Master Degree in ICT for Internet and Multimedia

Routing protocols in dynamic networks

Author

Mattia Perini

Supervisor

Prof. Leonardo Badia

Co-supervisor

Prof. Andrea M. Tonello

Academic Year 2018/2019

Ai miei genitori, Sandro ed Emanuela

"*Non fatevi scoraggiare da chi vi dice che la vita non è inseguire un sogno, da chi vi vuole sereni nella ristrettezza di un destino piccolo e di un futuro convenzionale. Sudate e lavorate duramente perché il vostro sogno si avveri o perché almeno possiate dire di averlo visto da vicino. La ricchezza piú grande di ognuno di noi è il futuro che sogna*".

Contents

Abstract

In recent times drones have found application in many scenarios in the field of wireless networks.
Drones can act as flying base station to enhance the coverage and rate performance in different scenarios such as temporary hotspot or emergency situations.
The main advantage compared to the terrestrial base stations is that they have a higher chance of *Line-of-Sight (LoS)* links to ground users because they work at higher altitude and so reflection and shadowing are diminished.
In this work we analysed two problems: the characteristics of a routing protocol and the optimal drones' deployment to cover the maximum number of users.
We implemented two protocols that are optimal with respect to the metrics that they optimize: one protocol minimizes the number of hops to reach the destination while the other one maximizes the transmission rate.
Then, we compared them through end-to-end delay.
The results show that the protocol that works better is the one that minimizes the number of hops.
Then in the second part we implemented two algorithms to maximize the number of covered users.
One solves an optimization problem while the other one computes the density map using convolution.
At the end we analysed the pros and cons of each algorithm.

Chapter 1

Introduction

Since drones or *UAVs (Unmanned Aerial Vehicles)* have become accessible to the public, industries started to explore new fields in which they can be useful.
Usually UAVs have been used in many surveillance and reconnaissance missions consisting in monitoring some areas and relaying information such as images, videos and sensor data to a distant control station.
But, in recent times, they have found application also in wireless networks. In fact, UAVs can act as flying base station to enhance the coverage and rate performance in different scenarios such as temporary hotspot or emergency situations.
They have several advantages compared to the terrestrial base stations: they have a higher chance of *Line-of-Sight (LoS)* links to ground users because they work at higher altitude and so reflection and shadowing are diminished, they can easily move and so they can be deployed in order to guarantee rapid communication.
In [1] are described some applications that include the use of UAVs in wireless networks.
For example, in case of disaster, UAVs can be used to maintain global network connectivity if a base station's failure occurs otherwise they can be used to improve network capacity and coverage in crowded areas (such as stadiums) by offloading the traffic from the cellular infrastructure.
Similarly, the networks can be densified by using UAV cells with the base stations together in order to enhance the achievable data rate and coverage for future generation systems such as 5G. Traffic offloading through network densification and the use of *millimeter-wave (mmWave)* technology are key enablers for 5G networks.
When we deal with a wireless network with UAVs, we have to face several challenges. [2] One of the main interesting challenge is trajectory planning,

due to constraints such as battery limitation and collision. For example, suppose we have to inspect a zone, the goal is to maximize the observed area consuming as less energy as possible.
Another interesting challenge is the optimal UAVs deployment. In this case we have to take into account more factors.
Depending by the application drones have to be placed in the best possible configuration but if we consider an application where an UAV needs to communicate with a ground user we have to optimize also the altitude of a drone and, in case of more than one drone, the intensity because they affect the Air-to-Ground channel.
For example, users in high-rise urban scenarios may require higher LoS connectivity, whereas users in suburban scenarios may need higher degree of path loss reduction. Remember that the higher altitude of drones promotes higher LoS connectivity since reflection and shadowing are diminished, whereas lower altitude ensures reduction in path loss.
So, after having analysed all the applications and the mains problem that we should have faced we decided to study the routing protocols in these dynamic networks.

In this thesis we implemented two routing protocols that are optimal with respect to a specific metric: one protocol minimizes the number of hops to reach the destination while the other one maximizes the transmission rate. Then the performance of these two protocols have been compared through the end-to-end delay defined as the time since a node checks the routing table to see if there is a valid route for the destination to the instant at which the packed is received.
The results show that the protocols that minimize the number of hops is better than the other one and not only from delay point of view but also in general.
The main problem that we faced is the mobility of drones because they can move fast and so a link can suddenly appear or disappear, or the characteristics of the channel can change. For this reason, we have considered georouting.
Georouting is a routing principle based on geographic position information. The basic idea is that the source sends a message to the physical location of the destination instead of the network address.
So, at each instant, every node needs to know the physical position of all others node in the network.
In the second part of the thesis, suppose we have a defined number of users randomly distributed in a certain area, we implemented two different algorithms for the optimal UAVs deployment whose goal is to maximize the

number of covered users.
The first algorithm try to solve an optimization problem with some constraints using *Karush-Kuhn-Tucker (KKT)*. So, at each instant, this algorithm provides the best drones' configuration.
The second algorithm, instead, try to solve this problem through a convolution between a matrix and a Gaussian filter in order to find the densest zones.
Once we found the densest zone we check if it is possible to place a drone in that position verifying if there is overlapping.
At the end we compared the two algorithm and analysed the pros and cons of each other.
This work is an implementation of what I have seen during my academic experience to see how a routing protocol works and the main problems that we have to face.
One of the biggest limitations of this work is that the routing protocols do not take into account the consumed energy even if it is one of the most important metrics.
In this way we could have chosen the best protocol also from this point of view since, as we read in many papers, one of the main goals is to increase the lifetime of the network.
Another interesting aspect would be to analyse the link, in particular for how much time there is a stable connection.

Chapter 2

Routing protocols

Routing protocols can be divided into four categories: proactive, reactive, hybrid and geographic protocols. [3]

2.1 Proactive routing protocols

Proactive Routing Protocols (PRP) use tables in their nodes to store all the routing information about other nodes in the network. The main advantage is that each node contains the latest information of the routes.
The problem is that to keep the table up-to-date we need to exchange a lot of messages and their reaction to topology changes is slow. This is unsuitable for UAV networks.
One of the main PRP is *Optimized Link State Routing (OLSR).*
In this protocol routes to all destinations are determined at the beginning and then maintained by an update process. Each node broadcasts the link-state costs of its neighbouring nodes to other nodes using a flooding strategy. When a source node S needs to send data to a destination node D, the next hop is chosen using a shortest path algorithm.
In UAVs network the node locations and links change rapidly and, as consequence, a high number of control messages (so packets) needs to be exchanged.
One way to optimize the efficiency of this protocol is to select some nodes as *Multipoint Relays (MPR)* that only forward the control traffic packet reducing the number of transmission required.
Another important PRP is *Destination Sequenced Distance Vector (DSDV)* that is a table driven protocol based on Bellman-Ford algorithm with some adjustments for ad hoc networks. It uses two types of update packets, 'full-

dump' and 'incremental'.
When the topology of the network changes an incremental packet is sent; this reduce the overhead but it is still large.
In general, proactive protocols require a large bandwidth and this is a problem for aerial network.

2.2 Reactive routing protocols

Reactive Routing Protocols (RRP) are on-demand routing protocols. It means that a route from a source node to a destination node is established only when and if it needed. In this way the overhead problem is overcome but, on the other hand, the procedure of finding routes can take a long time, increasing latency.
RRPs can be divided into two categories: source routing and hop-by-hop routing.
In source routing a packet contains the entire path from a source node to a destination node so, intermediate nodes can forward packets based on this information. In this way the packet error rate and the overhead increase in a way proportional to the size of the network.
In hop-by-hop routing, each data packet contains only the address of the next hop. The advantage is that routes are adaptable to a dynamic topology but the disadvantage is that each node needs to maintain routing information for each active route and be aware of its neighbours.
Two commonly used RRPs are *Dynamic Source Routing (DSR)* [4] and *Ad hoc on Demand Distance Vector (AODV)* [5].

2.2.1 Dynamic Source Routing (DSR)

When a source node S needs to transmit data to a destination node D, it first looks in its route cache if there is a valid path (node sequence) for the destination. If there is a valid path it sends packets to the next hop otherwise it starts a new route discovery.
Route discovery is accomplished by broadcasting a *Route Request (RREQ)* packet that contains;

- A unique request id;
- Source and destination address;

- Record field listing the address of intermediate node.

When the destination node is reached it replies with a *Route Reply (RREP)* that includes a copy of the record list and the address of the source node. Source node will receive the RREP from destination node and use the piggybacked list therein to route a RREP back to D containing the record list from D to S. DSR assumes that links are unidirectional.

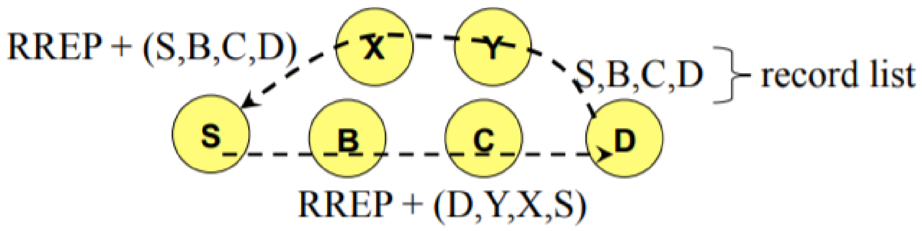


Figure 2.1: Example of RREP in DSR.

Alternatively, in case of bidirectional links, node D could just reverse the sequence of hops contained in the route record of the RREQ and use this as path for the route reply.
DSR uses also a mechanism of route maintainance in order to detect broken links.
If a node A sends a packet to node B it is the responsible for the packet delivery and it can request the sending of an *Acknowledgement (ACK)*. When the ACK request has been sent for a maximum number of times the link is marked as broken and a *Route Error (RERR)* message is returned to the source node.
On receiving the RERR, node S removes each path containing the broken link and checks if another path exists; if yes, the packet is routed using the alternative path otherwise a new route discovery starts.
In DSR, when a packet is sent, all the intermediate nodes cache the path to the destination and to each intermediate node. The reverse path is not cached because links are assumed unidirectional.

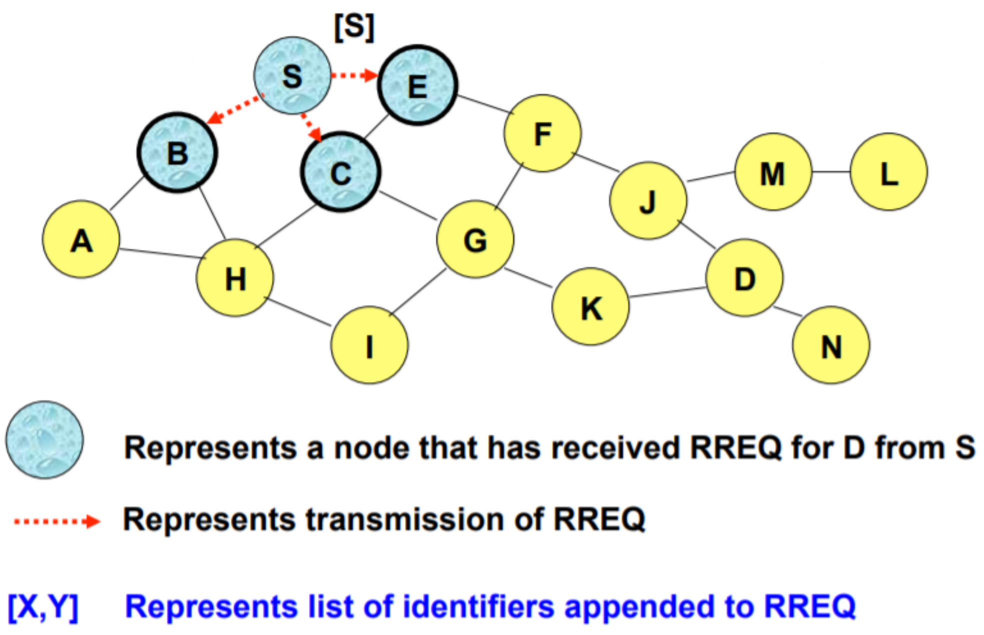

[S]
S
E
B
C
F
A
H
G
J
M
L
I
K
D
N
Represents a node that has received RREQ for D from S
Represents transmission of RREQ
[X,Y] Represents list of identifiers appended to RREQ

Figure 2.2: Example of route discovery in DSR.

