a problem.

2.3.1. Traveling Salesman Problem algorithm

One of the most common and straightforward methods for building a delivery route is the Traveling Salesman Problem (TSP) algorithm. It is based on a mathematical model that helps find the shortest path that connects all given pickup and delivery points. The TSP algorithm takes into account various factors, such as the distance between points, loading and unloading times, vehicle capacity constraints, and other limitations.

The main advantage of the Traveling Salesman Problem algorithm is its simplicity and ease of implementation. This algorithm uses a brute-force approach, where all possible combinations between points are considered. This makes it accessible for use in various fields and research and simplifies its integration with other parts of the software code.

There are many variations of the Traveling Salesman Problem-solving methods, such as the Monte Carlo method, the method of averaged coefficients, or the nearest neighbor method. The nearest neighbor method uses heuristic estimation in its calculations, significantly speeding up the search for the optimal route but not guaranteeing absolute optimality.

However, it is worth noting the disadvantages of this algorithm. Since the Traveling Salesman Problem algorithm, at each point, must choose the next point from those it has not yet visited, there are $(n-1)!$ routes for the asymmetric and $\frac{(n-1)!}{2}$ routes for the symmetric Traveling Salesman Problem. This means that the size of the search space depends exponentially on the number of points. For an average-sized problem, finding the optimal route can take an unacceptably long time, as most of it is spent on the enumeration of all possible combinations between points, which requires significant computational resources.

Another drawback of the Traveling Salesman Problem algorithm is that it provides only an approximate solution and does not guarantee finding the shortest path. Consequently, the accuracy of these calculations decreases proportionally as the problem size increases.

2.3.2. Dijkstra's algorithm

Dijkstra's algorithm is one of the most common algorithms for finding the optimal path in a graph and has broad applications in various fields, including telecommunications, transportation networks, routing, and logistics planning.

The working principle of the Dijkstra's algorithm involves iteratively updating the shortest distances from the initial node of the graph to all other nodes. During its operation, each vertex is examined, and the distance to adjacent vertices is calculated using the corresponding edges [9]. As a result of its work, the Dijkstra's algorithm not only determines the shortest distances from the initial node to all other nodes but also memorizes the corresponding routes.

The Dijkstra's algorithm is quite efficient and performs well at optimal scales. Its execution time depends on the number of vertices and edges in the graph, but with proper implementation, it has a time complexity of $O(n^2)$, where n is the number of vertices. Despite the fact that finding a route involves exploring all possible path variations, this feature can be considered an advantage to some extent because having data about all available routes guarantees the optimality of the found solution.

It is evident that as the number of vertices and edges in the input graph increases, exploring all possible variations will significantly slow down the process of finding the shortest path, rendering the algorithm unsuitable for use with such input data.

2.3.3. A* algorithm

The A* (A-star) algorithm is also aimed at finding a path in a graph and is an improved version of the Dijkstra's algorithm.

To achieve maximum efficiency with the A* algorithm, the heuristic function should be chosen according to the specific problem, as there is no one-size-fits-all solution. When tying the final path cost to the distance, more efficient heuristic functions such as the Euclidean distance or the Manhattan metric should become the preference.

One of the key advantages of the A* algorithm is its efficiency compared to the Dijkstra's algorithm. It uses a heuristic estimate (denoted as "h") to calculate the distance from the current node to the final destination. This heuristic helps the algorithm make decisions about which node is likely to lead to the shortest path. When the heuristic function is optimistic (i.e., it doesn't overestimate the distance), the A* algorithm guarantees finding the shortest path.

All of these factors make A* a popular choice and an efficient tool for route planning and optimization in various fields, including robotics, artificial intelligence development, and routing.

Another advantage of the A* algorithm is its ability to handle graphs of moderate size and relatively complex problems efficiently. While the computational complexity depends on the graph's size, A* demonstrates high efficiency with optimal implementation. It can quickly find the shortest path when using a heuristic function that provides spatial orientation information.

