Selection of Critical Nodes in Drone airways Graphs via Graph Neural Networks

Igone Morais-Quilez

Computational Intelligence Group, University of the Basque Country (UPV-EHU), Donostia-San Sebastian, Spain
igone.morais@ehu.eus

Abstract. This Master Thesis has two distinct parts. The first one models an application of Graph Neural Networks (GNN) for the identification of critical nodes in graphs that correspond to traffic networks. We call critical nodes those that can compromise the traffic flow in some subgraphs of the network. Specifically, the example data for the demonstration corresponds to the Vienna subway network, hence the linear subgraphs correspond to the subway lines with intersections at some key subway stations. Those critical nodes relative to a subway line compromise the traffic flow at this line, therefore, we propose three GNN based approaches for the identification of such critical nodes, reporting encouraging results. The second part of the Master Thesis illustrates the background research work on drone airspace management and a discussion of how the reported results may have some relevance for this emerging difficult problem. The main idea is that the urban airspace for drones, that may be carrying out delivery of either persons (aerotaxis) or goods, can be structured along airways that mimic the existing network of streets. The computational example explored in part one of the Master Thesis, thus, becomes relevant for the development of intelligent drone airspace management.

Keywords: Air traffic control · Aircraft navigation · Intelligent robots · Critical node identification · Neural networks · Pattern recognition.

1 Introduction

Graph neural networks Graph neural networks (GNN) have been proposed in several flavors aiming to cope with the processing of highly unstructured data that are better represented by a graph [38]. GNNs have been successfully applied to tackle problems in computer vision, natural language processing, traffic forecasting and control, recommender systems, and chemistry. Proposals for GNN architectures have been derived from diverse research approaches, such as convolutional based constructions, recursive neural networks, autoencoders, and spatio-temporal architectures dealing with time varying graph structures. In this Master Thesis we report computational experiments dealing with the identification of critical nodes in graph representations of traffic networks. Critical nodes are defined as the nodes that can severely influence the traffic flow over specific subgraphs.

Drones Applications of drones have augmented in the last decade, and their uses have increased in number. Recently, the availability of relatively cheap hardware has aroused the interest for aerial swarms where several flying robots collaborate to achieve a collective task [11], [29]. Multi-drone systems are proposed for a broad spectrum of missions including search and rescue [4], long-term monitoring [40], sensor data collection [34], indoor navigation, environment exploration [21], and cooperative grasping and transportation [22]. In the entertainment industry, some spectacular demonstrations of air effects have been achieved the coordination of fleets of hundreds of drones that light up the night sky with aerial shows, as **Intel** and **Ehang** have displayed.

Nonetheless, the initial point towards the deployment of such complex systems in real-world scenarios is simulation [31]. The development of algorithms and applications for autonomous aerial vehicles requires the availability of a suitable simulation framework for rapid prototyping and simulation in reproducible scenarios. This is desirable in all robotics fields, but it is especially relevant for collective systems such as drone swarms, where errors can propagate through the individuals and lead to catastrophic result. Moreover, having a tool that allows the user to monitor that propagation and the path planning will give the user a general overview of the current situation to plan ahead.

Long term motivation The work in the Master Thesis has been motivated by the involvement of the student in an Elkartek project related to the design of innovative air transport autonomous vehicles, i.e. drones for the transportation of goods and people. Specifically, the student was involved in the study of air traffic management, in order to have an appropriate distribution of multiple aircrafts in the shared airspace [5] as proposed by the European Commission due to the need of a separated space for Unmaned Aerial vehicles (UAV) and other aerial vehicles, such as airliners. The computational experimental work reported in this Master Thesis may be of future use in the control of urban air spaces densely filled with drones carrying out independent but similar tasks, such as deliveries of goods.

Contents of this Master Thesis: Section 2 provides an introductory review of Graph Neural Networks (GNN) with motivation and some formal definitions. Section 3 gives the definition of the target problem. Section 4 gives a short description of the data used for the computational experiments. Section 5 contains the data preprocessing carried out. Section 6 describes the three architectures that have been applied to the example graph data. Section 7 reports the results of the proposed computational methods obtained over the experimental data. Section 8 provides a view of the broad paradigm where the specific work carried out in the Master Thesis is embedded. Section 9 briefly discusses a systematic approach to create experimental settings. Section 10 gives our conclusions and some ideas for future work.

2 Graph Neural Networks: Motivation, definitions and Applications

A graph data structure consists of a finite (and possibly mutable) set of vertices (also called nodes or points), together with a set of unordered pairs of vertices, for undirected graphs, or a set of ordered pairs, for directed graphs. These pairs of vertices are known as edges (also called links or lines). For a directed graph edges can also be called arrows or arcs. The vertices may be part of the graph data structure, or may be external entities represented by integer indices or references. A graph data structure may also associate to each edge some value, such as a symbolic label or a numeric attribute (cost, capacity, length, etc.). Recently, research about analyzing graphs with machine learning is receiving more and more attention because of the great expressive power of graphs. As a unique non-Euclidean data structure for machine learning, graph analysis focuses on tasks such as node classification, link prediction, and clustering.

Many tasks require dealing with graph data containing valuable relational information among elements. Some domains demand models that are able to learn from graph input information (such as modeling physics systems, learning molecular fingerprints, predicting protein interface, and classifying diseases), whereas others require learning from non-structural data in order to reasoning on extracted graph, such texts and images, and reasoning on structures extracted from signal data, like the dependency trees of sentences or the scene graphs describing the content of images.

Graph neural networks (GNNs) are deep learning based methods that operate on graph domains. Due to their convincing performance, they have become a widely applied graph analysis method recently. Deep neural networks (DNN), specifically convolutional neural networks (CNN) have been widely used in many computer vision and pattern recognition tasks. Following this success, some approaches have been proposed to generalize the convolution operation over graph data. Overall, these methods can be categorized into spatial convolution and spectral convolution approaches. Spatial approaches define the graph convolution operation directly as an operation over node neighborhoods.

For example, Duvenaud et al. [10] propose a convolutional neural network that operates directly on graphs and provide an end-to-end feature learning for graph data. Atwood and Towsley [1] propose Diffusion-Convolutional Neural Networks (DCNNs) by employing a graph diffusion process to incorporate the contextual information of node in graph node classification. Monti et al. [23] present mixture model CNNs (MoNet) and provide a unified generalization of CNN architectures on graphs. By designing an attention layer, Velickovic et al. [6] present Graph Attention Networks (GAT) for semi-supervised learning.

On the other hand, spectral methods define the graph convolution operation based on spectral representation of graphs. For example, [Bruna et al.] [7] define the graph convolution in the Fourier domain based on the eigen-decomposition of the graph Laplacian matrix. Defferrard et al. [9] propose to approximate the spectral filters based on Chebyshev expansion of graph Laplacian to avoid the

high computational complexity of eigen-decomposition. Kipf et al. [16] propose a Graph Convolutional Network (GCN) for semi-supervised learning.

2.1 General design pipeline of GNNs

A graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\left|\mathcal{V}\right| = \mathcal{N}$ is the number of nodes in the graph and $\left|\mathcal{E}\right| = \mathcal{N}^{\urcorner}$ is the number of edges. Matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ with non zero entries corresponding to (un)directed edges is the adjacency matrix. For graph representation learning description, $\mathbf{h}_v$ and $\mathbf{o}_v$ denote the hidden state and output vector of node v.

As the first step in the design pipeline, it is important to find out the graph structure in the application. There are two scenarios:

1. **structural scenarios**: the graph structure is explicit, such as in applications on molecules, physical systems, knowledge graphs, and so on.
2. **non-structural scenarios**: graphs are implicit so that we have first to extract the graph from the raw data, such as building a fully-connected "word" graph for text or building a scene graph for an image.

After the graph is obtained, the ensuing design process steps attempt to find an optimal GNN model to work on this specific graph. Therefore, we have to find out the graph type and its scale. Graphs with complex types could provide more information on nodes and their connections. Graphs are usually categorized as:

- Directed/Undirected Graphs. Edges in directed graphs are all directed from one node to another, providing more information than undirected graphs. Each edge in undirected graphs can also be regarded as the collection of two directed edges.
- Homogeneous/Heterogeneous Graphs. Nodes and edges in homogeneous graphs have the same types, while nodes and edges have different types in heterogeneous graphs. Typification of nodes and edges play important roles in heterogeneous graphs and should be further considered.
- Static/Dynamic Graphs. When input features or the topology of the graph vary with time, the graph is regarded as a dynamic graph. The time information should be carefully considered in dynamic graphs.

Note these categories are orthogonal, which means these types can be combined, e.g. one can deal with a dynamic directed heterogeneous graph. There are also several other graph types designed for different tasks such as hypergraphs and signed graphs. We will not enumerate all types here but the most important idea is to consider the additional information provided by these graphs. Once we specify the graph type, the additional information provided by these graph types should be further considered in the design process.

Regarding graph scale, there is no clear classification criterion for "small" and "large" graphs. The criterion is still changing with the development of computation devices (e.g. the speed and memory of GPUs). When the adjacency matrix or the graph Laplacian (space complexity is quadratic) cannot be stored

and processed by the device, then we consider the graph as a large-scale graph, requiring decomposition or sampling methods for its processing.

For graph learning tasks, there are usually three kinds of tasks:

- Node-level tasks focus on nodes, which include node classification, node regression, node clustering, etc. Node classification tries to categorize nodes into several classes, and node regression predicts a continuous value for each node. Node clustering aims to partition the nodes into several disjoint groups, where similar nodes should be in the same group.
- Edge-level tasks are edge classification and link prediction, which require the model to classify edge types or predict whether there is an edge existing between two given nodes.
- Graph-level tasks include graph classification, graph regression, and graph matching, all of which need the model to learn graph representations.