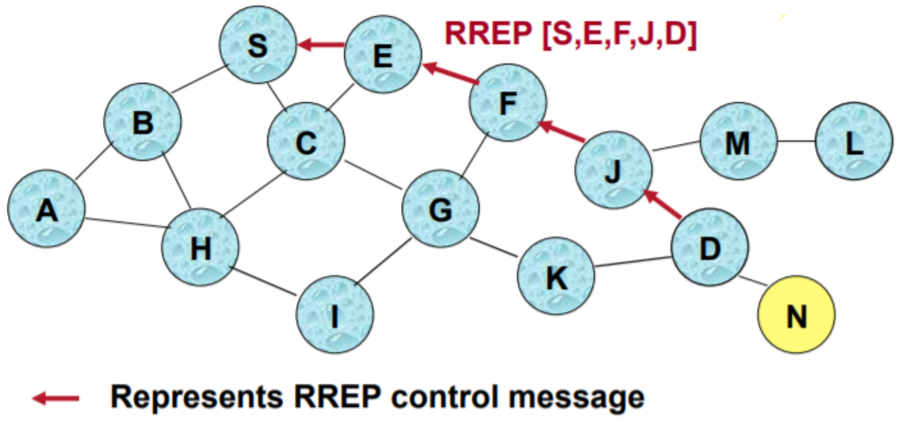


Figure 2.3: Example of RREP in DSR.

Advantages

- Routes maintained only between pairs of nodes that need to communicate. This reduces overhead of route maintenance;
- Route caching can further reduce route discovery overhead;
- A single route discovery may yield many routes to the destination because it replies to every RREQ;

Disadvantages

- Packet header size grows with route length due to source routing;
- Flood of route requests may potentially reach all nodes in the network;
- Potential collisions between route requests propagated by neighboring nodes that can be solved by inserting a random delays before forwarding RREQs;
- Route Reply Storm problem, increased contention if too many route replies come back due to nodes replying using their local cache;
- Stale caches will lead to increased overhead .

2.2.2 Ad hoc On-demand Distance Vector routing (AODV)

Ad hoc On-demand Distance Vector routing (AODV) retains the good feature of DSR that routes are maintained only between nodes which need to communicate and improve the performance of DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes.
The idea is the same of DSR but, in AODV, a route is determined either when the destination itself is reached or also when an intermediate node with a fresh-enough route to D is reached.
AODV selects the shortest delay path and not the shortest one in terms of number of hops because the destination node replies only to the first RREQ that it receives.
Contrarily to DSR, a routing table entry is removed if it is not used for an active route time out interval. Neighbouring nodes periodically exchange hello messages and when a link breaks all active neighbours are informed and a RERR message is propagated.

2.2.3 AODV vs DSR

- in DSR with a single RREQ a source node can learn the route to any intermediate node and each intermediate node can also learn routes to any intermediate node. Instead, in AODV each node learns only the next hop to reach the destination node;
- in DSR destination replies to all RREQs while, in AODV, it replies only to the first one;

- in DSR there is not a mechanism to refresh route, only when we are sending data and a link is not available any more a RERR message is sent. In AODV if a route is not used for a certain time the entry expires.

2.2.4 About mobility

With high mobility increases the probability of link failures so it is better to use AODV because it refreshes routing table more often since each node stores only one route for a given destination.
Using DSR if a route is not available any more it finds another route in its cache but it is highly probably that also that route is not valid.
Taking into account the MAC overhead, DSR is better because AODV starts route discovery more often. However, considering that in both AODV and DSR RREQ are broadcast, so RTS/CTS/data/ACK and MAC overhead aren't necessary, but RREP are sent using unicast transmission we see that overhead for DSR is higher than AODV. This is because DSR uses RREP more than AODV (a factor 4).

2.3 Hybrid Routing Protocols (HRP)

The third category are *Hybrid Routing Protocols (HRP)* that are a combination of reactive and proactive protocols.
They reduce the high latency due to route discovery in reactive protocols and the overhead problem of proactive protocols.
An example is *Zone Routing Protocol (ZRP)* [3].
In this protocol the network is divided in zones where intra-zone routing is performed with the proactive approach while inter-zone routing is done using the reactive approach. The parameter that most affect the efficiency of ZRP is the zone radius.
When two nodes belonging to two different zone need to communicate they send data to a subnet that is common to both nodes.
Another example is *Location Aided Routing (LAR)* but it will be explained in the next chapter because it is the protocol on which this work is based.
The last category are geographic protocols ([6] and [7]). In these protocols we assume that each node knows the geographic position of all others node and when a source node needs to send data to a destination node it sends the packet to the intermediate node closest to destination. This phase is called greedy forwarding. It can happen that the mechanism stops and the

recovering mechanism (called face routing) is applied to find another node where greedy forwarding can restart.

Chapter 3

Considered scenarios

3.1 Infrastructure

In an UAVs network links may be broken due to the drones' speed or because, suddenly, the channel changes. Depending by the mobility's level an UAV network can be infrastructure-based or ad hoc.
For example, in application that require UAV to act as flying base station to cover a certain area, we can use an infrastructure-based network where UAVs can communicate with each other and also with the control center.
Instead, in application where there is a high level of mobility is better to consider ad hoc structure.
So, when we had to choose the configuration of UAVs, we considered four options: star, multi-star, mesh and hierarchical mesh.
In a star topology all nodes are connected to one or more ground nodes and all communications among UAVs are routed through the ground nodes. This results in blockage of links, requirement of high bandwidth downlinks and higher latency. This is due to the downlink length because it is longer than the distance between UAVs since all communications must pass through a ground control center.
The multi-star topology is quite similar except the UAVs form multiple stars and one node from each group is connected to the ground station.
In case of mesh networks, the UAVs are interconnected and only a small number of them may be connected to the control center. So, for this reason, mesh networks are flexible, reliable and offer better performance.
So, in this way, if we have to send data from a source to a destination, the packet passes through intermediate nodes and find its way. Then, it is the routing protocol that ensure the delivery of the packets.

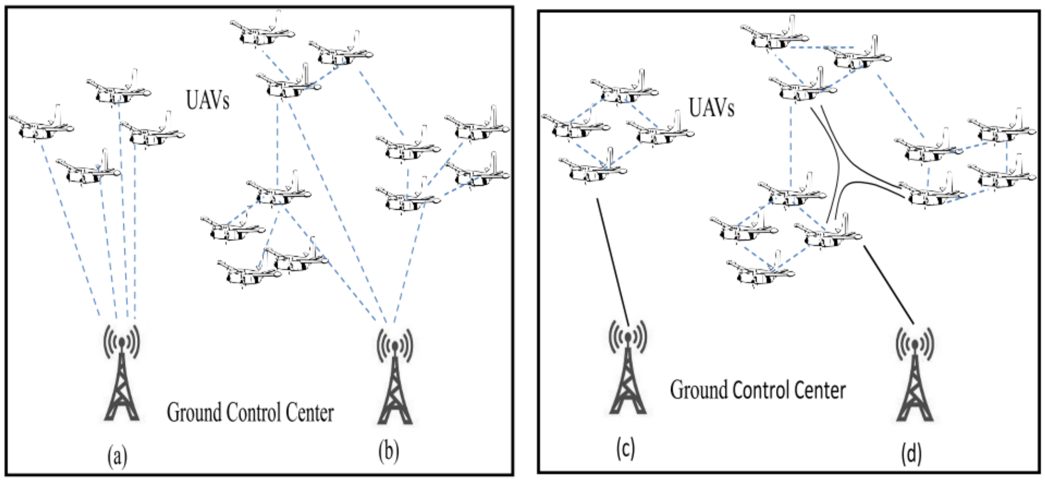


Figure 3.1: a
) Star Configuration b) Multi-star Configuration c) Flat Mesh Network d) Hierarchical Mesh Network

So, after examining all the cases I considered three scenarios:

- assisted cellular network, where each UAV can move independently from other UAVs and following the users' distribution. In this case is not necessary the use of a ground control center;
- swarm of drones without movement control, all UAVs move all together and we cannot control them;
- swarm of drones with movement control, all UAVs move all together and we decide the direction;

The main advantage of the use of a swarm of drones that move coordinately is that each node knows who its neighbours are and their position with respect to one node is constant.
In this way, each node has only to check periodically if a link exists between each neighbour and consequently update the routing table.
This scenario allows us to exploit all the advantages of georouting.
Since the idea is to use drones to guarantee a minimum users' rate this scenario is useful if we want to maximize the coverage area in a determined zone without considering users' distribution.
But, if we want to consider also the users' position, this model is not so efficient because there may be zones with a high number of users and, in this case, a drone can use more power than the others that become obsolete

because they do not serve any users.
For this reason and for the aim of this work the best solution is to consider a certain number of UAVs that move independently following the users' distribution. This is more efficient because we are able to move drones basing on our necessity such as maximize the number of covered users, minimize the consumed energy or maximize the transmission rate.
In general, we can say that this configuration is more suitable for our purposes because we can make better use of each UAV.

3.2 Topology formation

This work aims to implement two routing protocols that are optimal with respect to a specific metric and compare them with respect to end-to-end delay. One protocol aims to minimize the number of hops to reach the destination node, while the other one aims to maximize the data rate transmission.
Since we are considering an UAVs network, so aerial vehicles that are able to move with high speed, we want that these UAVs move following the users' distribution.
To reach this goal I proceeded by step.

First, I considered a static scenario with N UAVs acting as slaves and one UAV acting as master. The UAV that acts as master is the one closer to the Base Station (BS).
Every second the base station sends a packet containing the position of all UAVs in the network to the master that broadcasts this packet to all its neighbours using a sort of flooding.
This process ends when each node has updated its routing table and an ACK is sent to the master.
In this way, every second, we have an image of our network with all connections and positions.
The network has been modelled has a graph G = (V,E) where V is the set of UAVs (each UAV has x and y coordinates) and E is the set of link.
I stated that there is a link between two nodes if the received power P_{RX} is greater or equal than a certain threshold, P_{TH}.
At the beginning we considered only a simple model for the channel, i.e. an attenuation proportional to the square of the distance d and a noise component represented by α. As consequence, the received power of drone i from drone j is:

$$P_{RX}(i,j) = \frac{P_{TX}(j,i)}{d^2} + \alpha \tag{3.1}$$

where $P_{TX}(j,i)$ is the power transmitted by drone j to drone i and d^2 is the Euclidean distance between the two drones, computed as $d = \sqrt{(X_i^2 - X_j)^2 + (Y_i^2 - Y_j)^2}$. In figure (3.2) the best configuration is shown.

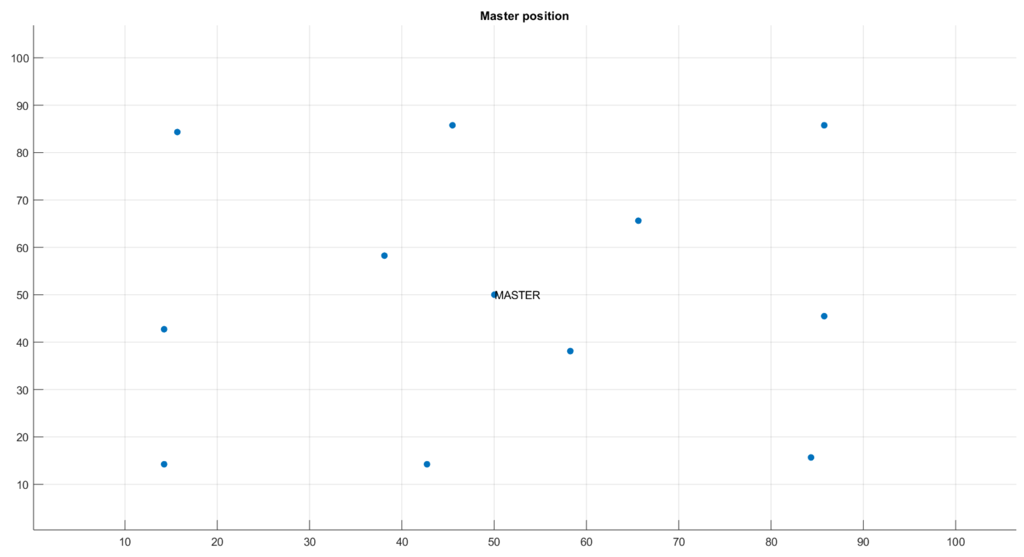


Figure 3.2: Best drones configuration.