Therefore, this algorithm is faster and more optimized for larger tasks compared to the Dijkstra's algorithm or the Traveling Salesman algorithm since it doesn't require exploring all possible route combinations.

However, one significant drawback of the A* algorithm is its potential to get trapped in local maxima. This means that an incorrectly defined heuristic function or an insufficiently informative estimate of a particular distance can influence the algorithm to choose the wrong path, which consequently is not the shortest. This can be problematic, especially when solution accuracy is critical, such as in robotics or automated route planning.

Another issue with the A* algorithm is high memory usage. Since it keeps track of all visited nodes, the memory requirements for storing this information can significantly increase for large input graphs or complex-sized problems. This may necessitate size limitations on problems that can be effectively solved using this algorithm without compromising its performance.

2.3.4. Reinforced learning

The fourth algorithm, considered when choosing a method for constructing an optimal route, employs a reinforcement learning approach to build delivery routes. It is based on machine learning concepts and uses the learning process to make decisions regarding the selection of the shortest and most efficient routes [7].

One of the advantages of reinforcement learning algorithm is its ability to self-learn and adapt to a dynamic environment. It can interact with the environment, learn based on provided rewards, and refine its strategy over time. This allows the algorithm to effectively operate in dynamic and uncertain situations, where predefined rules may be insufficient or inefficient.

Another advantage of the reinforcement learning algorithm is its ability to optimally utilize resources [10]. It can find a balance between exploring new possibilities and exploiting existing knowledge, maintaining a trade-off between exploration and task execution. This makes it valuable for real-time decision-making and managing complex systems like drone delivery.

The main drawback of this algorithm is the need for a large amount of data and proper data preparation. Reinforcement learning algorithm requires an adequate quantity of high-quality

initial data for effective learning. Improper data preparation can lead to errors during the model training phase, as the algorithm is sensitive to noise.

2.3.5. Choosing the final algorithm

To choose the appropriate algorithm for solving the drone delivery route problem, we first identified the main criteria and requirements for the developed software. Let's examine the identified issues in detail:

- Execution speed: the selected algorithm should be highly efficient in terms of computation time, as this ensures reduced delays in drone management.
- Scalability: the limitations of the algorithm regarding its maximum computational capacity and the size of the problem it can handle should be taken into account, ensuring scalability.
- Implementation simplicity: due to the extensive work involved in creating the software for the drone delivery system, the chosen algorithm should be relatively easy to integrate with other software modules. Guided by these requirements and criteria, let's evaluate the suitability of the previously analyzed most common route optimization algorithms.

The traveling salesman algorithm aims to find the shortest path that passes through each node in the graph and returns to the starting node. While it guarantees finding the shortest path and is relatively simple to implement, its computational complexity increases rapidly with the number of delivery points. This, in turn, affects processing speed and scalability, making it potentially less suitable for large-scale problems.

Dijkstra's algorithm is a classic approach to finding the shortest path in a graph with non-negative edge weights. It works by layer-wise propagation from the starting node to the destination. Its effectiveness lies in its ability to find the shortest path to every node in the graph. However, as the number of nodes and edges grows, the exhaustive search of all possible route combinations slows down the search process, affecting both processing speed and scalability.

The A* algorithm combines ideas from Dijkstra's algorithm and heuristic methods. It uses estimates of distances to the destination to expedite the search process. It can find the shortest path when information about the graph's structure is available. A* is particularly useful in complex state space problems or situations with limited resources. However, its computational complexity depends directly on the efficiency of the heuristic estimate. Additionally, it may require significant memory resources during route computation, which correlates with the input problem's size.

Reinforcement learning is a different approach to solving route optimization problems. It's based on the idea of training a model through trial and error. An agent learns to make decisions based on rewards and penalties received during specific actions. This approach allows the agent to adapt to changing environmental conditions and seek optimal solutions. While reinforcement learning can be time and resource-intensive during the training phase, the computational demands are primarily associated with the training phase rather than the actual deployment.