From the perspective of supervision, we can also categorize graph learning tasks into three different training settings:

- Supervised setting requires labeled data for training.
- Semi-supervised setting uses a small amount of labeled nodes and a large amount of unlabeled nodes for training. In the test phase, the transductive module predicts the labels of the given unlabeled nodes, while the inductive module provides new unlabeled nodes from the same distribution to enter the inference computations. Most node and edge classification tasks are semi-supervised. Most recently, a mixed transductive-inductive scheme is undertaken by Wang and Leskovec (2020) [36] and Rossi et al. (2018), creating a new path towards a mixed setting.
- Unsupervised setting processes only unlabeled data aiming for the model to find patterns (clusters) in data. Node clustering is a typical unsupervised learning task.

Knowing the task type and the training setting, we can design a specific loss function for the task. For example, for a node-level semi-supervised classification task, the cross-entropy loss can be used for the labeled nodes in the training set.

A typical GNN model is usually built by combining the following operators: The convolutional operator, recurrent operator, sampling module and skip connection are used to process and propagate information in each layer. The pooling module is added to extract high-level information by selective compression of the data. These layers are usually stacked to obtain a hierarchy of representations. Note this architecture can generalize most GNN models with some exceptions. Figure 1 summarizes this design pipeline.

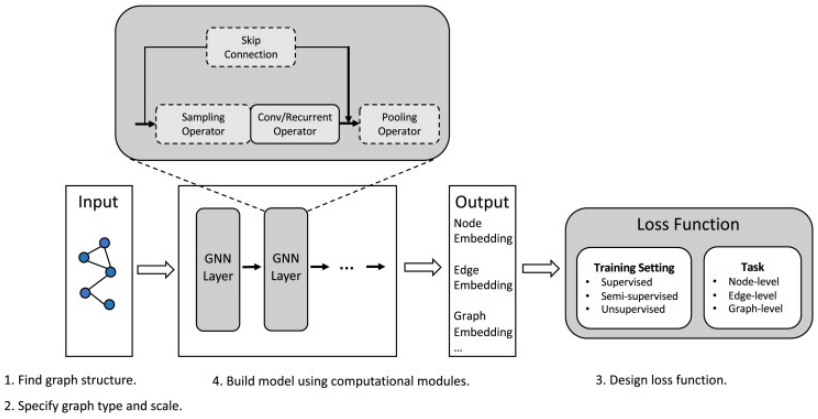


Fig. 1: The general design pipeline for a GNN

2.2 Traffic prediction with advanced graph neural networks

Safe transit and routing problem.[Nogami et al. 1996] [24] is a typical large-scale scheduling problem that has the following characteristics:

1. Immediate decision making at any discrete decision point is needed, so that real operation (traffic flow) is not delayed.
2. The environment surrounding this decision making process change dynamically with time, and involve various uncertain elements, some are governed by stochastic processes and some others are difficult to formulate mathematically.
3. Optimization criteria cannot be obtained until large unspecified number of successive decision timings occur, requiring the long-term anticipation of the discrete events expected to occur throughout the planning horizon.
4. Various strategies to avoid the so-called combinatorial explosion are needed to deal with a great number of constraints
5. Negotiation and regulation among multiple agents, which have individual requirements conflicting with each other are needed

Traffic prediction in road networks has been recently tackled with GNNs approaches by the well known **DeepMind** branch of Google. The initial proof of concept began with a straight-forward approach that used the existing traffic system as much as possible, specifically the existing segmentation of road networks and the associated real-time data pipeline. This meant defining Supersegments covering a set of road segments, where each segment has a specific length and corresponding speed features. A first approach consisted on training a single fully connected neural network model for every Supersegment. Promising initial results demonstrated the potential in using neural networks for predicting travel time. However, given the dynamic sizes of the Supersegments, a separately trained neural network model is required for each one. To deploy this approach

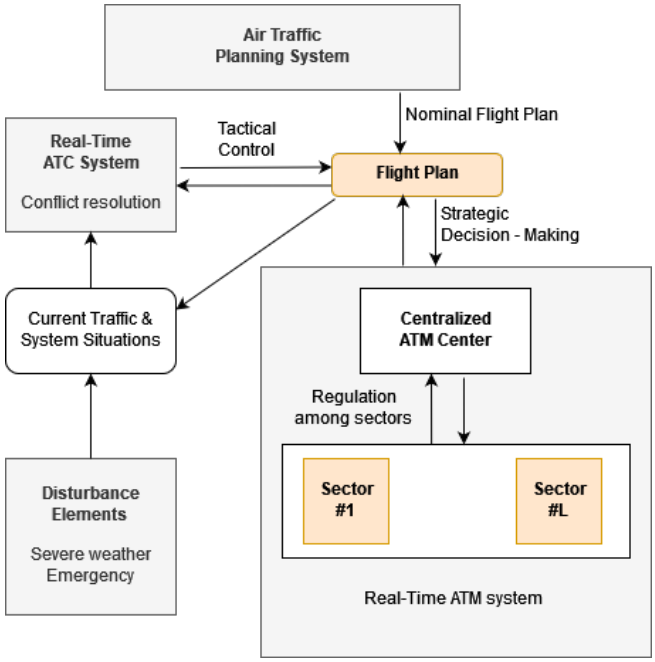


Fig. 2: The general software design pipeline for the Air Traffic Management (ATM) problem

at scale implies training millions of these models, which poses a considerable infrastructure challenge. This lead DeepMind researchers to look into models that could handle variable length sequences, namely Recurrent Neural Networks (RNNs). However, incorporating further structure from the road network proved difficult. In modeling traffic, it is important how cars flow through a network of roads, and GNNs are able to model network dynamics and information propagation.

In the end, DeepMind's model treats the local road network as a graph, where each route segment corresponds to a node and edges exist between segments that are consecutive on the same road or connected through an intersection. In a Graph Neural Network, a message passing algorithm is executed where the messages and their effect on edge and node states are learned by neural networks. From this viewpoint, the Supersegments are road subgraphs, which were sampled at random in proportion to traffic density, as we can see in Fig. 3 . A single model can therefore be trained using these sampled subgraphs, and can be deployed at scale.

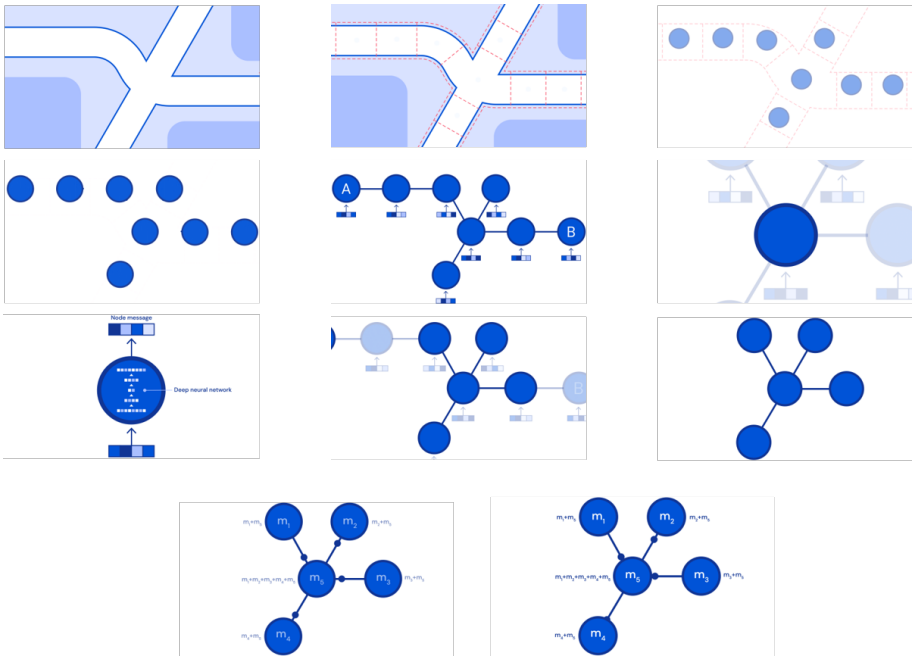


Fig. 3: Deepmind approach to road network traffic modeling and prediction

3 Target problem

We focus on analyzing the transportation networks where we can observe intersections as nodes, and path segments as edges connecting them. These edges can be enriched with additional attributes such as the path length, current or historical speeds along their segment and the like, but in this work we will work with attribute-less edges. Our problem consists on identifying the most critical nodes: for a given candidate route over the transportation graph, we want to identify the critical nodes that must be avoided for navigation maintaining the estimated time of arrival. Critical nodes of the transportation network are those whose congestion that may induce great delays in the overall traffic flow or where there is high risk of multiple collisions [30] [26] [39]. GNNs have shown immense promise for this task, and can be used to directly predict the critical nodes for a relevant subgraph of the network.

4 Data

The contents of the dataset that has been used for the computational experiments reported below have been obtained from **Kaggle**. It consists in the graph corresponding to the Vienna subway network. The analysis of the path network is done by expecting that every path has a start and a stop, and every path

can be considered as separated from the others, like in the usual subway lines, where each start and stop is a subway station. Since we will be considering the network as graph, the real use of the network is dispensed given this representation serves to analyze both subways networks, airways or road networks. For the Vienna subway network we have the following information: Start station, Stop station, Line and Color, as it is represented in the Table 1, representing the segments of a line as shown in Table 2. The next step will be to represent the information of the table as a graph as shown in Fig. 4.