As we can imagine this model is not efficient because the links change very fast and the routing table contains a lot of useless information because they may contain routes that are not available anymore.
A better solution would be to implement a distributed protocol where each node contains only needed routes and information about its neighbours.

So, in the next step we considered only N UAVs move randomly in the area. The use of a drone that acts as master is not necessary anymore.
As before, if the received power is greater or equal to a certain threshold there is a link between two nodes.
Every time that a node has a new neighbour or it loose one it updates its neighbours list.
Furthermore, if we suppose that the link that connect node A to node B disappears then node A removes from the routing table all the paths that have node B as next hop.
In Chapter 4 we will see in more details the mechanism.

3.2.1 Channel model

The design of an efficient wireless communication system requires an understanding of the radio propagation environment [8].
The characteristics of the channel depend by the operating frequency and the mode of propagation, e.g., *Line of Sight (LoS)*, diffraction, scatterers, satellite vs. terrestrial links.

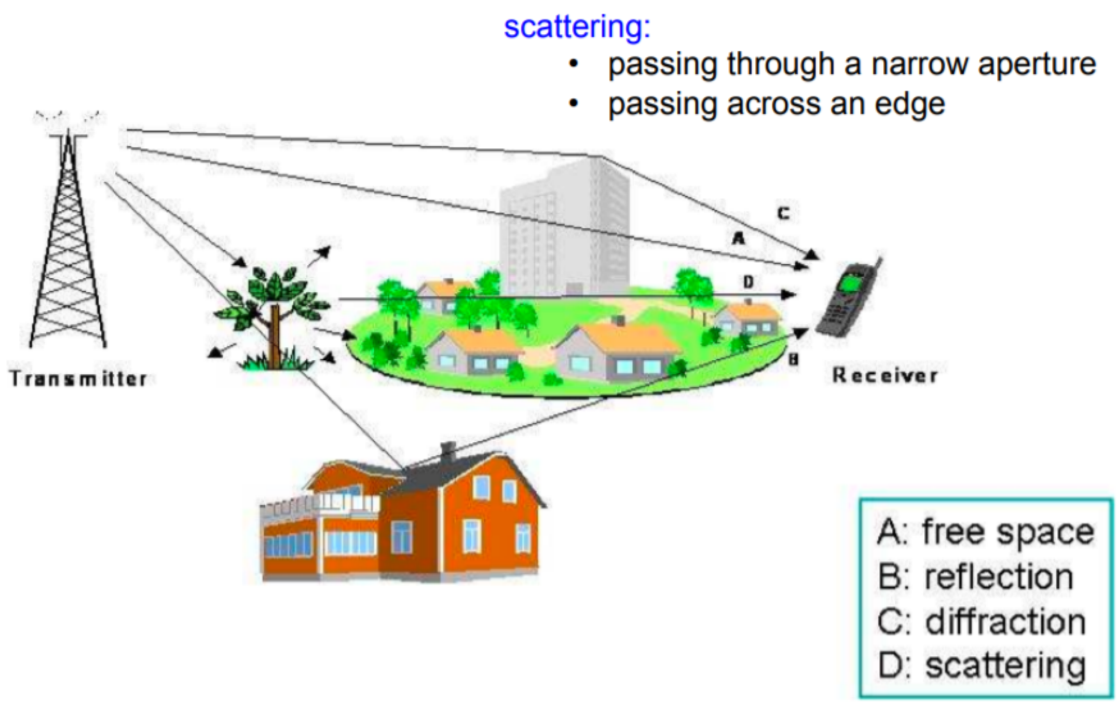
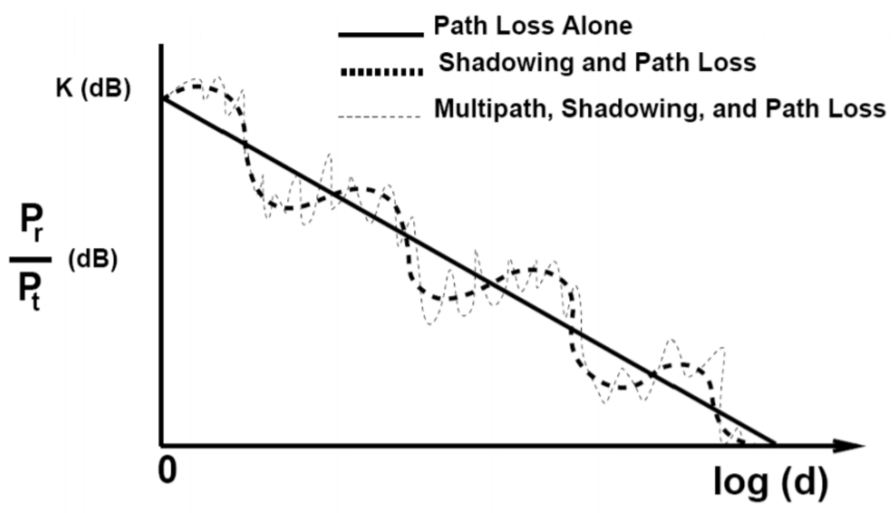


Figure 3.3: Radio propagation environment.

Channel attenuation depends by:

- *Path Loss (PL)*, caused by the dissipation of the power radiated by the transmitter (proportional to $\frac{1}{d^2}$);
- Shadowing, it is caused by obstacles between transmitter and receiver that attenuate signal power through absorption, reflection, scattering and diffraction;
- Multi-path fading, scatterers.



PL	→	variations over 100 -1000 meters, large scale
Shadowing	→	variations over 10 – 100 meters, medium scale
Multi-path	→	variations over very short distances (signal wavelength) small-scale propagation effects

Figure 3.4: Channel attenuation with respect to the distance.

We can distinguish between LoS and NLos channel.
LoS channel refers to the case where we have a strong direct signal with a number of weaker multipath echoes.
Indeed in a NLoS channel we don't have a strong direct signal but only a number of weaker components.
In this work the channel model that we considered is the following:

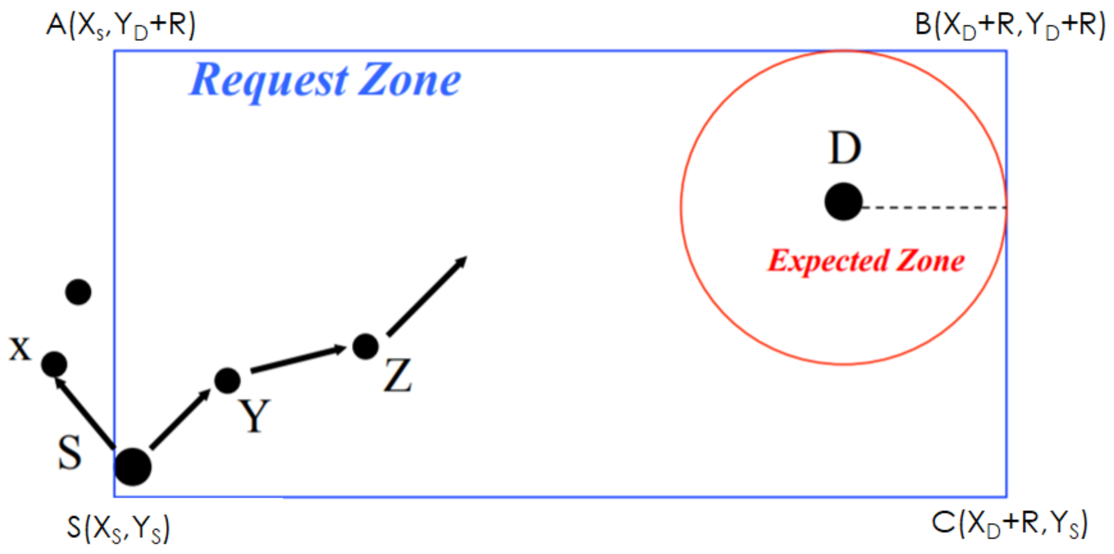
$$P_r\;[dBm] = P_t + 20\;log_{10}(G) + 10\;log_{10}\left(\frac{4\pi f_c d}{c}\right) - \phi \tag{3.2}$$

where

- P_r is the received power;
- P_t is the transmitted power;
- G is the antenna gain;
- f_c is the carrier frequency;
- d is the distance between two nodes;
- $\phi \sim N(0, \theta)$ is the shadowing term.

3.3 Location Aided Routing (LAR)

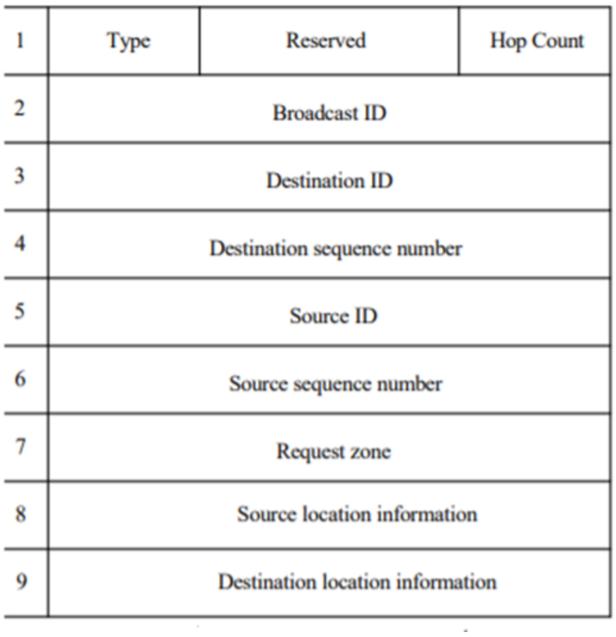
The next step to improve the quality of the work was to include georouting. The idea of georouting, as we said before, is to forward the packet to the node closer to destination (greedy forwarding) and, when a node is not founded, we use a recovery mechanism to find a node where greedy forwarding can restart.
Including georouting we reduce the traffic load.
My approach is based on LAR [9], that is similar to AODV and DSR. If a source node needs to transmit data to a destination D, it first looks its routing table. If there is a valid route to the destination it sends packet to the next hop otherwise a new route discovery starts. If we reach an intermediate node that does not know how to reach D, it sends a RERR to S and a new route discovery starts.



The difference between LAR and the others reactive protocols is the route discovery, let's how it works. First of all, the source node looks its routing table and if it knows the location of the destination node D at time t_0 it defines two zones:

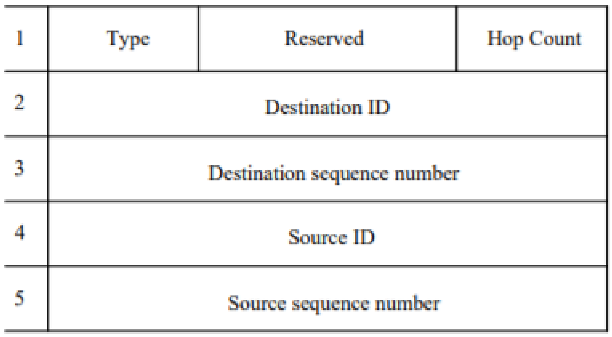
- a Request zone: a node forwards a RREQ only if it belongs to the request zone. It is defined as a smallest rectangle that includes both the current location of the source node and the expected zone. The coordinates of the four corners of the request zone are included in the packet;
- the Expected zone, the region that most likely contains the destination node. To calculate it, source node need to know the location (L) and the speed of destination (v) at time t_0. Assume that current time is t_1, the expected zone is a circular region of radius v(t_1-t_0), centered at location L.

Otherwise, if the source does not know the previous location of the destination, the RREQ is forwarded into the whole network.
Once the destination node is reached, it sends a RREP through a reversed path in which it includes: destination current location, system current time and destination current speed and direction.
The problem of LAR is that nodes need to know their physical locations.
We have to find a method that allows us a fast updating for what concern positions otherwise we have to start a complete route discovery increasing the traffic load.
When a source node needs to transmit data to a destination node for which it has not routing information it starts a route discovery broadcasting a RREQ packet [10] to its neighbours. The RREQ packet format is shown in the figure below.
The size of the packet is bigger than the one of AODV and DSR because it contains all informations about location such as Request zone and source and destination location.
This is valid also for the RREP. Since the main problem of georouting is the positions' update we decided to introduce some improvement.
Through RREP every intermediate node learns the location and the route of any others intermediate node. For example, if node A needs to send data to node E and the route is A-B-C-D-E then node B learns the route for E, C and D because RREP contains all information about these nodes.
In this way we need less time to find a valid route for a specific node because a route discovery is not necessary.