Considering the outlined criteria and requirements, the choice of algorithm for solving the drone delivery route problem depends on the specific characteristics and constraints of the problem, the availability of domain-specific information, and the balance between computational

complexity and scalability. Each of the algorithms mentioned has its strengths and weaknesses, making them suitable for different scenarios. The selection should be driven by the specific needs and goals of the drone delivery system.

Based on the analysis of the mentioned algorithms, it can be argued that reinforcement learning is the most optimal solution for the drone delivery route problem. Its ability to self-learn and adapt to changing conditions makes it an ideal choice. Reinforcement learning enables the system to quickly and efficiently determine the best route, avoid obstacles, and optimize delivery. Considering the need for speed and accuracy, this algorithm will facilitate optimal delivery with minimal resource consumption.

3. System development

3.1. General system architecture

Within the research of the main algorithms for the drone delivery software, a necessary step is to identify its key structural elements, essential for the implementation and operation of the chosen algorithm, and the methods of communication between them. It has been determined that such software should consist of three main modules, which, during the actual implementation, form a client-server architecture. The architectural structure of the system is schematically depicted in figure 1.

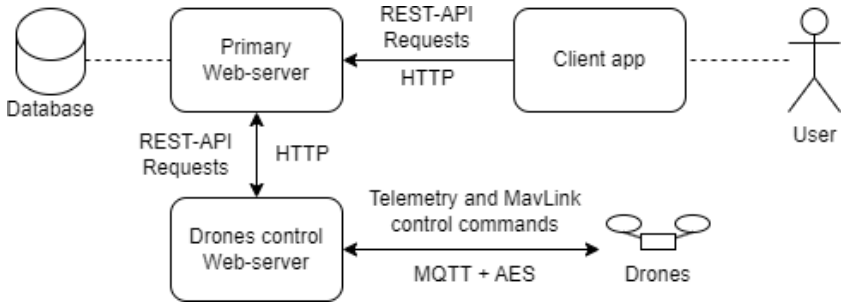

Figure 1: The architecture of the drone delivery system.

The client in this system is a user application, the purpose of which is to facilitate the user's interaction with the system.

Using the REST API interface, the application sends HTTP requests to a web server, which, in turn, processes them and performs necessary actions on the data, specifically basic CRUD operations (Create, Read, Update, Delete). This way, all client actions regarding interaction with the system, such as registration, creating or viewing lists of shipments, and their statuses, are handled.

A separate component of the system is the drone management module, which accumulates all the necessary methods and communication protocols. It also calculates optimal delivery routes. Utilizing the MQTT protocol [6], this part of the software creates a "Publisher-Subscriber" environment with the drones. This approach is commonly used in the Internet of Things [11, 12, 13] because it allows the server and hardware components to exchange messages freely

without the need for continuous monitoring of the system's status, as is required to receive updates through HTTP requests.

The data being transmitted consists of commands for drone control described using the "MavLink" protocol, which is a universal communication method with unmanned vehicles [14]. It enables both the sending of commands and receiving telemetry data from the drone, loading mission routes, and switching flight modes.

3.2. Hardware simulation

Due to limited testing capabilities caused by military actions in the territory of Ukraine, a crucial step is the selection of a sufficiently powerful and flexible technology for simulating the system's operation in a real environment. According to Chen et al. [15], one such technology is ArduPilot SITL, open-source autopilot software that allows the simulation of the control process for various types of unmanned vehicles, including drones. It provides access to a wide range of functionalities such as UAV mission planning, autonomous takeoff and landing, GPS waypoint navigation, and more. Thanks to ArduPilot, critical points of the system and the possibility of its further physical implementation can be easily assessed.

ArduPilot SITL also allows to monitor the position of a simulated quadcopter on an interactive map with satellite images, which comes very handy when looking on the actual navigation path in the cities.

Figure 2: ArduPilot SITL Map view with multiple simulated quadcopters [16].

3.3. Route building subprogram

The code blocks below show the final path finding code which was developed during the research work. It utilizes Q-Learning – a popular approach of reinforced learning that allows an agent to effectively learn efficient routes based on the given transportation costs in the graph. By tuning the hyperparameters the code was optimized to work reliably on any given set of waypoints.