Table 1: Vienna subway network dataset raw information

Start (Station)	Stop (Station)	Line	Color
Oberlaa	Neulaa	1	Red
Neulaa	Alaudagasse	1	Red
Alaudagasse	Altes Landgut	1	Red
Altes Landgut	Troststrasse	1	Red
Troststrasse	Reumannplatz	1	Red
...	...	...	...

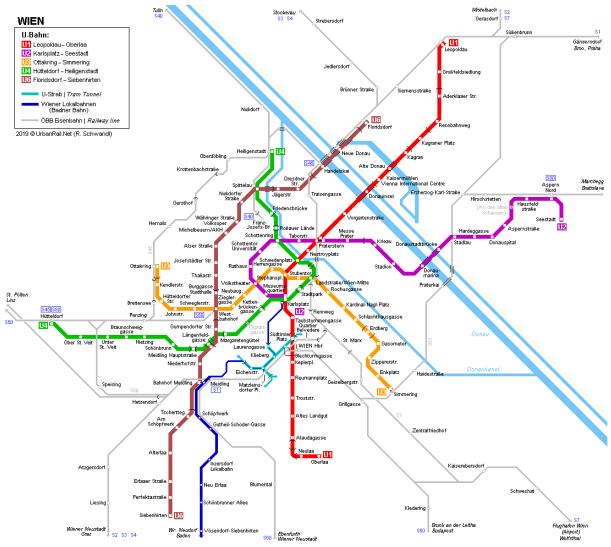
Table 2: Vienna subway network dataset raw information

Start (Station)	Stop (Station)	Line	Color
Oberlaa	Leopoldau	1	Red
Karlsplatz	Seestadt	2	Purple
Simmering	Ottaking	3	Orange
Hüttledorf	Heiligenstadt	4	Green
Floridsdorf	Slebenhirten	6	Brown

5 Data preprocessing

Machine learning is recognized as an effective method by which some novel and evolutionary knowledge is acquired and generalized systematically. The following Unsupervised Learning machine learning techniques and well-known graph embedding algorithms have been applied for network data preprocessing whose results are shown in Fig. 6.

PageRank(PR): It is a link analysis algorithm that assigns a numerical weighting to each element of a hyperlinked set of documents, with the purpose of measuring its relative importance within the set. The numerical weight that it assigns to any given element E is referred to as the PageRank of E and denoted by PR(E). The rank value indicates an importance of a particular page.



(a) Subway map network

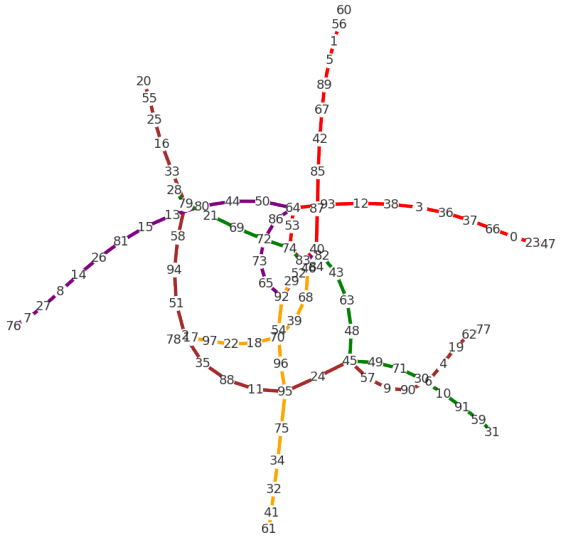
(b) Subway network graph

Fig. 4: (a) Map network of the Vienna subway train (b) The Vienna subway network visualized as a graph network

A hyperlink to a page counts as a vote of support. The PageRank of a page is defined recursively and depends on the number and PageRank metric of all pages linked to it ("incoming links"). A page that is linked to by many pages with high PageRank receives a high rank itself. In a similar way, we could apply this algorithm to find the relevance of a node in a graph set. Overlooking the graph we could say that a critical node would be the point in the graph several lines intersect. By obtaining the page ranking, we have a ranking of the most influential nodes to later on select critical nodes to each line

DeepWalk [KDD 2014]: It uses a randomized path traversing technique to provide insights into localized structures within networks. It does so by utilizing these random paths as sequences, that are then used to train a Skip-Gram Language Model. We apply this algorithm to calculate the relation between the lines, to analyze the dependencies between the lines trying to locate the most important lines.

Node2Vec: algorithmic framework for representational learning on graphs. Given any graph, Node2Vec can learn continuous feature representations for the nodes, which can then be used for various downstream machine learning tasks. We apply this algorithm to calculate the transition probabilities between nodes



(a) Network labels

Fig. 5: (a) Vienna subway network with the numerical labels corresponding to nodes.

6 Computational methods

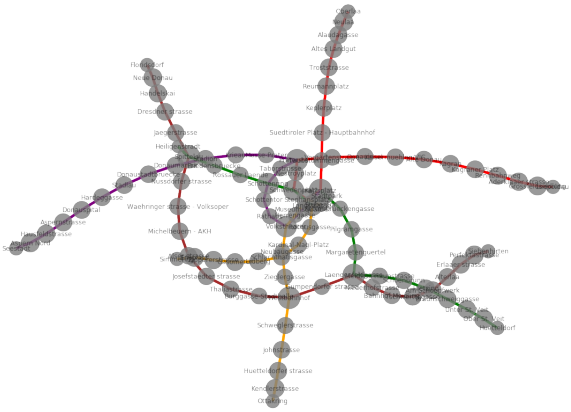
This work started trying to reproduce the results of a previous **published analysis** published analysis over a metro network whose graph representation is relatively small and manageable. We have extended this analysis proposing two additional GNN architectures.

6.1 Semi-supervised Learning with Graph Convolutional Network:

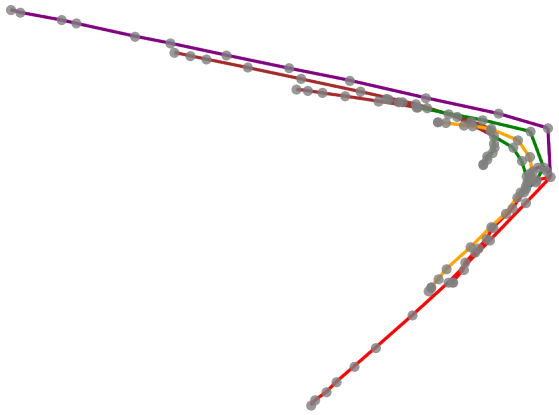
Deep neural networks have been long-established for computer vision and pattern recognition tasks. Here is considered the problem of classifying nodes in a graph, where labels are only available for a small subset of nodes. This problem can be framed as graph-based semi-supervised learning, where label information is smoothed over the graph via graph-based regularization by using a graph Laplacian regularization term in the loss function [17]:

$$\mathcal{L} = \mathcal{L}_0 + \lambda \mathcal{L}_{reg}, \text{with } \mathcal{L}_{reg} = \sum_{i,j} A_{ij} \ \|f(X_i) - f(X_j)\|^2 = \ f(X)^\top \Delta f(X) \quad (1)$$

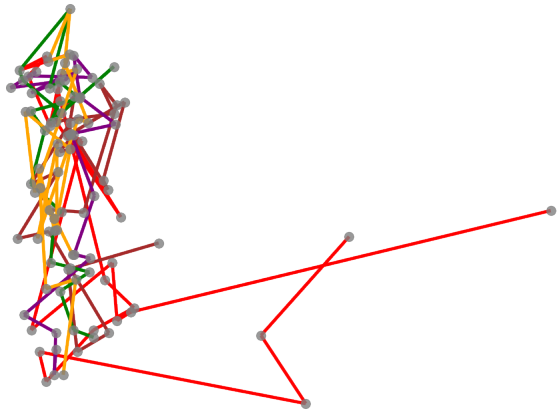
Here, $\mathcal{L}_0$ denotes the supervised loss w.r.t. the labeled part of the graph, $f(\cdot)$ can be a neural network-like differentiable function,λ is a weighing factor and



(a) PageRank analysis results



(b) DeepWalk analysis results



(c) Node2Vec analysis results

Fig. 6: Vienna subway network node labeling by unsupervised approaches (a) Map network of the Vienna subway train indicating the most relevant nodes by PageRank (b) Representation of the relation between lines in the Vienna subway network (c) Representation of the transition probabilities between the stations in the Vienna subway network

X is a matrix of node feature vectors X_i. $\Delta = D - A$ denotes the unnormalized graph Laplacian of an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with N nodes $v_i \in \mathcal{V}$, edges $(v_i, v_j)\mathcal{E}$, an adjacency matrix $A \in \mathbb{R}^{N \times N}$ (binary or weighted) and a degree matrix $D_{ii} = \sum_j A_{ij}$. The formulation of Eq1 relies on the assumption that connected nodes in the graph are likely to share the same label. This assumption, however, might restrict modeling capacity, as graph edges need not necessarily encode node similarity, but could contain additional information. This way, we can encode the graph structure directly using a neural network model $f(X, A)$ and train on a supervised target $\mathcal{L}_0$ for all nodes with labels, thereby avoiding explicit graph-based regularization in the loss function. Conditioning $f()$ on the adjacency matrix of the graph will allow the model to distribute gradient information from the supervised loss $\mathcal{L}_0$ and will enable it to learn representations of nodes both with and without labels.