1	Type	Reserved	Hop Count
2	Broadcast ID		
3	Destination ID		
4	Destination sequence number		
5	Source ID		
6	Source sequence number		
7	Request zone		
8	Source location information		
9	Destination location information		

Figure 3.5: Example of RREQ format in LAR.



1	Type	Reserved	Hop Count
2	Destination ID		
3	Destination sequence number		
4	Source ID		
5	Source sequence number		

Figure 3.6: Example of RREP format in LAR.

Regarding the routing table they have the following fields:

- Destination node;
- Next hop;
- Destination position;
- Time, it represents the time instant at which the position is recorded;

Another further step, it would be to memorize also the direction of a node in order to know an approximate position at a certain instant t and define a request zone with constant size.
The problem is that the direction can change a lot of times from the instant that we record the position and the instant at which we send data.

Chapter 4

MATLAB implementation

MATLAB code is structured as follow.
In the first part we generate a random number of users. Each user has x and y random coordinates.
Then using one of the two algorithms we implemented we compute the optimal position of the UAVs in order to cover the maximum number of users.
Since at each iteration a new set of positions is computed, we stated that every time a drone moves towards the new closest position in order to consume less energy as possible.
Once we computed the new set of positions we have to define the topology of the network. As we already stated if the received power is greater or equal than a certain threshold there is a connection otherwise not.
At this point we update the routing table and the positions. So, we remove from the routing table all the paths whose next hop is not a neighbour anymore and we update the position of all other neighbours.
Once we did this, we choose a source and a destination node and we try to send data.
In the following sections it is explained how the functions send, RREQ, RREP and RERR related to the protocols have been implemented.

4.1 `send()` function

Algorithm 1 send (source,destination)

```
repeat
    S checks the routing table
    if S has a route for the destination then
        It sends data to the next hop C
        repeat
            C checks the routing table
            if C has a route for the destination D then
                it sends data to the next hop
            else
                it sends RERR to source S
                break;
            end if
        until destination D is reached
    else
        it starts a new route discovery
        repeat
            starting from S data are sent to next hop
        until destination D is reached
    end if
until destination D is reached
```

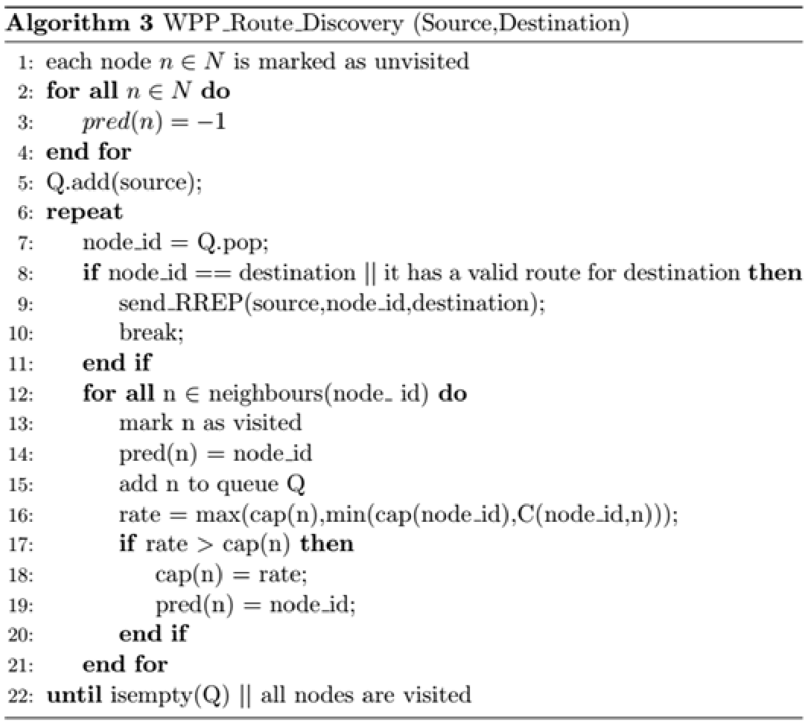
Figure 4.1: Send method .

This function implements what we have already discussed regarding how data are sent.
In practice, if a source node S needs to send data to a destination node D it first looks its routing table and if there is a valid route to destination it sends data to the next hop otherwise it starts a new route discovery.
It can happen that the source node has a valid route to destination but an intermediate node does not know how to reach D and, in this case, it sends a RERR to the source and a new route discovery starts. This process ends when a valid route is found or there is not a valid path.

4.2 `route_discovery` function

The difference between the two protocols is in route discovery because we have to optimize different metrics and so there is a different way to find the best route.

First, we discuss the route discovery that allow us to find the route with maximum capacity.

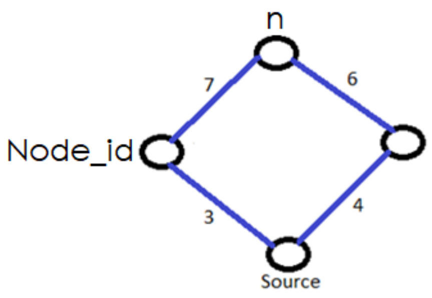


Algorithm 3 WPP_Route_Discovery (Source,Destination)

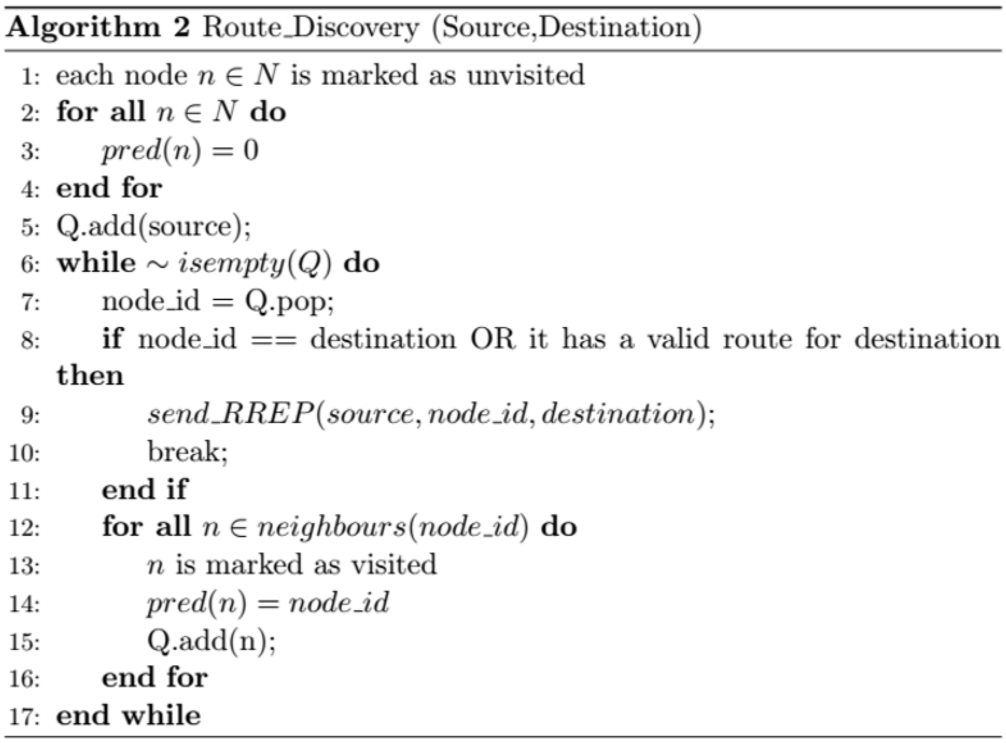
```
each node $n \in N$ is marked as unvisited
for all $n \in N$ do
    $pred(n) = -1$
end for
Q.add(source);
repeat
    node_id = Q.pop;
    if node_id == destination || it has a valid route for destination then
        send_RREP(source,node_id,destination);
        break;
    end if
    for all n $\in$ neighbours(node_ id) do
        mark n as visited
        pred(n) = node_id
        add n to queue Q
        rate = max(cap(n),min(cap(node_id),C(node_id,n)));
        if rate > cap(n) then
            cap(n) = rate;
            pred(n) = node_id;
        end if
    end for
until isempty(Q) || all nodes are visited
```

Figure 4.2: Route discovery for maximizing capacity.

First of all, each node is marked as unvisited and the predecessor is set to -1. Predecessor is an array where the entry i contains the previous node of the path from source to node i. Then, we add the source node to a queue. At this point we extract the first element of the queue (we call this element node_id) that, at the first iteration, will be the source node. So, if node_id is the destination node or it has a valid route to destination it sends a RREP to the source otherwise I consider each of its neighbours n. I mark n as visited, set the predecessor of n equal to node_id and add n to the queue.
I defined a vector cap where cap(n) represents the maximum minimum capacity of the path from the source node to n. To update this value, I choose the maximum between cap(n) and the minimum between cap(node_id) and the capacity of the link that connects node_id to n.
Let's see an example:



In this case, suppose we are considering node_id; we have that cap(node_id) is equal to 3. Then, the capacity of node n will be equal to the maximum between cap(n) and the minimum between cap(node_id) and the capacity of the link that connects them. So, cap(n) will be equal to 3.
Now, let's see how the route discovery that minimize the number oh hops works.
As before we mark each node as unvisited, set the predecessor equal to -1 and add the source to a queue.
Then we extract the first element of the queue (we call it node_id) and if this node is the destination node or it has a valid route to destination it sends a RREP.
If it is not, we consider all neighbours n of node_id and for each node n we mark it as visited, set the predecessor equal to node_id and add it to the queue.
We proceed in this way until we reach the destination or a node that has a valid route to it.



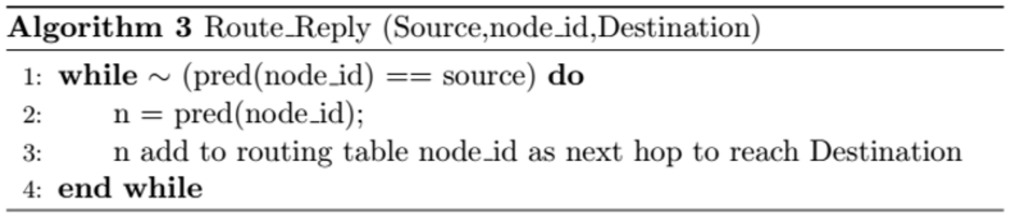
Algorithm 2 Route_Discovery (Source,Destination)

```
each node $n \in N$ is marked as unvisited
for all $n \in N$ do
    $pred(n) = 0$
end for
Q.add(source);
while $\sim isempty(Q)$ do
    node_id = Q.pop;
    if node_id == destination OR it has a valid route for destination then
        $send\_RREP(source, node\_id, destination)$;
        break;
    end if
    for all $n \in neighbours(node\_id)$ do
        $n$ is marked as visited
        $pred(n) = node\_id$
        Q.add(n);
    end for
end while
```

Figure 4.3: Route discovery to minimize number of hop.

4.3 `route_reply` function

Route reply has been implemented in a very simple way using the vector predecessor that we created during the route discovery.
So, starting from the destination or from the node that knows how to reach it at each step we visit the predecessor of the considered node (we call it node_id) and we update the routing table adding node_id as next hop to reach the destination node. We continue in this way until we reach the destination.



Algorithm 3 Route_Reply (Source,node_id,Destination)

```
1: while ~ (pred(node_id) == source) do
2:     n = pred(node_id);
3:     n add to routing table node_id as next hop to reach Destination
4: end while
```

Figure 4.4: Implementation of Route Reply method.

About the route error we worked in the same way, the only difference is that, at each step, instead of adding node to the routing table we remove them because, since we are sending RERR it means that a route is not available anymore.

4.4 Results

The parameters used for simulations are represented in the table below.