The two main functions of a Q-Learning agent are the ones responsible for choosing an action and learning the consequences of executing a chosen action. Depending on a random choice and the current exploration probability (which decreases after each learning epoch) the agent will either choose a random action available from the current state, or utilize an action which brought the most reward during past iterations. On the each subsequent episode it will less likely explore the new moves and will instead exploit the collected route building data that is stored in his Q-table.

```
def choose_action(self, state):
    if np.random.uniform(0, 1) < self.exploration_prob:
        return np.random.choice(self.num_actions)  # Explore
    else:
        return np.argmax(self.q_table[state, :])  # Exploit
```

In order to remember the efficiency of all the action combinations, the agent is updating its Q-Table after taking each action by comparing the predicted outcome based on the past runs with the real reward obtained after the latest move.

```
def learn(self, state, action, reward, next_state):
    predict = self.q_table[state, action]
    target = reward + self.discount_factor *
        np.max(self.q_table[next_state, :])
    self.q_table[state, action] += self.learning_rate *
        (target - predict)
```

The overall learning process consists of moving across the graph and calculating the total cost of a route, repeating until a given amount of episodes (epochs) is not completed. By collecting the different rewards the agent is able to successfully learn the valid behavior that is leading him to maximum reward, thus finding the most optimal route.

```
# Make agent learn the graph for given episodes count
for episode in range(max_episodes):
    state = start
    total_reward = 0
    visited_nodes = []

    # While all necessary nodes are not visited
    while len(visited_nodes) != len(nodes_to_visit):
```

```
        # Choose a random move or exploit known data
        action = agent.choose_action(state)
        next_state = action

        # Add a negative reward for revisiting the same waypoint
        if action == state:
            reward = -100
        else:
            # Negative cost for shortest path
            reward = -graph[state, action]

            # Slightly increase the reward when agent visits a route he
            # had to visit and didn't visit yet
            if next_state in nodes_to_visit and
                not (next_state in visited_nodes):
                reward *= 0.001
                visited_nodes.append(next_state)
            elif next_state in visited_nodes:
               # Add a negative reward for revisiting the same waypoint
                reward -= 100

        agent.learn(state, action, reward, next_state)
        state = next_state
        total_reward += reward

    # Decay exploration probability
    agent.exploration_prob *= agent.exploration_decay
```

After the agent is done learning it is building the final path once again, which will eventually be the most efficient one, based on the pre-set hyperparameters and reward calculation logic.

```
# Graph array represents the costs for traveling from
# node A to B (graph[A][B])
graph = np.array([
    [0, 50, 20, 30, 40],
    [50, 0, 10, 30, 80],
    [20, 10, 0, 40, 10],
    [30, 30, 40, 0, 20],
    [40, 80, 10, 20, 0]
])

start = 0
nodes_to_visit = [4, 2] # Destination routes
visited_nodes = []
```

```
path = [start]
while len(visited_nodes) != len(nodes_to_visit):
    # Choose new actions until all nodes_to_visit are visited
    action = agent.choose_action(path[-1])
    if action in nodes_to_visit:
        visited_nodes.append(action)
    path.append(action)
print("Shortest Path:", path)
```

The example graph used in the code is assuming each waypoint is accessible from any other waypoint and the travel cost is the same when moving in both directions. In reality the departments graph could be dynamically generated by modifying the costs regarding the weather and wind directions, thus also optimizing the route built for the real world conditions. The agent itself could also utilize the drones battery level, maximum travel distance left, total cargo capacity and multi-package delivery optimizations in his reward system to even better improve the UAV path for maximum productivity.