Given that our target is, given a graph with incomplete node labelling, predict the class of the remaining nodes, being the class relevant or no relevant. For our problem, the relevancy will be translated as critic or non critic. Each node has unique id that will be the number of the station, and a representative nodes for each color will be chosen as follows: Red(60, 47, 38), Brown(20,77,35), Purple(76,80), Green(31,69), Orange(78,61), that correspond to the first station of the line (the ends of the graph), the end station of the line, and in some cases when the line is to long, an intermediate station is chosen (near a point of many intersections) Fig. 5(a).

Viewing this problem as a semi-supervised node classification, we can relax certain assumptions typically made in graph-based semi-supervised learning by conditioning our model $f(X, A)$ both on the data X and on the adjacency matrix A of the underlying graph structure.

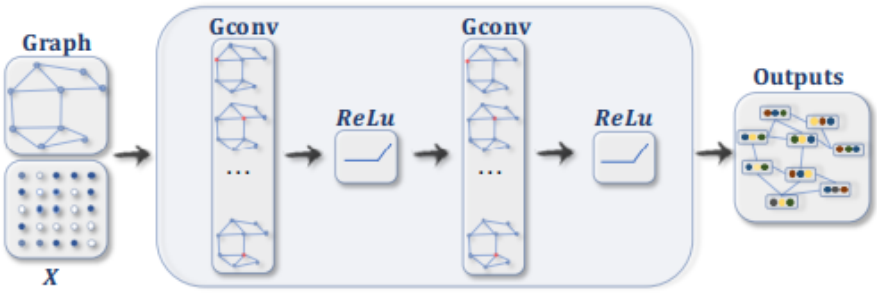


Fig. 7: Architecture of the proposed basic GCN

6.2 Enhanced GCN

The second approach is to enhance a GCN taking into account the following guidelines [15]:

1. Let $X = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^{n \times p}$ be the collection of n data vector in p dimensions.
2. Let $G(X, A)$ be the graph representation of X with $A \in \mathbb{R}^{n \times n}$ encoding pair wise relationship (similarities between neighbours) among data X.
3. The GCN contains one input layer, several propagation (hidden) layers and one final perceptron layer.
4. Given an input $X^{(0)=X}$ and a graph A, GCN conducts the following layer-wise propagation show in Eq2

$$X^{(k+1)} = \sigma(D^{-1/2}AD^{-1/2}X^{(k)}W^{(k)}), \tag{2}$$

where, $k = 0, 1, \ldots, K-1$ and $D = diag(d_1, d_2, \ldots, d_n)$ is a diagonal matrix with $d_i = \sum_{j=1}^{n} A_{ij}$. $W^{(k)} \in \mathbb{R}^{d_k \times d_{k+1}}$, $d_0 = p$ is a layer-specific weight matrix needing to be trained. $\sigma(\cdot)$ denotes an activation function, in this case we used $ReLU(\cdot) = max(0, \cdot)$, and $X^{k+1} \in \mathbb{R}^{n \times d_{k+1}}$ denotes the output of activations in the k-th layer.

For semi-supervised classification, GCN includes a final perceptron layer defined as

$$Z = softmax(D^{-1/2}AD^{-1/2}X^{(K)}W^{(K)}) \tag{3}$$

where $W^K \in \mathbb{R}^{d_K \times c}$, and c denotes the number of classes. The final output $Z \in \ltimes^{n \times c}$ denotes the label prediction for all data X in which each row Z_i denotes the label prediction for the i-th node. The optimal weight matrices $W^{(0)}, W^{(1)}, \ldots, W^K$ are trained by minimizing the cross-entropy loss function defined as follows,

$$\mathcal{L}_{Semi-GCN} = -\sum_{i \in L} \sum_{j=1}^{c} Y_{ij} ln Z_{ij} \tag{4}$$

where L indicates the set of labeled nodes.

The architecture of the proposed enhanced GCN shown in Fig. 8 [38], consists of 3 hidden layers topped with a softmax layer and loss, a size of the hidden layer of 16, 0.3 of dropout, , weight decay of 5e-4, a learning rate of 0.01. This architecture training time is set to 200 epochs.

6.3 Third approach

Finally the third approach is a more advanced GCN to improve the results of the second approach, which includes a GCN layer, that will contain a graph convolution layer and for each hidden layer another graph convolution layer, with a MLP layer, that will contain a dropout layer and linear regressor. There would be 16 graph convolutions and 8 mlps, as shown in Fig. 9 [38]. It will be trained for 100 epochs.

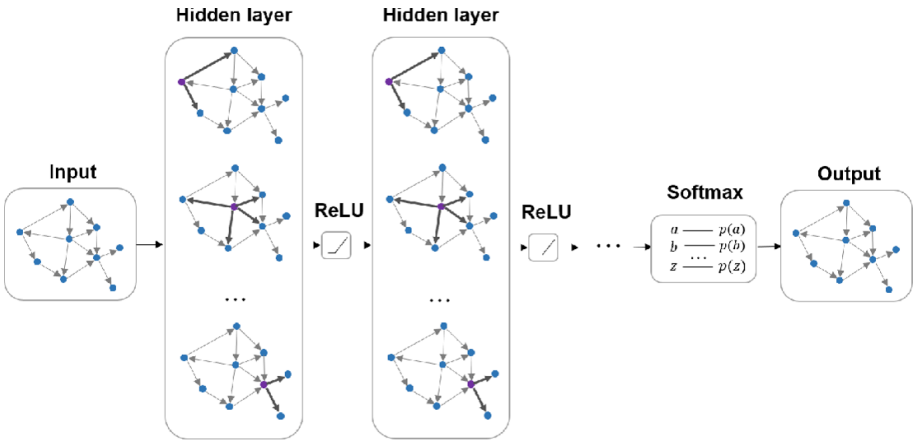


Fig. 8: Architecture of the proposed enhanced GCN

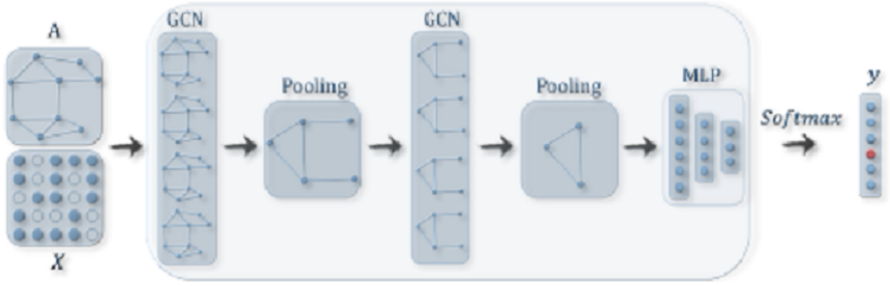


Fig. 9: Architecture of the proposed Advanced GCN

7 Results

The models proposed in the Section 6 have been simplified for a better adaptation to the functions of PyTorch libraries, as it happened with the loss function, for example.

7.1 First aproach results

First, we apply the basic graph convolutional network composed of two layers, a two-layer GCN for semi-supervised node classification on a graph with a symmetric adjacency matrix A (binary or weighted), Fig. 7 [38], that consist of a graph convolution layer, a ReLu layer and an up graph convolution and 100 epochs. We want to predict which node is most relevant to which line, classify a node with the color of the line they might create a block on. In the visual results shown in Fig. 10(b), for instance, a brown node on a purple line would indicate that this node is critical for the brown line.

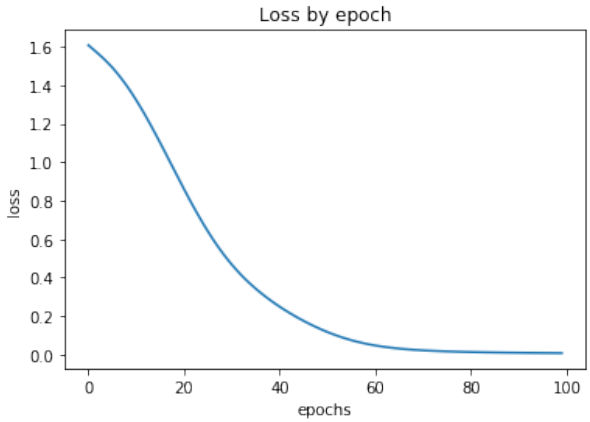
More specifically, the biggest nodes, representing the most relevant nodes for each line, is influential to the line they represent. Moreover, we can also see that a node influences the line they belong to, even though, a big percentage of the nodes influence the purple line, a lesser percentage influence the brown one, and a very few of them the red line. This test has obtained an accuracy of 52.01%

7.2 Second approach results

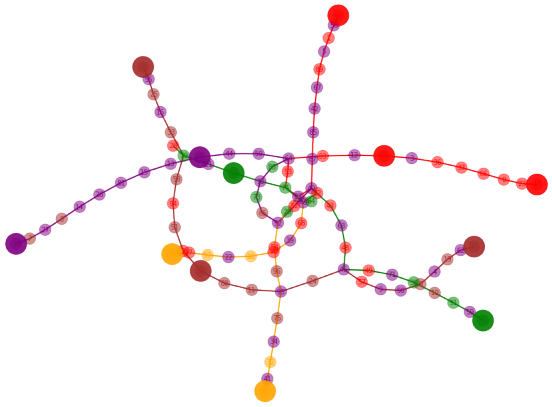
The Fig. 11 shows us that still the representative nodes influence their own line as expected, but mostly a big percentage of the rest of the nodes influence the green line, and to a lesser extent, the purple line, which we can see are the very own nodes for that line. A very few of them influence the brown line, being also the ones for that line, and the same applies in the case for the nodes influencing the orange and red lines. In this case, we obtained a bigger accuracy for the test, being 60.12%