Parameters	Value
P_{TX} (transmission power)	35 dBm
B (bandwidth)	40 MHz
f_C (carrrier frequency)	2 GHz
G (antenna gain)	$29000/80^2$ [dB
C (light speed)	$3 \cdot 10^8$ m/s
Ψ (Shadow fading)	N $(0, \sigma^2)$ with $\sigma = 6$
Area	5 Km x 5 Km
UAV-UAV TX Range	2.5 Km
Number of UAV	10 - 15 - 20 - 25 - 30 - 35 - 40
n (path loss exponent)	2

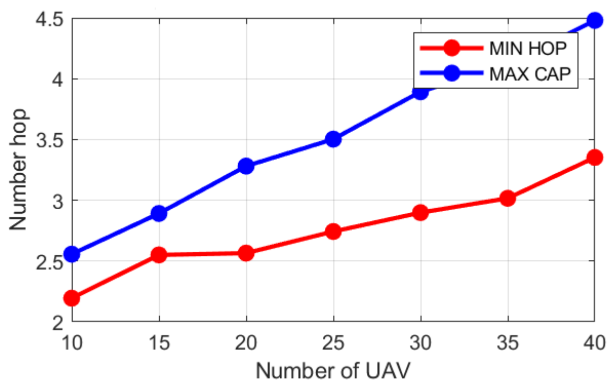


Figure 4.5: Number of hop with respect to number of UAVs.

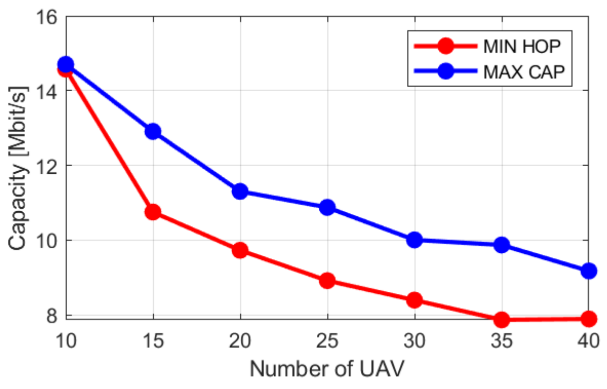


Figure 4.6: Transmission rate with respect to number of UAVs.

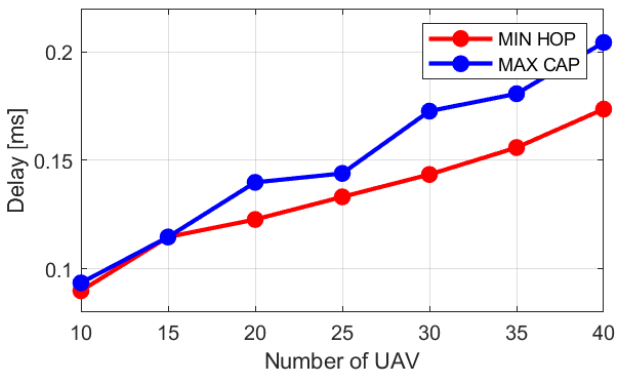


Figure 4.7: End-to-end delay with respect to number of UAVs.

Figure (4.5) shows the mean numbers of hops to reach the destination while figure (4.6) shows the mean transmission rate to reach the destination. Both graphs are with respect to the number of UAVs in the network.
As expected, each protocol works better than the other one with respect to the metric that it optimizes. The interesting thing is that the difference in the performance become more evident when the number of UAVs in the network increases.
The two protocols have been compared using end-to-end-delay as comparison metric. Delay has been defined as the time since the sender node checks the routing table to see if there is or not a valid route to destination to the time that the packet reaches the destination node.
The necessary time to transmit a packet from node A to node B has been computed as:

$$t = \frac{L}{R} \tag{4.1}$$

where L is the packet length and R is the rate of the link that connects node A to node B.
For this reason we had to define the size of the packet used for route request, route reply, route error and data transferred.
So, after some research we decided to use the following values:

- RREQ: 36 bytes;
- RREP: 20 bytes
- RERR: 20 bytes;
- DATA: 512 bytes;

From this point of view, as figure (4.7) shows, the algorithm that minimize the number of hops is better because route discovery is faster since each node does not forward another RREQ with the same ID of the previous one.
Instead, if we want to maximize the capacity, we have to consider all links and it requires a lot of time.
Even if we did not simulate and computed the traffic load we can say that also from this point of view the algorithm that minimizes the number of hops is better.
In the following plot is represented how the number of hops to reach destination varies changing the number of drones in the network and the transmission range.

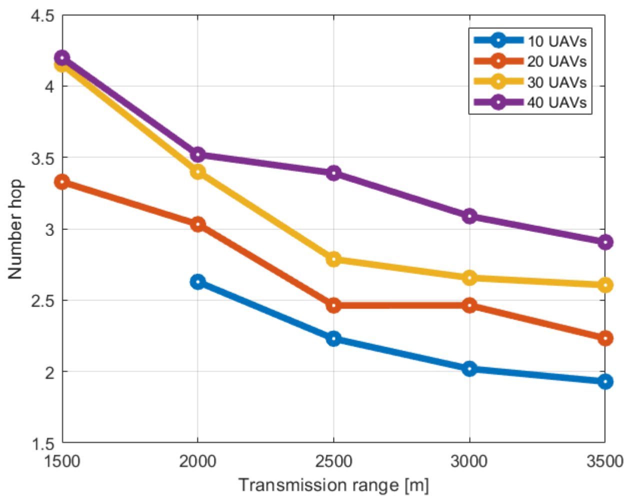


Figure 4.8: Number hop with respect to transmission range.

The first thing that we notice is that using 10 drones with a transmission range of 1500 m is not good because there are not enough links.
Looking the graph, we can observe that for a given number of drones the mean number of hop decrease as the transmission range increase. This is an expected result since increasing the transmission range increase also the number of neighbours and it is more probably that there is a shortest path.

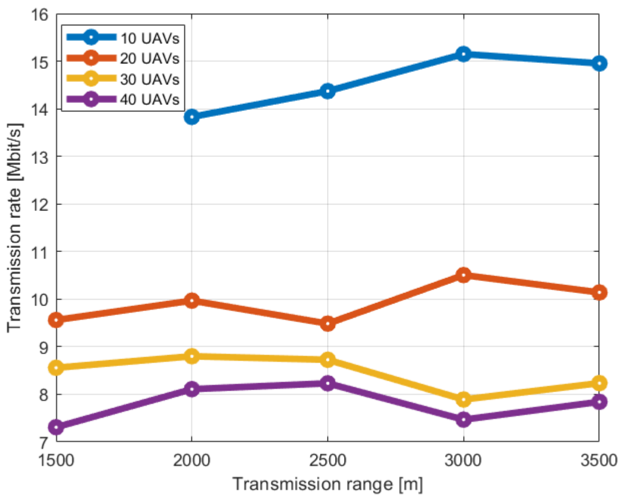


Figure 4.9: Transmission rate with respect to transmission range for MIN HOP protocol.

If we consider the transmission rate we can see that it is almost constant, the transmission range does not affect this metric too much. Only changin the number of UAVs the transmission rate decreases due to interference.

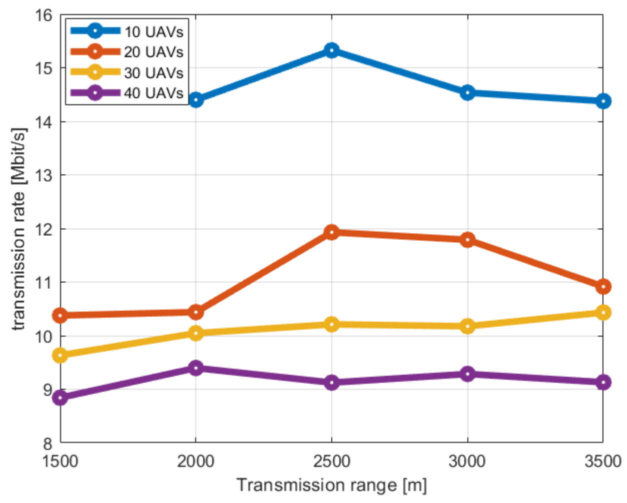


Figure 4.10: Number hop with respect to transmission range for MAX CAP protocol.

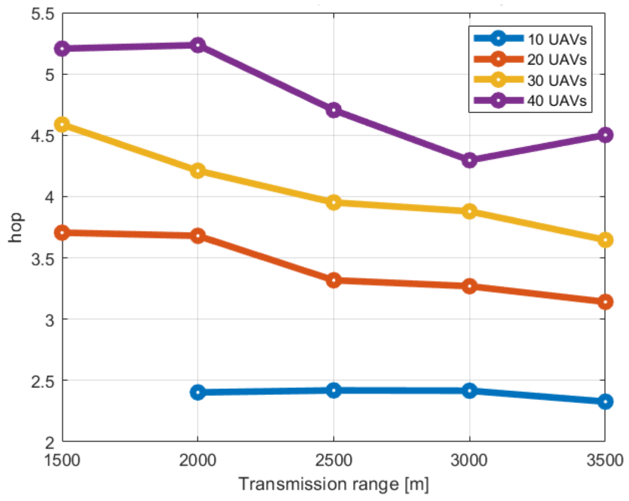


Figure 4.11: Transmission rate with respect to transmission range for MAX CAP protocol.

In the graph above is represented how varies the transmission rate changing the number of UAVs in the network and the transmission range for the protocol that maximizes the transmission's capacity.
As we can notice, given a fixed number of UAVs and changing the transmission range, the capacity remains almost constant while the mean number of hops decrease a little bit.

Instead, using a high number of UAVs the capacity decreases because increases also the interference of other UAVs.
Definitely, if we want to use the protocol that maximizes the capacity the best solution is to use the minimum number of UAVs with the maximum transmission range.
Remember that in this work is not considered the consumed energy. In this further case this would not be the best solution because it means that the coverage range and the distance between UAVs are bigger and so the consumed energy increases.

At this point, we are able to understand how much the power affect the project of a network because one of the goals is to increase the life time of an UAV. Once we modelled the consumed energy, we can say which is the better solution based on all data we got.

Chapter 5

Optimal UAVs deployment

The second part of this thesis regards the optimal UAVs deployment based on the users' position. This challenge finds application in many scenarios.
For example, as we have already said UAVs can be used to inspect a certain zone, capture some data (i.e. videos or images) and send them to a control center.
Or, in case of disaster, we can suppose that a base station is damaged and we can place a drone to restore connectivity. Again, we can use them in rural area that cannot be reach using the base stations.
Another application is as support for base stations if the traffic load is too high, such as at the stadium when we watch a concert or in city centre.
In this case is also necessary to analyse if it is useful to place a drone because otherwise, we consume a lot of energy and this is useless.
In all this application is necessary to know the position of the users.
The algorithms for the optimal deployment can be divided in two categories: the ones that maximize the coverage area and the ones that maximize the number of covered users.
In the first case, given a certain number of drones, we have to find the maximum radius and the position of each drone in order to maximize the coverage area.
Most of these algorithms try to solve the circle packing problem, i.e., to deploy N circles inside a given surface such that the packing density is maximized and none of the circles overlap.
In [11] a heuristic algorithm has been implemented. Heuristic means that the solution is built step by step.
This algorithm is based on the concepts of feasible position and damage.
Feasible position means that if we center a circle at that coordinates there is not overlapping. Indeed, damage is defined as the number of positions that becomes infeasible once we placed an UAV.

So, at each step a circle is placed at the location where the damage is minimized.
In the second category, algorithms have to take into account also the users' distribution. In [12] has been implemented an algorithm that minimize the total required transmit power of UAVs while satisfying the users' rate requirements.
To this end, the optimal locations of UAVs as well as the cell boundaries of their coverage areas are determined.
To find those optimal parameters, the problem is divided into two sub-problems that are solved iteratively.
In the first sub-problem, given the cell boundaries corresponding to each UAV, the optimal locations of the UAVs are derived using the facility location framework. In the facility location problem, given sets of facilities and clients, the goal is to find the optimal placement of facilities to minimize total transportation costs between the clients and facilities. Here, UAVs are considered as the facilities and users as the clients.
In the second sub-problem, the locations of UAVs are assumed to be fixed, and the optimal cell boundaries are obtained using tools from optimal transport theory.
In optimal transport theory we supposed that piles of sand and some holes with the same total volume of sands are randomly distributed over a geographical area.
The objective is to find the optimal moves to transport the entire piles to the holes with a minimum transportation cost. The optimal moves depend on the cost function which is a function of distance between pills and holes.
The results show that the total required transmit power is significantly reduced by determining the optimal coverage areas for UAVs.
These results also show that, moving the UAVs based on users' distribution, and adjusting their altitudes can lead to a minimum power consumption.