4. Conclusion

The research allowed us to examine the overall aspects of drone delivery system creation. After analyzing the existing commercial systems and research in the area we were able to determine the crucial aspects of such systems and develop the necessary architecture. The key focus

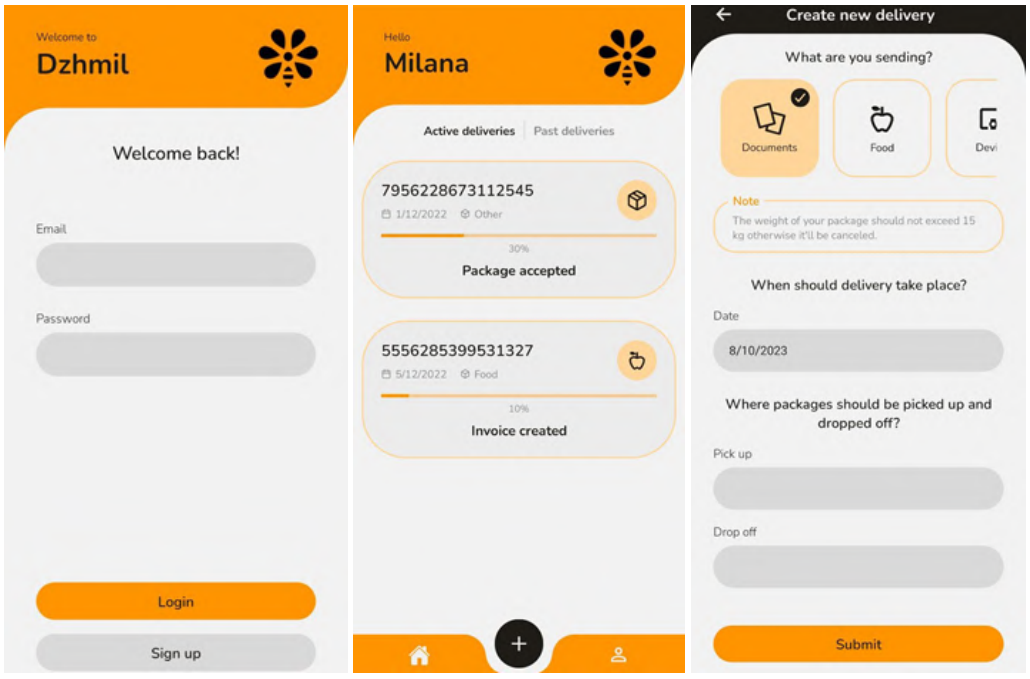

Figure 3: Examples of the user interface of the developed client application.

was given to the route planning algorithm that should create a valid path between start and destination location through a given set on departments (waypoints).

A backend and mobile applications were developed to be used by both regular users and delivery managers. The developed mobile app screenshots are displayed in figure 3.

In order to properly test all the system aspects, the drone flights were simulated in the ArduPilot SITL environment. This will also allow to directly apply the developed code to the UAVs running ArduPilot flight controller firmware.

Speaking of further development, the reinforced learning agent used for path finding can be improved by including different aircraft sensor data and environment conditions into the calculations. This will allow to embed such aspects as the payload weight, battery level or wind speed before the flight or even during the flight itself to better navigate the drone through the area.

The developed system had shown itself as a well working prototype that is easy to adapt and scale according to the desired conditions and requirements.

Acknowledgments

We acknowledge the contribution of ChatGPT to the refinement of this paper. ChatGPT's assistance in language enhancement and phrase generation significantly contributed to the quality of the final manuscript. It's imperative to highlight that the responsibility for reviewing and aligning the generated content with the narrative of our manuscript solely rests with the authors. We ensured that all content generated by AI tools, particularly regarding well-known concepts or definitions, underwent meticulous scrutiny to verify accuracy and relevance. Proper references to the original content were included to maintain academic integrity and acknowledge the intellectual contributions of others.

References

[1] A. R. Petrosian, R. V. Petrosyan, I. A. Pilkevych, M. S. Graf, Efficient model of PID controller of unmanned aerial vehicle, Journal of Edge Computing 2 (2023) 104–124. doi:10.55056/jec.593.

[2] Amazon Prime Air prepares for drone deliveries, 2022. URL: https://www.aboutamazon.com/news/transportation/amazon-prime-air-prepares-for-drone-deliveries.