7.3 Third approach results

In this case the Fig. 12(c) shows the achieved critical node predictions. We find that still the representative nodes influence their own line to which they belong, with the difference that the representative nodes for the purple lines also influence the orange line, one of the representative nodes for the brown line (top, corresponding to Floridsdorf station) also influences the orange line, whereas the other brown representative one (left, corresponding to Siebenhirten station) influences the green line. Both representative nodes fro the purple line influence the orange one, and one of the representative ones of the orange line influences the red line. The vast majority of nodes are critical for the orange line and green lines. Very few of them influence the red line. This test obtained a bigger accuracy reaching the 72.23%

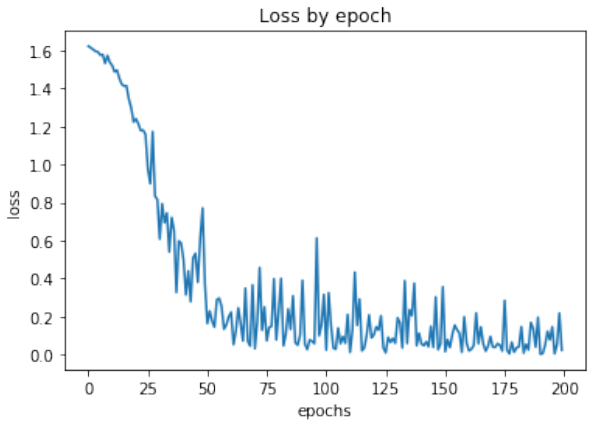


(a) Loss by epoch

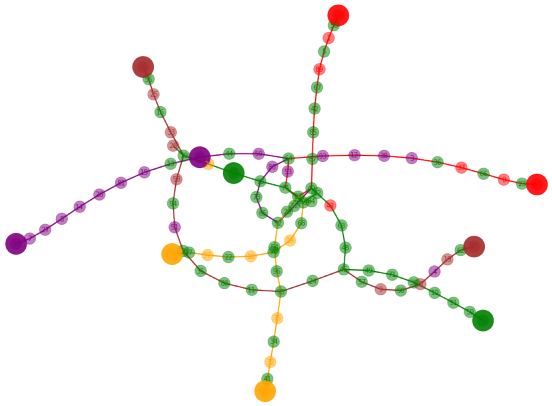


(b) Basic 2-layer GCN Predictions

Fig. 10: First approach: (a) Evolution of the loss function along the 100 training epochs (b) Predictions of the basic 2-layer GCN of the critical nodes for each line.



(a) Loss by epoch



(b) Sophisticated GCN Predictions

Fig. 11: Second approach: (a) Graph of the loss by 200 epochs (b) Critical node predictions from the enhanced GCN

It is interesting to visualize the predictions with the t-Distributed Stochastic Neighbor Embedding for two components, since a big percentage of the nodes are either classified as orange or green, and this way we can try to verify if there actually exists two subgroups. As shown in Fig. 12(c), we can see that two subgroups are formed.

8 Discussion of the broad paradigm of this work

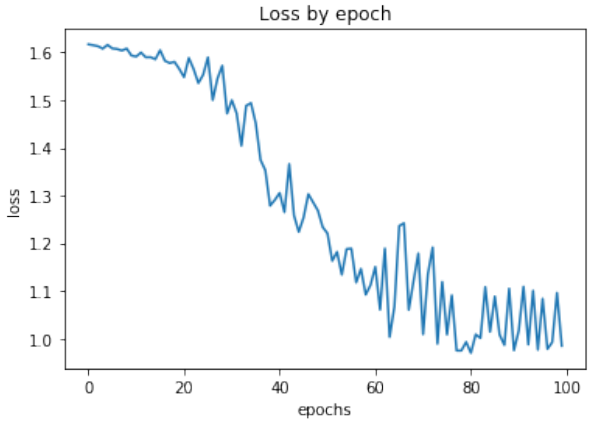
8.1 Air Traffic Management

The Single European Sky Air Traffic management (ATM) Research (SESAR) project is the technological pillar of the European Commission's Single European Sky Initiative to modernize ATM. They describe the process of establishing SESAR and the main parts of the project: the research and development (RD) part, which is led by the SESAR Joint Undertaking; the deployment part, which is managed by the SESAR Deployment Manager; and the European ATM Master Plan, which collects and lays out both the RD and deployment needs. The latest European ATM Master Plan was adopted just prior to the current pandemic. The huge loss in air traffic due to the pandemic, and the speed of the recovery of the aviation industry will require re-prioritization, but the main elements that have been established—particularly those in support of the environment—remain valid.

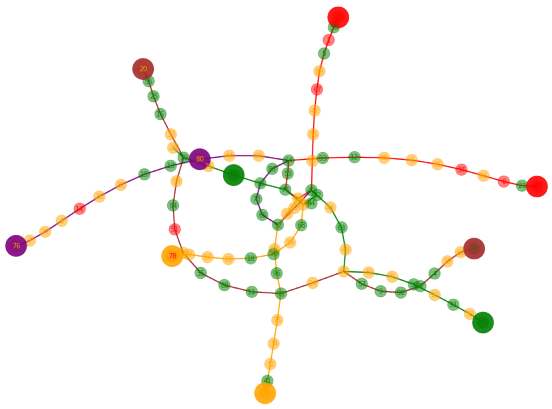
8.2 UAV Environment interaction

Small unmanned aerial vehicles have gained significant interest in the last decade. More specifically these vehicles have the capacity to impact package delivery logistics in a disruptive way. The research problems and state-of-the-art solutions that facilitates package delivery [28], are multiple and in many fields. Different aerial manipulators and grippers are listed along with control techniques to address stability issues. Landing on a platform is next discussed which encompasses static and dynamic platforms. Landing on a dynamic platform presents further challenges. This includes the delayed control responses and poor precision of the relative motion between the platform and aerial vehicle. Subsequently, risks such as weather conditions, state estimation and collision avoidance to ensure safe transit is considered. Finally, delivery UAV routing is investigated which categorises the topic into two areas: drone operations and drone-truck collaborative operations. Additionally, it is compared the solutions against design, environmental and legal constraints.

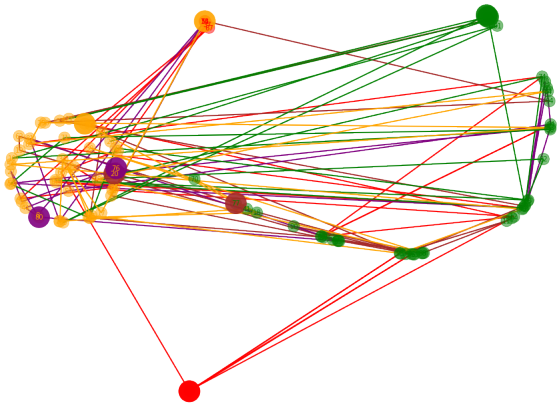
1. **Aerial Manipulation:** Different manipulator designs have been identified which includes: rigid linked, cable, continuum, foldable, hydraulic, and ejection; providing contrasting advantages and disadvantages. One major issue with aerial manipulation relates to the coupling effect experienced when operating a package. Centralized and decentralized methods deal with the disturbances, caused when operating the manipulator, differently. Specifically



(a) Loss by epoch



(b) Advanced GCN Predictions



(c) tSNE of the predictions

Fig. 12: Third approach: (a) Plot of the loss function by 100 epochs (b) Critical node predictions of the advanced GCN (c) t-distributed stochastic neighbor embedding (t-SNE) for the predictions of the third approach

for cable based controllers, cable tension must be accounted for to generate smooth or aggressive trajectories. Additionally, calculating the tension within the cable is still an open research question due to the difficulty in modelling the kinematics of the cable attached to the UAV.

2. **Aerial Grasping:** Additionally, different end-effector designs have been identified, which includes: ingressive, magnetic, tendon, and vacuum based. Center of mass misalignment can occur due to errors when gripping the payload. Stability can be compromised if not this is not incorporated into the control system through an adaptive mechanism. Furthermore, the centre of mass won't always alight with the centre of geometry. The contents of the package are not known a priori, which can lead to non-uniformity. Damage to the payload can be prevented through haptic feedback and gripper compliance using soft-tissue based materials. Dynamic grasping can also be utilized to improve the efficiency of the logistic operation by grasping the object mid-flight.
3. **Autonomous Landing:** Before a package can be delivered, the target zone must first be located. More recently, this has been done using computer vision and machine learning techniques. However, alternative techniques exist, such as GPS based. Nevertheless, GPS based methods are not suitable for dynamic platforms due to the inaccurate measurements. Landing on dynamic platforms is advantageous for truck and UAV collaborative operations. Detecting the relative motion between the two vehicles is crucial. The delayed control response and poor precision of the relative motion makes this problem extremely difficult. Researchers are either able to use pure computer vision techniques or build a dynamic model of the ground vehicle to calculate the relative motion. Finally, researchers have investigated reducing the impact force caused when landing through passive and active landing gears.
4. **Safe Transit:** Environmental conditions have an impact on component health, battery life, and flight performance. Software can aid with the analysis of weather data and forecasting for safe flight. When landing, the UAV is exploring an unknown and potential GPS-denied environment. This can lead to difficulties observing features for localization and obstacle detection. Cooperative vehicles, on the other hand, can utilize radar to broadcast positional information to surrounding vehicles using automatic dependent surveillance broadcast systems. In spite of this, broadcasting systems have security flaws such as denial of service attacks and spoofing. Unmanned traffic management systems provide a solution to these attacks while also being scalable for autonomous flight, which is crucial for autonomous package delivery.
5. **Routing:** Routing problems can be classified into either drone or drone-truck combined operations. Furthermore, depending on the location of the depot, drone-truck combined operations can be further classified into flying sidekick or parallel operations. Flying sidekick based operations result in the truck and UAV working in tandem. Whereas, no cooperative behaviour occurs in parallel operations. Researchers found the latter to be more optimal when the target location was within the UAV's flight range of the depot. Dif-

ferent constraints can be incorporated into the optimization problem which includes design, operational and legal constraints