5.1 Scenario

We supposed to have a set of N users uniformly distributed and each user has been represented with x and y coordinates.

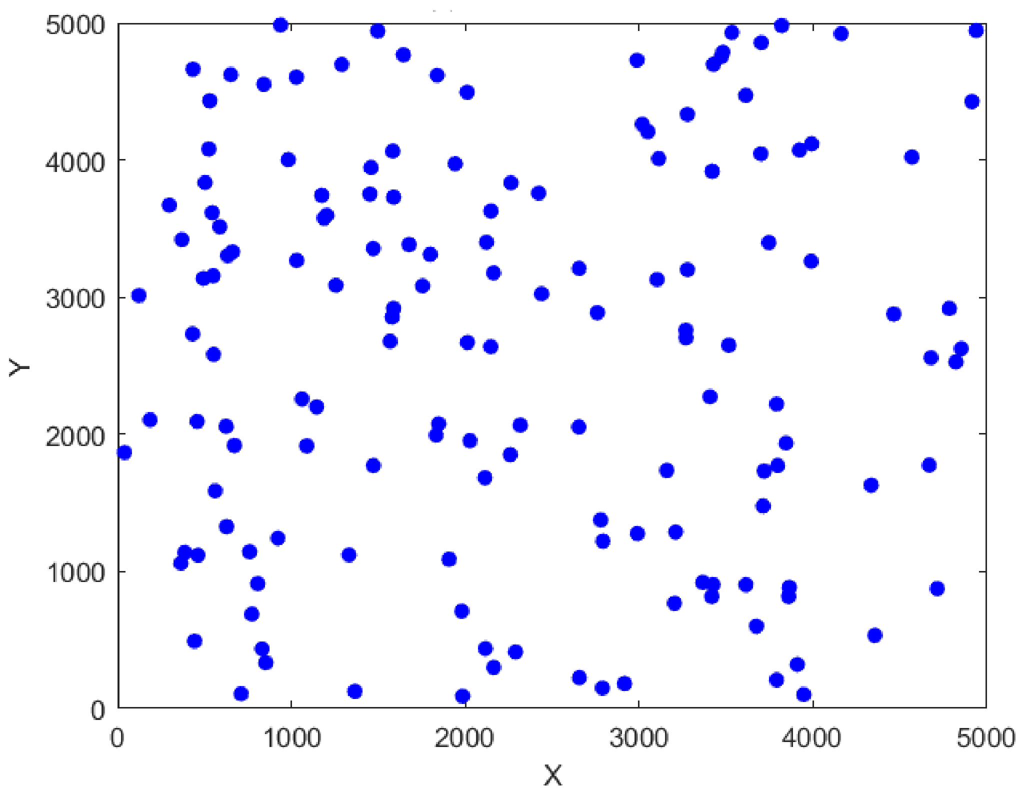


Figure 5.1: Users' distribution.

Then, we supposed two cases: in the first case we know how many drones we have and we place them in order to maximize the number of covered users; while, in the second case, we don't know how many drones we need and so, through density computation, we are able to determinate it.
At the end, we implemented two algorithm and we compared them. One algorithm solves an optimization problem with constraints using the function fmincon ([13] and [14]) in MATLAB while, the other one, solve the problem through convolution only using matrices and filters.
But let's proceed step by step.

5.1.1 How many drones do we have?

This problem has been seen as a clustering problem. Given a set of points belonging to the Euclidean space, with a notion of distance between points, clustering aims at grouping them into a number of subsets (clusters) such that:

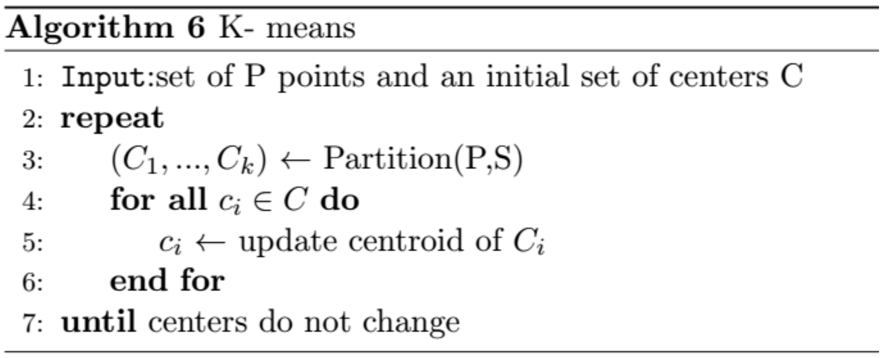
- Points in the same clusters are "close" to one another;
- Points in different clusters are "distant" from one another;

The distance represents the concept of similarity. A clustering problem optimizes a given objective function. Some of the most common objective function are:

- $\Phi(C) = max_{i=1}^{k} max_{a \in C_i} d(a, C_i)$
- $\Phi(C) = \sum_{i=1}^{k} \sum_{a \in C_i} (d(a, C_i))^2$
- $\Phi(C) = \sum_{i=1}^{k} \sum_{a \in C_i} d(a, C_i)$

We know how many drones we have

If we know how many drones we have we apply a simple k-means algorithm.



Algorithm 6 K- means

```
1: Input:set of P points and an initial set of centers C
2: repeat
3:     $(C_1, ..., C_k) \leftarrow$ Partition(P,S)
4:     for all $c_i \in C$ do
5:         $c_i \leftarrow$ update centroid of $C_i$
6:     end for
7: until centers do not change
```

Figure 5.2: k-means algorithms' implementation.

Algorithm 7 Partition(P,S)

```
1: Input: set of N points and an initial set of centers
2: for all p ∈ P − S do
3:     $l \leftarrow argmin_{i=1,k}$ d(p,$C_i$)
4:     $C_l \leftarrow C_l \cup p$
5: end for
```

Figure 5.3: partition methods' implementation.

K-means algorithm receives as input a set of points P and a set of initial clusters C. Then, using the method Partition, we assign each point to the cluster of its closest center.
At this point we update the centroid of each cluster. Centroids are computed using the following formula:

$$c(P) = \frac{1}{N} \sum_{X \in P} X \tag{5.1}$$

We proceed in this way until the set of centers does not change. The quality of the solution and the speed of convergence depend by the initial set of centers. Consider this example:

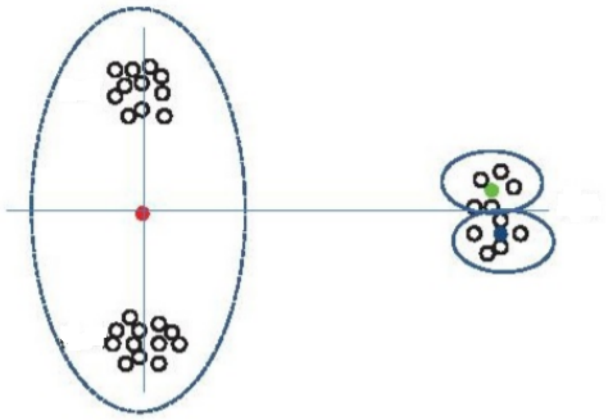
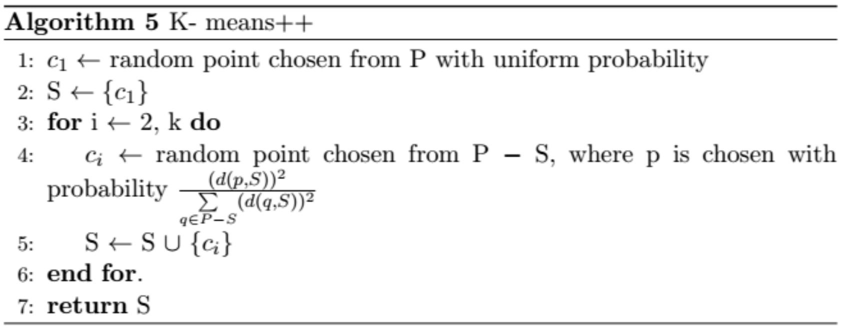


Figure 5.4: Clustering with wrong set of initial centers.

If initially one center falls among the points on the left side, and two centers fall among the points on the right side, it is impossible that one center moves from right to left, hence the two obvious clusters on the left will be considered as just one cluster.
For this reason we used kmeans++ that provide us a good initial set of centers. In this way, we obtain a clustering that is not too far from the optimal solution.



Algorithm 5 K- means++

```
1: $c_1 \leftarrow$ random point chosen from P with uniform probability
2: S $\leftarrow \{c_1\}$
3: for i $\leftarrow$ 2, k do
4:     $c_i \leftarrow$ random point chosen from P $-$ S, where p is chosen with
       probability $\frac{(d(p,S))^2}{\sum\limits_{q \in P-S} (d(q,S))^2}$
5:     S $\leftarrow$ S $\cup \{c_i\}$
6: end for.
7: return S
```

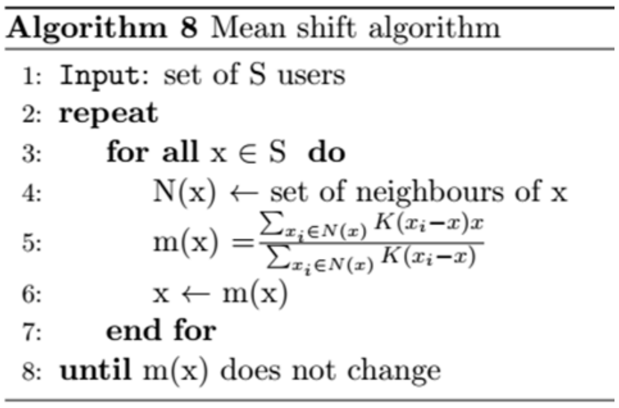
Figure 5.5: k-means++ algorithm's implementation.

As we already said k-means++ allows us to find a good initial set of centers. The first center is a random point chosen from the set of all points P with uniform probability and adds to the set of centers S.
Now, at each iteration the next center is chosen from the set P - S with probability $\frac{(d(p,S))^2}{\sum\limits_{q \in P-S} (d(q,S))^2}$ and we continue in this way until we get k centers.

We do not know how many drones we have

If we do not know how many drones we need we use mean - shift theory that allows us to find the densest zone and, consequently, the number of necessary drones. This method uses kernels.
Kernel Density Estimation (KDE) [15] is a non parametric technique for density estimation. Non parametric means that it is not based on parametrized families of probability distribution where common parameters are the mean and variance.
It can be viewed as a generalisation of histograms.
When we use histograms the results depend by the choice of the anchor point (the lower left corner of the histogram grid) and for this reason not always they provide a good solution.
Instead, using kernel, the results do not depend by the grid because we simply do a weighted sum based on the distance from the kernel's center.

In the figure we can see how the algorithm works.



Algorithm 8 Mean shift algorithm

```
1: Input: set of S users
2: repeat
3:     for all x ∈ S do
4:         N(x) ← set of neighbours of x
5:         m(x) = $\frac{\sum_{x_i \in N(x)} K(x_i - x)x}{\sum_{x_i \in N(x)} K(x_i - x)}$
6:         x ← m(x)
7:     end for
8: until m(x) does not change
```

Figure 5.6: Mean shift algorithm's implementation.

First of all, for each point we consider a set of neighbours. The neighbours of a point are all points at a distance less or equal to a specified threshold. Then we perform a sort of weighted mean using a Gaussian kernel on the distance. Gaussian kernel has the following expression:

$$K(x - x_i) = \frac{1}{\sqrt{2\pi}\sigma} exp(-\frac{||x - x_i||^2}{2\sigma^2}) \tag{5.2}$$

In formula (5.2) σ represents the variance and we want to choose it as small as possible but we have to take into account that there is always a trade-off between the bias of the estimator and the variance itself. There are some algorithms used for the computation of the correct value of the variance.
A range of kernel functions are commonly used: uniform, triangular, biweight, triweight, Epanechnikov, normal, and others.
Then we are able to compute the value m(x). We continue in this way until the values of m(x) don't change anymore. The difference m(x)- x represents the mean shift.
It is not guarantee that the algorithm converges, for this reason the algorithm run for a defined number of iteration.