[3] Starship Technologies: Autonomous robot delivery, 2023. URL: https://www.starship.xyz.

[4] Zipline Instant Delivery & Logistics, 2023. URL: https://www.flyzipline.com.

[5] N. Michel, Z. Kong, X. Lin, Optimal Control of a Multirotor Unmanned Aerial Vehicle Based on a Multiphysical Model, volume Volume 2: Intelligent Transportation/Vehicles; Manufacturing; Mechatronics; Engine/After-Treatment Systems; Soft Actuators/Manipulators; Modeling/Validation; Motion/Vibration Control Applications; Multi-Agent/Networked Systems; Path Planning/Motion Control; Renewable/Smart Energy Systems; Security/Privacy of Cyber-Physical Systems; Sensors/Actuators; Tracking Control Systems; Unmanned Ground/Aerial Vehicles; Vehicle Dynamics, Estimation, Control; Vibration/Control Sys-

tems; Vibrations of *Dynamic Systems and Control Conference*, 2020, p. V002T36A004. doi:10.1115/DSCC2020-3239.

[6] G. Mehrooz, E. Ebeid, P. Schneider-Kamp, System Design of an Open-Source Cloud-Based Framework for Internet of Drones Application, in: 2019 22nd Euromicro Conference on Digital System Design (DSD), 2019, pp. 572–579. doi:10.1109/DSD.2019.00087.

[7] H. Jahanshahi, A. Bozanta, M. Cevik, E. M. Kavuk, A. Tosun, S. B. Sonuc, B. Kosucu, A. Başar, A deep reinforcement learning approach for the meal delivery problem, Knowledge-Based Systems 243 (2022) 108489. doi:10.1016/j.knosys.2022.108489.

[8] G. Aiello, R. Inguanta, G. D'Angelo, M. Venticinque, Energy Consumption Model of Aerial Urban Logistic Infrastructures, Energies 14 (2021) 5998. doi:10.3390/en14185998.

[9] H. Huang, A. V. Savkin, C. Huang, Drone Routing in a Time-Dependent Network: Toward Low-Cost and Large-Range Parcel Delivery, IEEE Transactions on Industrial Informatics 17 (2021) 1526–1534. doi:10.1109/TII.2020.3012162.

[10] T. Lorido-Botran, M. K. Bhatti, ImpalaE: Towards an optimal policy for efficient resource management at the edge, Journal of Edge Computing 1 (2022) 43–54. doi:10.55056/jec.572.

[11] N. M. Lobanchykova, I. A. Pilkevych, O. Korchenko, Analysis and protection of iot systems: Edge computing and decentralized decision-making, Journal of Edge Computing 1 (2022) 55–67. doi:10.55056/jec.573.

[12] O. V. Klochko, V. M. Fedorets, M. V. Mazur, Y. P. Liulko, An IoT system based on open APIs and geolocation for human health data analysis, CTE Workshop Proceedings 10 (2023) 399–413. doi:10.55056/cte.567.

[13] Y. B. Shapovalov, Z. I. Bilyk, S. A. Usenko, V. B. Shapovalov, K. H. Postova, S. O. Zhadan, P. D. Antonenko, Harnessing personal smart tools for enhanced STEM education: exploring IoT integration, Educational Technology Quarterly 2023 (2023) 210–232. doi:10.55056/etq.604.

[14] A. Sharma, P. Vanjani, N. Paliwal, C. M. Basnayaka, D. N. K. Jayakody, H.-C. Wang, P. Muthuchidambaranathan, Communication and networking technologies for UAVs: A survey, Journal of Network and Computer Applications 168 (2020) 102739. doi:10.1016/j.jnca.2020.102739.

[15] W. Chen, Y. Dong, Z. Duan, DPM: Towards Accurate Drone Position Manipulation, IEEE Transactions on Dependable and Secure Computing 20 (2023) 813–826. doi:10.1109/TDSC.2022.3144319.

[16] Multiple Vehicles with MAVProxy, 2023. URL: https://ardupilot.org/mavproxy/docs/getting_started/multi.html.