6. **Design, Environmental, and Legal Constraints:** Aviation regulatory bodies establish national and international standards and regulations to ensure safe and efficient use of the airspace. These heavily impact the logistic operations of unmanned package aerial vehicles. Regulations on autonomous flight are heavily limited, requiring special licences to fly. However, regulations on piloted flight are currently more flexible. These regulations consist of operational constraints which are influenced by: privacy, security, safety, and environmental impacts. Operational constraints consist of vertical height limitations and critical separation distances from secure locations such as airports and military bases. Furthermore, certain technical capabilities may be required, such as the ADS-B out transmitters. Privacy issues arise when the delivery UAV needs to land on private property. Developers need to adhere to a privacy-by-design philosophy to ensure protection of personal data. Classification algorithms can capture personal belongings and identifiable human features, which can be intrusive. Furthermore, lost packages can lead to personal information and, in the worst case scenario, identity theft. Addressing privacy issues, along with provable reliability of the system, will help to increase public trust of delivery UAVs. Equivalent level of safety of manned aerial vehicles is required for the safe integration of autonomous delivery UAVs, which is currently an open research area. Hazard assessment, identifying the major risks of delivery UAVs, would also provide benefits to insurance companies. These hazards would include take-off and landing in unknown environments and aerial manipulation of the package. To provide a complete insurance policy, insurance companies need to understand all the parameters that influence these risks. This includes: legality, operational use, training and experience with a list of human factors, system reliability, and system or package value. Finally, environmental factors consist of noise pollution, aesthetic impact and effects on wildlife. Researchers have shown noise from UAVs is considerably more irritating than road traffic.
7. **Future Trends:** Organizations attempting to break into the autonomous aerial package delivery space will struggle to 'lift-off' due to restrictive legislative constraints and design problems. Cable based manipulators show promise due to the ability to deliver a package without landing and lowering the payload to the target area. This payload delivery mechanism has been therefore being used by Google Wing. This overcomes the privacy issue of landing within the curtilage of private property. Furthermore, noise generated from the UAV is minimized as the separation distance is large. Alternatively, soft-based tendon grippers also serve as landing gears which reduces the force impact and can also utilize perching for UAV-truck collaborative operations. However, more research is required in reducing fatigue caused by consistent landing. Gripping based methods need further work to better generalize to all packages. Ingressive and magnetic grippers require custom designed packages. Furthermore, vacuum based grippers are too susceptible to the environment due to pressure leakage. However, Soft-tendon

based grippers provide package compliance which shows promise. On the other hand, stability and control of gripping while ensuring strong and reliable grip on the package is a challenge. Finally, the use of unmanned traffic management systems will be utilized to cooperative identify other unmanned aerial vehicles, while equivalent sense and avoid capabilities will loosen regulation and enable safe integration into the airspace.

8.3 Last Mile Problem

Rapid technological developments in autonomous unmanned aerial vehicles (UAV or drones) and an evolving legislation may soon open the way for their large-scale implementation in the last mile delivery of products. The use of drones could drastically decrease labour costs and has been hyped as a potential disrupt to the parcel delivery industry. Online retailers and delivery companies such as Amazon [32] [2], are already filing up patents for the development of multi-level fulfilment centres for unmanned aerial vehicles or "drone-beehives" that would allow the deployment of this technology within built environment. A substantial amount of research has been carried out in the last years on the potential use of drones for parcel delivery, principally in the area of logistic optimisation. However, little is known about the potential market and economic viability of such services in Europe. Modelling framework using EU-wide high-resolution population and land-use data to estimate the potential optimal location of drone-beehives based on economic viability criterion, estimates the potential number of EU28 citizens that could potentially benefit from last mile-drone delivery services under four scenarios.

8.4 Algorithms for obstacle avoidance and drone flock interaction

In drone swarm control it is important the flock interaction and communication for a successful behaviour in obstacle avoidance. The success of swarm behaviors often depends on the range at which robots can communicate and the speed at which they change their behavior. Challenges arise when the communication range is too small with respect to the dynamics of the robot, preventing interactions from lasting long enough to achieve coherent swarming. To alleviate this dependency, most swarm experiments done in laboratory environments rely on communication hardware that is relatively long range and wheeled robotic platforms that have omnidirectional motion. Instead, Reynolds flocking [13] focus on deploying a swarm of small fixed-wing flying robots. Such platforms have limited payload, resulting in the use of short-range communication hardware. Furthermore, they are required to maintain forward motion to avoid stalling and typically adopt low turn rates because of physical or energy constraints. The trade off between communication range and flight dynamics is exhaustively studied in simulation in the scope of Reynolds flocking and demonstrated with up to 10 robots in outdoor experiments.

There exists also a theoretical framework for design and analysis of distributed flocking algorithms [25]. Two cases of flocking in free-space and presence of multiple obstacles are considered using three flocking algorithms: two for free-flocking and one for constrained flocking. The first algorithm embodies all three rules of Reynolds. This is a formal approach to extraction of interaction rules that lead to the emergence of collective behavior. This algorithm generically leads to regular fragmentation, whereas the second and third algorithms both lead to flocking. A systematic method is provided for construction of cost functions (or collective potentials) for flocking. These collective potentials penalize deviation from a class of lattice-shape objects called /spl alpha/-lattices, and it is used a multi-species framework for construction of collective potentials that consist of flock-members, or /spl alpha/-agents, and virtual agents associated with /spl alpha/-agents called /spl beta/- and /spl gamma/-agents. We show that migration of flocks can be performed using a peer-to-peer network of agents, i.e., "flocks need no leaders." A "universal" definition of flocking for particle systems with similarities to Lyapunov stability is given. Several simulation results are provided that demonstrate performing 2-D and 3-D flocking, split/rejoin maneuver, and squeezing maneuver for hundreds of agents using the proposed algorithms.

Aerial restrictions are often a problem in order to analyze the correct behaviour of a single drone or swarm,in adition, a fundamental issue of collective motion of aerial robots presents itself: how to ensure that large flocks of autonomous drones seamlessly navigate in confined spaces [35]. The numerous existing flocking models are rarely tested on actual hardware because they typically neglect some crucial aspects of multi-robot systems. Constrained motion and communication capabilities, delays, perturbations, or the presence of barriers should be modeled and treated explicitly because they have large effects on collective behavior during the cooperation of real agents. Handling these issues properly results in additional model complexity and a natural increase in the number of tunable parameters, which calls for appropriate optimization methods to be coupled tightly to model development. The proposal of such a flocking model for real drones incorporating an evolutionary optimization framework with carefully chosen order parameters and fitness functions, is numerically demonstrated that the induced swarm behavior remained stable under realistic conditions for large flock sizes and notably for large velocities. Coherent and realistic collective motion patterns persisted even around perturbing obstacles. Furthermore, the model is validated on real hardware, carrying out field experiments with a self-organized swarm of 30 drones. This is the largest of such aerial outdoor systems without central control reported to date exhibiting flocking with collective collision and object avoidance. The results confirmed the adequacy of our approach. Successfully controlling dozens of quadcopters will enable substantially more efficient task management in various contexts involving drones.

Often, this algorithms are tested using convex obstacles. It is important to investigate a particular behavior of autonomous mobile robots for obstacle avoidance, where the geometry of obstacle is non convex in nature [19]. The detection

and avoidance of the obstacle is done by means of robot exploration in the given environment. This behavior is modeled using Hybrid Cellular Automata controllers.

There exist the need to also take into account dynamic environments, where objects may have a movement and therefore can interfere with the previously set path. In the problem of decentralized multi-robot target tracking and obstacle avoidance in dynamic environments [33], each robot executes a local motion planning algorithm which is based on model predictive control (MPC). The planner is designed as a quadratic program, subject to constraints on robot dynamics and obstacle avoidance. Repulsive potential field functions are employed to avoid obstacles. The novelty of this approach lies in embedding these non-linear potential field functions as constraints within a convex optimization framework. This method convexifies non-convex constraints and dependencies, by replacing them as pre-computed external input forces in robot dynamics.

9 An approach to carry out systematic simulation experiments

We used the existing obstacle avoidance simulator SwarmLab [31], adapting it to use different kinds of obstacles, for a first simulation of the flight movement, making a comparative of the decentralized algorithms from the state of the art for the navigation of aerial swarms in cluttered environments, Olfati-Saber's [25] and Vasarhelyi's [35] algorithms.

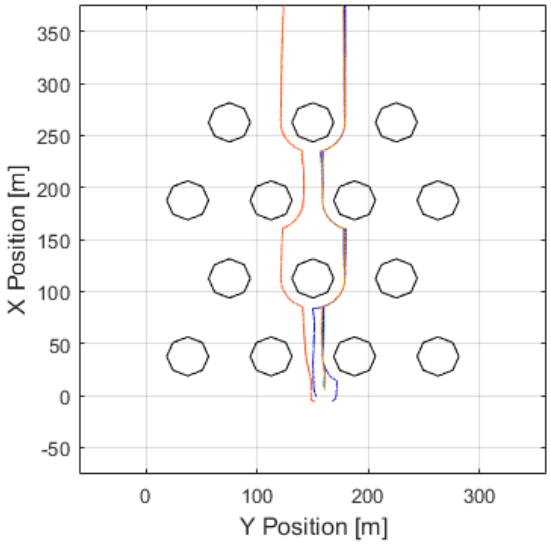
We apply this algorithms on a 3D occupancy grid mapping approach, Fig. 17, that provides data structures and mapping algorithms in C++ particularly suited for robotics. The map implementation is based on an Octree and is designed to meet several requirements [14] fulfilled by the OctoMap library.