5.2 1^{st} algorithm: optimization problem

The first algorithm that we implemented for an optimal UAVs deployment solve the following optimization problem:

$$max \sum_{u_i \in U} u_i \tag{5.3}$$

s.t.

$$||u_i - C_k|| \leq R \tag{5.4}$$

$$||C_j - C_k|| \geq 2R \tag{5.5}$$

where U is the set of users and u_i a specified user. Constraint (5.4) states that a user is covered if the distance to its closest center is less ore equal to the coverage radius of the UAV. If the user i is covered then $u_i = 1$.
Constraint (5.5) states that the distance between two UAVs must be greater or equal than 2R in order to avoid overlapping.
This problem has been solved using the MATLAB function fmincon through Lagrangian multipliers. This function receives as input a set of constraints that can be linear on non-linear.

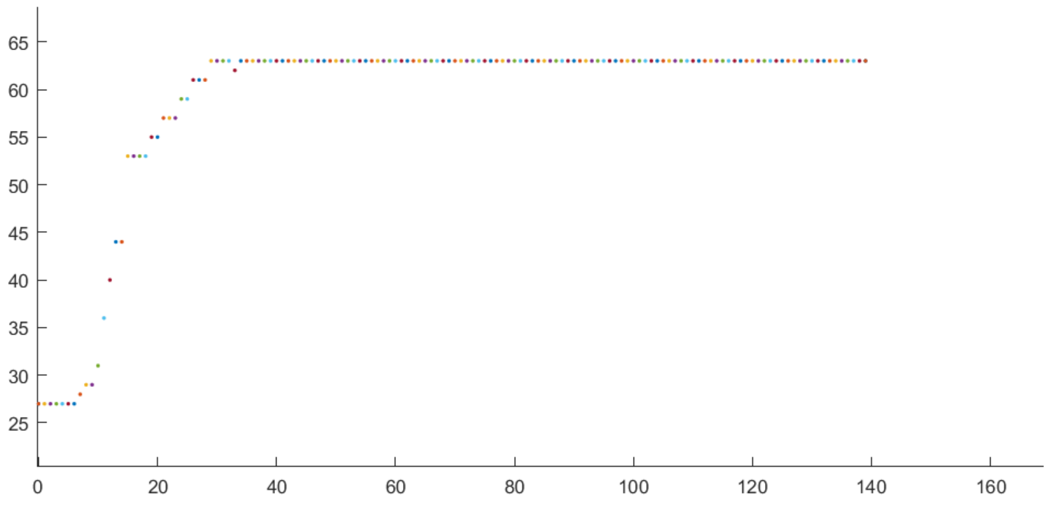


Figure 5.7: Covered users at each iteration.

In figure is represented the number of covered users at each iteration. As we can see the number increases at each step. fmincon() receives as input also the minimization function. In our case the minimization function is:

$$\min \sum_{i=1}^{k} \sum_{a \in C_i} (d(a, C_i))^2 \tag{5.6}$$

As we have already said, the performance of fmincon() depend by the set of initial centers that we provide as input and it is for this reason that we used clustering.
fmincon and clustering aim to minimize the same objective function.
The output set of centers from clustering is the input of fmincon and so also the objective function is the starter point of the function that fmincon minimizes. The cost function of fmincon increases with respect to the one provided as input because in order to satisfy all constraints the position of the centers needs to change. In particular, we want to maximize the number of covered users avoiding overlap and this is what causes this increase.

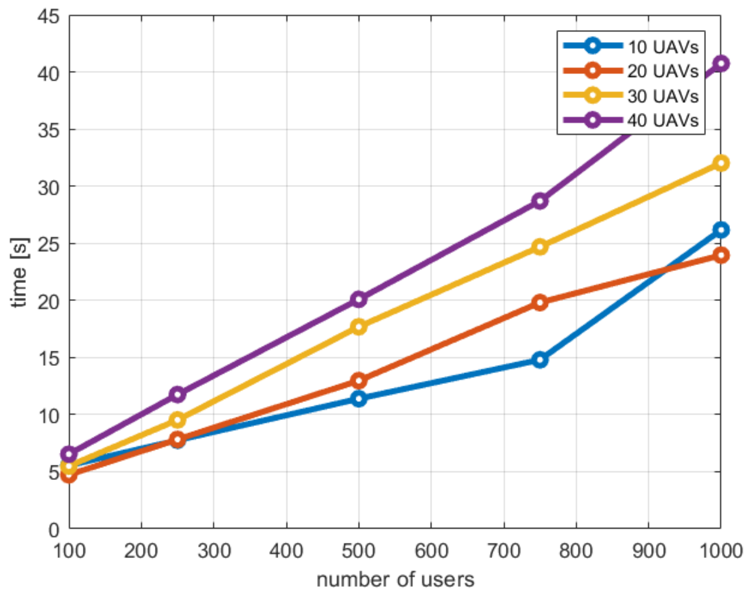


Figure 5.8: Processing time with respect to number of users.

As we can see in figure (5.8) the performance of the algorithm increases as the number of users and drones increase.
This is due to the fact that we add further constraints and this requires more time to find a solution. In fact, increasing the number of users increase also the time necessary for the k-means++ to find a good set of initial centers and also the function fmincon needs more time because there are others constraint to satisfy.
But, the thing that mainly affects the performance of the algorithm is the number of UAVs because the number of iteration of k-means increases exponentially with respect to the number of UAVs and, also, the number of constraints increases. Remember that we have to satisfy the constraints about the minimum distance between two centers and the distance of a user from the closest center. So, also the number of iterations to find the closest center increases exponentially.

5.2.1 fmincon

In general, an optimization problem is described as:

$$\min_x f(x) \tag{5.7}$$

subject to

$$G_i = 0 \tag{5.8}$$

$$G_i \leq 0 \tag{5.9}$$

fmincon is a function that tries to solve an optimization problem using *Karush-Kuhn-Tucker (KKT)* equations.
These equations are a necessary conditions to find an optimal solution of a constrained optimization problem.
If the problem is a so-called convex programming problem, that is, f(x) and G_i(x), i = 1,...,m, are convex functions, then the KKT equations are both necessary and sufficient for a global solution point.
The Kuhn-Tucker equations can be stated as:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \cdot \nabla G_i(x^*) = 0 \tag{5.10}$$

$$\lambda_i \cdot G_i(x^*) = 0 \qquad \text{i} = 1,...,m_e \tag{5.11}$$

$$\lambda_i \geq 0 \qquad \text{i} = m_e + 1,...,\text{m} \tag{5.12}$$

The first equation represents the difference of the gradients between the objective function and the active constraints at the solution point.

In order to obtain zero as result, Lagrange multipliers (λ_i, i = 1,...,m) have to balance the difference in magnitude of the two terms.
Lagrange multipliers of the constraints that are not included in the equation are equal to 0. The solution of the KKT equations forms the basis to many nonlinear programming algorithms.
These algorithms attempt to compute the Lagrange multipliers directly.
For example, quasi-Newton methods guarantee superlinear convergence using second-order information regarding the KKT equations.
In general, these methods are commonly referred to as *Sequential Quadratic Programming (SQP)* methods since, at each iteration, a QP subproblem is solved.

5.2.2 Sequential Quadratic Programming (SQP)

SQP methods represent the state of the art in non-linear programming methods. The method fmincon allows to imitate Newton's method for constrained and unconstrained optimization.
At each major iteration, the Hessian of the Lagrangian function is approximated using a quasi-Newton updating method.
This is then used to generate a QP subproblem whose solution is used in a line search procedure to find the direction along which the objective function is reduced and then compute the step size that determines how far **x** should move along that direction.
The general method is stated here. Given the problem description in GP (Equation 5.7) the principal idea is the formulation of a QP subproblem based on a quadratic approximation of the Lagrangian function.

$$L(x,\lambda) = f(x) + \sum_{i=1}^{m} \lambda_i \cdot g_i(x) \tag{5.13}$$

5.2.3 Quadratic Programming (QP) Subproblem

$$\min_{d\in\mathbb{R}^n} \frac{1}{2} d^T H_k d + \nabla f(x_k)^T d \tag{5.14}$$

$$\nabla g_i(x_k)^T d + g_i(x_k) = 0 \qquad \text{i=1,...,}m_e \tag{5.15}$$

$$\nabla g_i(x_k)^T d + g_i(x_k) \leq 0. \qquad \text{i=}m_e\text{+1,...,m} \tag{5.16}$$

This subproblem can be solved using any QP algorithm. The solution is used to form a new iterate

$$x_{k+1} = x_k + a_k d_k \tag{5.17}$$

The step length parameter α_k is determined by an appropriate line search procedure so that a sufficient decrease in a merit function is obtained.
The matrix H_k is a positive definite approximation of the Hessian matrix of the Lagrangian function (Equation 5.13). H_k can be updated by any of the quasi-Newton methods, although the quasi-Newton approximation method appears to be the most popular. A non-linearly constrained problem can often be solved in fewer iterations than an unconstrained problem using SQP. One of the reasons for this is that, because of limits on the feasible area, the optimizer can make informed decisions regarding directions of search and step length.

5.3 2^{nd} algorithm: density map and filters

Algorithm 9 Sub-Optimal deployment

```
area = zeros(size);
drones = zeros(size);
for all $user \in users$ do                  ▷ each user has x and y coordinates.
    area(x,y)=1;
end for
gaussian_filter(size,variance);
filter_UAV = ones(2*radius,2*radius);
repeat
    D = filter * drones;                     ▷ D is a submatrix of drones
    M = gaussian_filter * area;              ▷ M is a submatrix of area
    index = max(M);                          ▷ x and y coordinates of densest zone
    repeat
        if drones(idx) == 1 then             ▷ if there isn't a drone
            esagono(idx) = 1;                ▷ I place a drone
        end if
    until UAV is placed
until all UAVs are placed
```

Figure 5.9: Pseudo code of the algorithm.

We considered two matrices with size equals to the simulation area: a matrix is used for the users (we call it users) while the other one is used for UAVs (we call it drones). These matrices are initialized with all zeros and, considering the matrix users, I assumed that the entry (x,y) is equal to one if there is an user located at position (x,y).
Then we create two filters: a Gaussian filter and a simple mask made of all ones.

To create a Gaussian filter I created a mesh grid of the desired size where each entries of the matrix is equal to:

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{5.18}$$

Then, to normalize each entry I divided the matrix for the maximum value on it.

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Figure 5.10: Example of Gaussian filter and mask of all ones.

Then we perform a sort of convolution (because it's more a cross correlation) between the Gaussian filter and the matrix users.

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau \tag{5.19}$$

Convolution [16] is defined as the integral of the product of the two functions after one is reversed and shifted. We are not doing this because we are simply doing the inner product between two matrices, functions are not reversed.
We also define a stride, a step size that allow us to slide through the matrix. Stride equals to one means that the filter slides through the matrix entry by entri, while stride equals to two means that the filter slides through the matrix by moving 2 entries per step. We are performing a sort of down-sampling.
If we do this using machine learning, computing the convolution or the cross-correlation is the same because the filter's coefficients are automatically computed during training and, for this reason, they are equal to the ones obtained reversing the function.
In this way we obtain a sub matrix where each entry is a number much higher if the zone is dense. So, we obtain something similar to a density map.
We do the same for drones matrix; we do not use a Gaussian filter because

we do not need a density map but only to know if we already placed a drone. For example, when there is perfect overlap between the filter and the zone occupied by a drone, we have the highest possible value; while, if there is not overlapping, we obtain zero.
At this point we take the highest value from the submatrix obtained from users (it represents the densest zone, i.e. the zone with the highest number of users) and, since we also know the x and y coordinates, we check in the UAV matrix if we already placed a drone in that zone.
We can place a drone in that position only if the value of the entry (x,y) is zero otherwise it means that there would be overlapping.
If we cannot place a UAV there, we look for the next densest zone. We continue in this way until all drones are placed.
When we placed a drone we marked each entry of the circumscribed hexagon to the circle of radius R equals to one.
The problem of placing a defined number of drones in a rectangular area, as we already said, is called circle packing problem and the results say that the best configuration is obtained if we marked the occupied area with a hexagon.
Regarding the performance of this algorithm, speed and efficiency depend by the stride, the size of the filters and size of the area.