Once we have analysed the obstacle avoidance problem, we need to analyze the movement in predefined path itself in a simulated environment [12] to monitorize the flow of the drones, using several techniques of reinforcement learning for the UAVs to follow a proper and safe route [27]. This raises a new topic of Traffic Forecasting [18] [3] [8] using Graph Neural Networks [6] [37] - [20] as a guide.

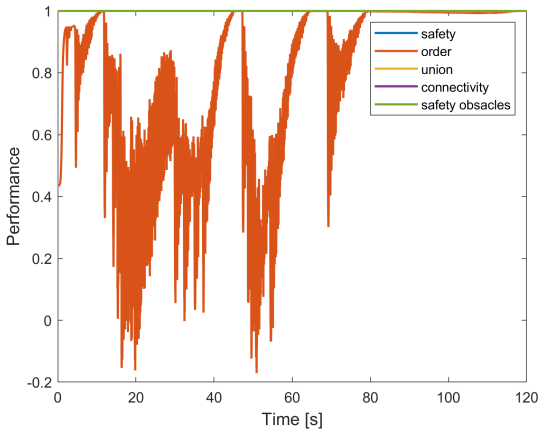
10 Conclusions and Future Work

This work started trying to reproduce the results of a previous published analysis over a metro network whose graph representation is relatively small and manageable. We have extended this analysis proposing two additional GNN architectures. The assumption is that the aerial paths may be shaped in a similar way as the subway network paths following approximately the layout of streets in the city: In fact the airspace allows more freedom to maneuver, taking into account the limitations to the legislation.

The analysis has been made with a relatively manageable network where the results of the analysis are the expected and promising, but it still remains to be

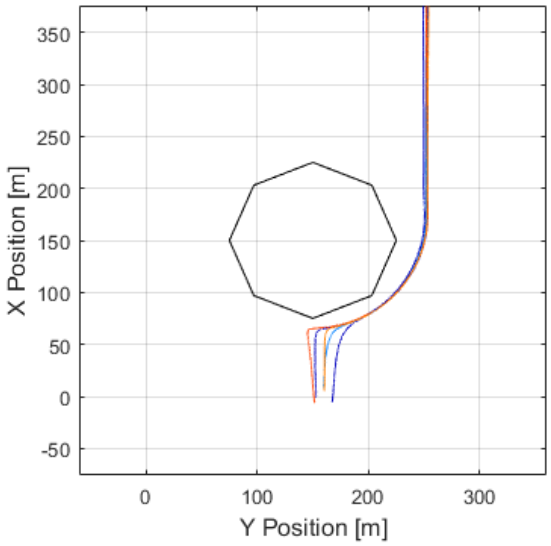


(a) Trajectories generated by Olfati-Saber's algorithm

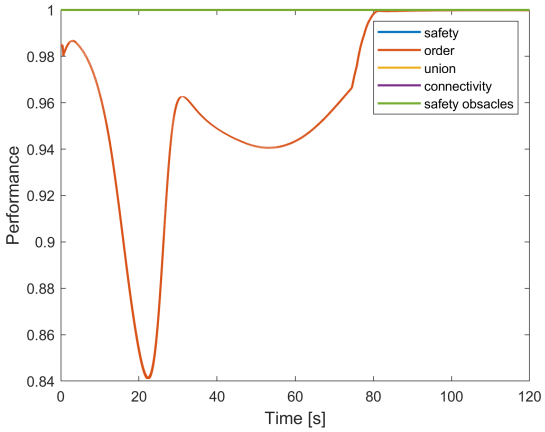


(b) Performance of the algorithm

Fig. 13: Olfati-Saber's algorithm for multiple obstacle avoidance: (a) Trajectories generated for a swarm of 5 drones (b) Overall performance

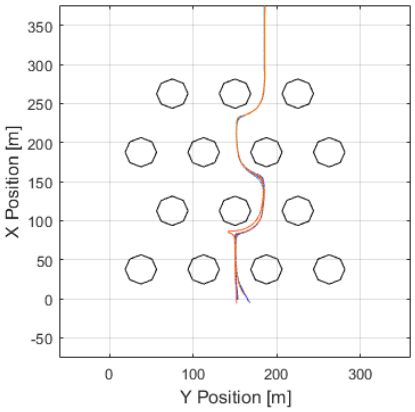


(a) Trajectories generated by Vasarhelyi's algorithm

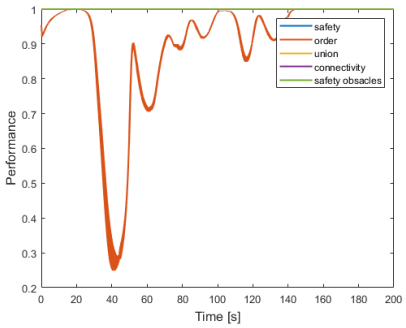


(b) Performance of the swarm group

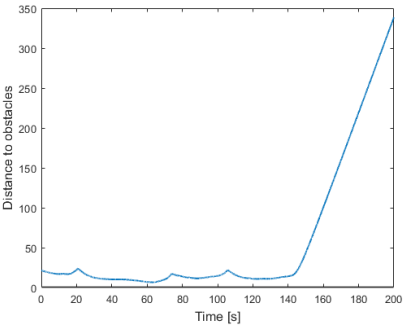
Fig. 14: Vasarhelyi's algorithm for a single obstacle avoidance: (a) Trajectories generated by Vasarhelyi's algorithm (b) Performance of the swarm group according to the algorithm



(a) Trajectories generated by Vasarhelyi's algorithm

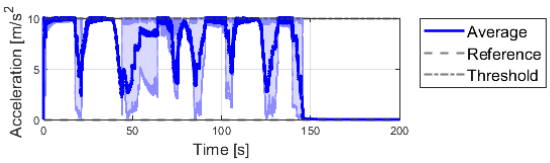


(b) Performance of the algorithm

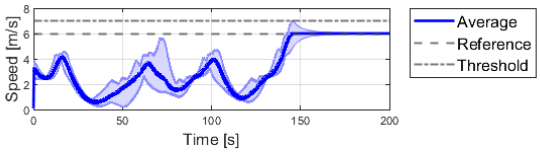


(c) Distances to the obstacles

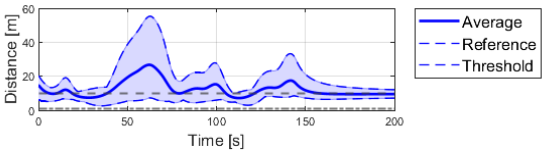
Fig. 15: Vasarhelyi's algorithm for multiple obstacles avoidance: (a) Trajectories generated by Vasarhelyi's algorithm (b) Performance of the swarm group according to the algorithm



(a) Global Aceleration

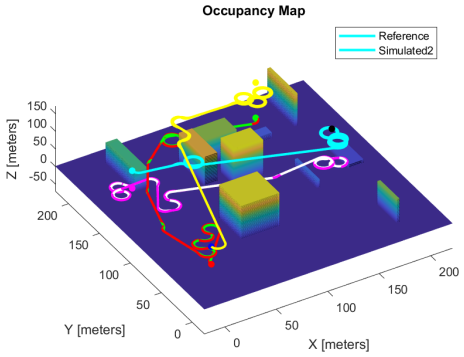


(b) Global Speed

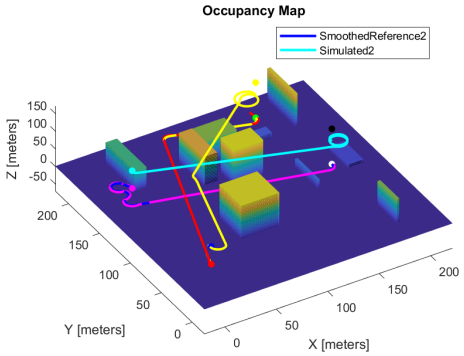


(c) Global distance

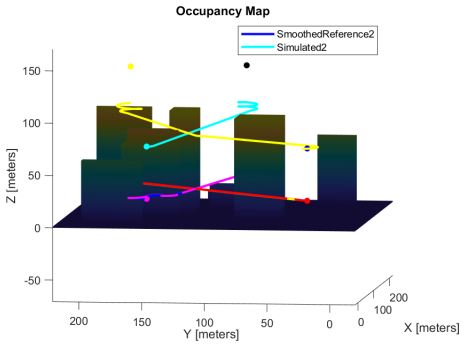
Fig. 16: Global values of the swarm: (a) Global acceleration of the swarm during time (b) Global speed of the swarm during time (c) Global distance of the swarm during time



(a) Trajectories generated by Olafti-Saber's algorithm without smoothing



(b) Trajectories generated by Vasarhelyi's algorithm with smoothing



(c) Side view of the trajectories generated by Vasarhelyi's algorithm with smoothing

Fig. 17: Trajectories generated on an occupancy map: (a) Trajectories generated by Olafti-Saber's algorithm without smoothing (b) Trajectories generated by Vasarhelyi's algorithm with smoothing (c) Side view of the trajectories generated by Vasarhelyi's algorithm with smoothing

analyzed if the result is just as promising in a more complete and extensive graph, and in the same way, in a much less extensive graph. Since we are analyzing the importance of the most influential nodes in the paths, the Graph Neural Networks allow us to incorporate more information in each node and therefore more restrictions on the state of each line can be added, and it would allow us to merge the analysis of the network with state analysis.

Future works will address the extraction of airways graphs from simulation and from real data, in order to extend our analysis to more complex graphs and to test the identification of critical nodes against the actual congestion in the simulated or real network when such nodes fail or are compromised. This lead to the notion of traffic security that may be further examined.

References

1. Atwood, J., Towsley, D.: Diffusion-convolutional neural networks (2015). https://doi.org/10.48550/ARXIV.1511.02136, bluehttps://arxiv.org/abs/1511.02136
2. Aurambout, J.P., Gkoumas, K., Ciuffo, B.: Last mile delivery by drones: an estimation of viable market potential and access to citizens across european cities. European Transport Research Review **11** (12 2019). https://doi.org/10.1186/s12544-019-0368-2
3. Bai, L., Yao, L., Li, C., Wang, X., Wang, C.: Adaptive graph convolutional recurrent network for traffic forecasting (2020)
4. Bernard, M., Kondak, K., Maza, I., Ollero, A.: Autonomous transportation and deployment with aerial robots for search and rescue missions. Journal of Field Robotics **28**, 914–931 (11 2011). https://doi.org/10.1002/rob.20401
5. Bolić, T., Ravenhill, P.: Sesar: The past, present, and future of european air traffic management research. Engineering **7**(4), 448–451 (2021). https://doi.org/https://doi.org/10.1016/j.eng.2020.08.023, bluehttps://www.sciencedirect.com/science/article/pii/S2095809921000503
6. Bronstein, M.M., Bruna, J., Cohen, T., Veličković, P.: Geometric deep learning: Grids, groups, graphs, geodesics, and gauges (2021)
7. Bruna, J., Zaremba, W., Szlam, A., LeCun, Y.: Spectral networks and locally connected networks on graphs (2013). https://doi.org/10.48550/ARXIV.1312.6203, bluehttps://arxiv.org/abs/1312.6203
8. Chen, C., Li, K., Teo, S.G., Chen, G., Zou, X., Yang, X., Vijay, R.C., Feng, J., Zeng, Z.: Exploiting spatio-temporal correlations with multiple 3d convolutional neural networks for citywide vehicle flow prediction. In: 2018 IEEE International Conference on Data Mining (ICDM). pp. 893–898 (2018). https://doi.org/10.1109/ICDM.2018.00107
9. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering (2016). https://doi.org/10.48550/ARXIV.1606.09375, bluehttps://arxiv.org/abs/1606.09375
10. Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P.: Convolutional networks on graphs for learning molecular fingerprints (2015). https://doi.org/10.48550/ARXIV.1509.09292, bluehttps://arxiv.org/abs/1509.09292

11. Floreano, D., Wood, R.J.: Science, technology and the future of small autonomous drones. Nature **521**, 460–466 (2015)
12. Glossner, J., Murphy, S., Iancu, D.: An overview of the drone open-source ecosystem (2021)
13. Hauert, S., Leven, S., Varga, M., Ruini, F., Cangelosi, A., Zufferey, J.C., Floreano, D.: Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate. In: 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5015–5020 (2011). https://doi.org/10.1109/IROS.2011.6095129
14. Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C., Burgard, W.: OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots (2013). https://doi.org/10.1007/s10514-012-9321-0, bluehttps://octomap.github.io, software available at bluehttps://octomap.github.io
15. Jiang, B., Zhang, Z., Lin, D., Tang, J., Luo, B.: Semi-supervised learning with graph learning-convolutional networks. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11305–11312 (2019). https://doi.org/10.1109/CVPR.2019.01157
16. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). https://doi.org/10.48550/ARXIV.1609.02907, bluehttps://arxiv.org/abs/1609.02907
17. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks (2016). https://doi.org/10.48550/ARXIV.1609.02907, bluehttps://arxiv.org/abs/1609.02907
18. Kumar, D.: Video based traffic forecasting using convolution neural network model and transfer learning techniques. Journal of Innovative Image Processing **2**, 128–134 (06 2020). https://doi.org/10.36548/jiip.2020.3.002
19. Kumar, K.K., Vaidyan, M.V.: Modeling of non-convex obstacle detection and avoidance mobile robot by hybrid cellular automata. In: 2013 Third International Conference on Advances in Computing and Communications. pp. 9–12 (2013). https://doi.org/10.1109/ICACC.2013.9
20. Li, Y., Ma, L., Zhong, Z., Liu, F., Cao, D., Li, J., Chapman, M.A.: Deep learning for lidar point clouds in autonomous driving: A review (2020)
21. Mcguire, K., De Wagter, C., Tuyls, K., Kappen, H., Croon, G.: Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. Science Robotics **4**, eaaw9710 (10 2019). https://doi.org/10.1126/scirobotics.aaw9710
22. Mellinger, D., Shomin, M., Michael, N., Kumar, V.R.: Cooperative grasping and transport using multiple quadrotors. In: DARS (2010)
23. Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., Bronstein, M.M.: Geometric deep learning on graphs and manifolds using mixture model cnns (2016). https://doi.org/10.48550/ARXIV.1611.08402, bluehttps://arxiv.org/abs/1611.08402
24. Nogami, J., Nakasuka, S., Tanabe, T.: Real-time decision support for air traffic management, utilizing machine learning. Control Engineering Practice **4**(8), 1129–1141 (1996). https://doi.org/https://doi.org/10.1016/0967-0661(96)00113-X, bluehttps://www.sciencedirect.com/science/article/pii/096706619600113X
25. Olfati-Saber, R.: Flocking for multi-agent dynamic systems: algorithms and theory. IEEE Transactions on Automatic Control **51**(3), 401–420 (2006). https://doi.org/10.1109/TAC.2005.864190
26. Pandhre, S., Mittal, H., Gupta, M., Balasubramanian, V.N.: Stwalk. Proceedings of the ACM India Joint International Conference on Data Science and Management

of Data (Jan 2018). https://doi.org/10.1145/3152494.3152512, bluehttp://dx.doi.org/10.1145/3152494.3152512
27. Placed, J.A., Castellanos, J.A.: A deep reinforcement learning approach for active slam. Applied Sciences **10**(23) (2020). https://doi.org/10.3390/app10238386, bluehttps://www.mdpi.com/2076-3417/10/23/8386
28. Saunders, J., Saeedi, S., Li, W.: Autonomous aerial delivery vehicles, a survey of techniques on how aerial package delivery is achieved (2021)
29. Schilling, F., Lecoeur, J., Schiano, F., Floreano, D.: Learning vision-based flight in drone swarms by imitation. IEEE Robotics and Automation Letters **4**, 4523–4530 (2019)
30. Shen, Y., Chen, J., Huang, P.S., Guo, Y., Gao, J.: M-walk: Learning to walk over graphs using monte carlo tree search (2018)
31. Soria, E., Schiano, F., Floreano, D.: Swarmlab: a matlab drone swarm simulator (2020)
32. Sunghun, J., Hyunsu, K.: Analysis of amazon prime air uav delivery service. Journal of Knowledge Information Technology and Systems **12**, 253–266 (2017)
33. Tallamraju, R., Rajappa, S., Black, M.J., Karlapalem, K., Ahmad, A.: Decentralized mpc based obstacle avoidance for multi-robot target tracking scenarios. 2018 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR) (Aug 2018). https://doi.org/10.1109/ssrr.2018.8468655, bluehttp://dx.doi.org/10.1109/SSRR.2018.8468655
34. Tuysuz Erman, A., Hoesel, L., Havinga, P., Wu, J.: Enabling mobility in heterogeneous wireless sensor networks cooperating with uavs for mission-critical management. Wireless Communications, IEEE **15**, 38 – 46 (01 2009). https://doi.org/10.1109/MWC.2008.4749746
35. Vásárhelyi, G., Virágh, C., Somorjai, G., Nepusz, T., Eiben, A.E., Vicsek, T.: Optimized flocking of autonomous drones in confined environments. Science Robotics **3**(20), eaat3536 (2018). https://doi.org/10.1126/scirobotics.aat3536, bluehttps://www.science.org/doi/abs/10.1126/scirobotics.aat3536
36. Wang, H., Leskovec, J.: Unifying graph convolutional neural networks and label propagation (2020). https://doi.org/10.48550/ARXIV.2002.06755, bluehttps://arxiv.org/abs/2002.06755
37. Wang, T., Liao, R., Ba, J., Fidler, S.: Nervenet: Learning structured policy with graph neural networks. In: ICLR (2018)
38. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems **32**(1), 4–24 (jan 2021). https://doi.org/10.1109/tnnls.2020.2978386
39. Yu, E.Y., Wang, Y.P., Fu, Y., Chen, D.B., Xie, M.: Identifying critical nodes in complex networks via graph convolutional networks. Knowledge-Based Systems **198**, 105893 (2020). https://doi.org/https://doi.org/10.1016/j.knosys.2020.105893, bluehttps://www.sciencedirect.com/science/article/pii/S0950705120302409
40. Zhang, J., Hu, J., Lian, J., Fan, Z., Ouyang, X., Ye, W.: Seeing the forest from drones: Testing the potential of lightweight drones as a tool for long-term forest monitoring. Biological Conservation **198**, 60–69 (Jun 2016). https://doi.org/10.1016/j.biocon.2016.03.027