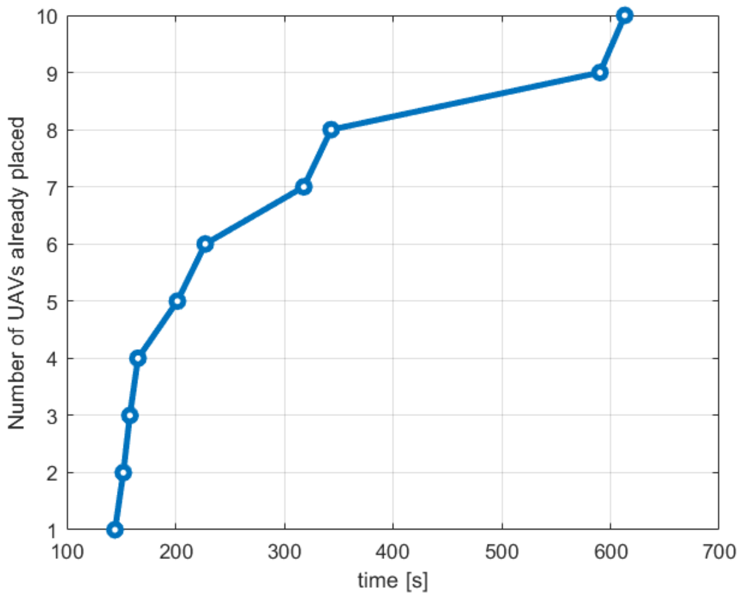


Figure 5.11: Instant when drones are placed.

The problems of the algorithm are: it requires time to apply the filter to the whole matrix and only after some trials we are able to find a trade-off between time and accuracy. The second problem is that the algorithm is quite fast to find the densest zone, in fact we can see in figure (5.11) that the first circles are placed in not too much time but then, the last circles requires more time because it has to be placed nearly occupied zone and so we have to search in the matrix an available position. This step requires too much time. In figure (5.12) is represented the necessary time to perform convolution with respect to the step size. As we can see the time decreases in an exponential way but increasing the step size decreases also the accuracy. For this reason we need to find a trade-off.

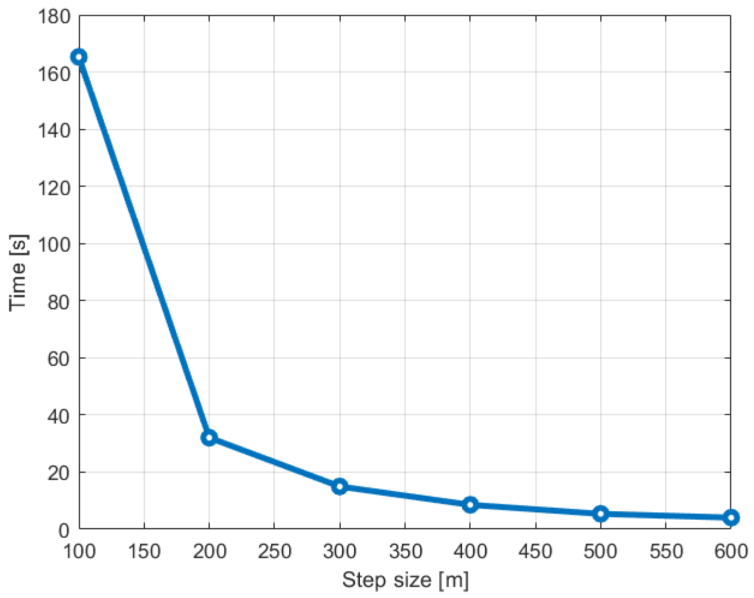
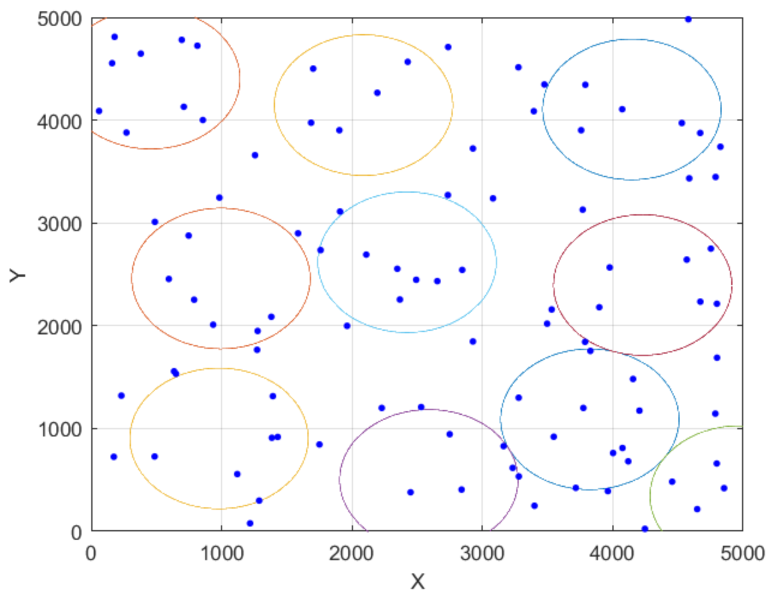


Figure 5.12: Time required for convolution.

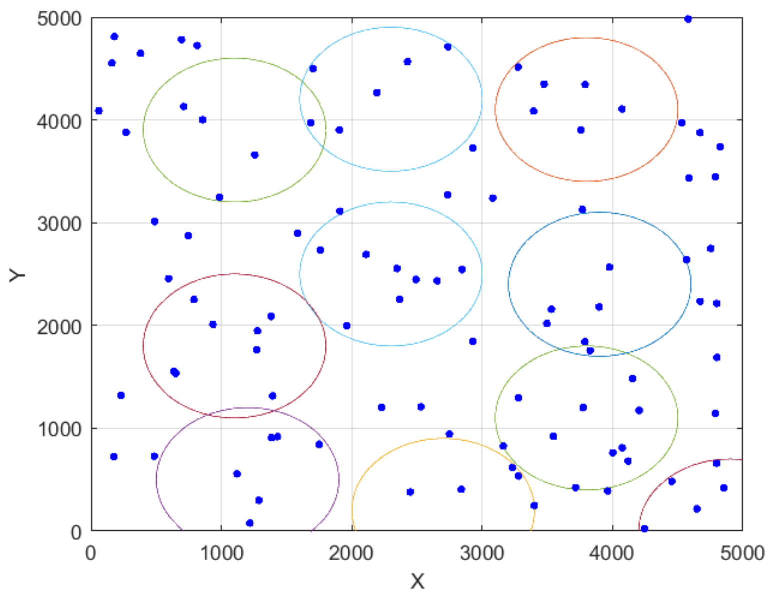
5.4 Comparison

The two algorithms have the same objectives but they proceed in different ways.
Considering the first algorithm (the one that uses fmincon) we can see that the solution depends by the set of initial centers that we provide as input. For this reason, before applying fmincon, we have to perform clustering and subsequently we are able to optimize the solution.

Certainly, we cannot obtain an optimal solution because it is not possible to cover all users but only 70 - 80 %. The main disadvantage of this algorithm is that, at each instant, it provides the best configuration. We do not know which are the densest zone, starting from an initial set of point we obtain the solution.
The second algorithm, instead, using convolution, is able to provide a density map of the users and then we can decide where to place our UAVs.
The problem is that the performance of the algorithm depends by the size of the filters and the stride that we choose because the number of computations increases with the accuracy.
Instead, the performance does not depend by number of users.
In this work we did not found the optimal parameters, only after several trials we choose the parameters that provide the best performance.
Maybe using convolutional neural network [17], the performance would be better because they allow to find the optimal parameters.
Another problem of this algorithm is that it requires the size of the radius R because it is necessary to check if there is overlapping. Using fmincon, instead, at each iteration we can maximize the radius choosing it equals to the minimum distance between two centers divided by two. In figure (5.13) we can see the different results of the two algorithms.



(a) *UAVs deployment using fmincon*



(b) *UAVs deployment using convolution.*

Figure 5.13

In the second algorithm there is also the problem that: if there is not enough free space for all drones the algorithm stops its execution. Consider

that we want to place 10 drones and for example after we placed 7 drones there is not enough space. In this case the algorithm stops. It would be better if we can adapt the radius of the circle during the execution but the problem is that we placed a drone only considering the densest zone, without taking into account the others zone or others UAVs, there is not dependency. We should add an optimization: once we found the densest zones, we place UAVs in order to maximize the numbers of covered user and the radius. This is what fmincon does and we are not able to do by hand.
Comparing the execution time of the two algorithms we can notice that the algorithm that uses fmincon is faster and it takes around 9s to find a solution while the other one takes 9 minutes.
Examining in more detail the performance of the 1^{st} algorithm we have that kmeans++ requires a time $O(kk^N)$ because at each iterations we have to compute the distance of each point from the set of centers (in the worst case they are k) and the number of iteration is equal to the number of UAVs we need.
Then, k-means algorithm in the worst case takes a time $O(N^{kd})$ where N is the set of points that we consider, k is the number of cluster and d is the dimension in which we work. Using k-means++ as initializer it has been proved that kmeans finds a solution in less than N iterations.
The 2^{nd} algorithm requires much more time because the moltiplication between the filter and matrix for the users requires a time equal to $O(L^2)$ where L is the size of the filter (it is bigger than the radius of the circles) and we have to perform multiplication entry by entry. The problem is that we have to do this for a number of times equal to the dimension of the area divided by the stride.
The last steps are to find the maximum and verify that that there is not overlapping and this require a time equal to $O(W^2)$ where W is the size of the filter fro drones and we have to do it for a maximum of time equal to the size of submatrix obtained from convolution.
The algorithm starts to go slower when we have already placed some drones and overlapping starts to occur.

Chapter 6

Future work

This work is a good starting point for what regards routing protocol in dynamic networks because it shows as a link can disappear or appear depending by the channel's conditions.
It also provides an idea about the performance since the used parameters are based on Wi-Fi, so it is a practical example.
Starting from here, this work can be improved in many directions. For example, we can develop a routing protocol based completely on georouting exploiting the physical position of UAVs without using routing table.
Furthermore, if we want to improve the efficiency, we can take into account the lifetime of an UAV considering the necessary power to transmit data. Then also the protocol has to be adapted and, for example, we can search the shortest path with the minimum energy consumption for a given destination.
Also, it would be interesting to simulate these two protocols using software like NS3 or Omnet++ because they allow to generate more packet and so, we can consider also collision.
At the end, if we want to simulate a more realistic scenario it would be interesting to assume that channel is mono directional, i.e. there is a link from drone i to drone j but not the opposite for example. In this case, the only difference would be in the route reply because we cannot use the reverse path to send the RREP packet, but we have to start a route discovery again. For reasons of simplicity this case has not be considered and it has been seen as a further step.
Indeed, if we look the algorithm for the optimal deployment, we can decide to deploy a drone in order to guarantee a minimum rate to all users. It is not so complicated because the idea is the same of the one about if a link between two UAVs exists and we have to add only the constraints in the function fmincon.

The algorithm that uses filter and matrix can be improved using Neural Network because the idea is almost the same. We can use the output of the function fmincon as training sample and then the convolutional neural network should be trained and being able to provide as output a good set of centers.
Also, from the time point of view the performance should be better.
Furthermore, a good idea would be to adapt this method to work continuously. It means that it always finds the densest zones and then we decide where to place our drones. The goal is not to obtain a set of centers at the end of each iteration but only know the densest zone and available positions.

Chapter 7

Conclusion

Starting from my academic studies, this thesis aims to implement two routing protocols and analyse them with respect to a defined metric.
One protocol aims to minimize the number of hops to reach the destination while the other one maximizes the transmission rate.
As expected, if we compare these two protocols with respect to end-to-end delay we see that the protocol that minimizes the number of hops is better. This is mainly due to the route discovery which is faster in this case because each node broadcasts a RREP and discards it if it had already received another one with the same id. In the other protocol, since we have to find the maximum-minimum capacity we have to analyse each link, and this require more time.
Then, in the second phase, we implemented two algorithms for the optimal UAVs deployment. From the beginning this problem has been seen as a clustering problem where the final position of an UAV is the center of the cluster. The problem of this approach is that it founds the centers without including in the process the distance between centers and, at the end, we obtained two centers that were too close leading to a small coverage range and consequently a small number of covered users.
For these reasons we decided to process in another way and we exploited the function fmincon. The obtained results are good since the percentage of covered users is around 80%.
Also, the performance of the second approach are quite good but it is slower because the computations are on a matrix of high dimensions.
Because the basic idea is the same as the convolutional neural network trying with this one the performance should improve and provide good results.

Bibliography

[1] H. P. Syed Ahsan Raza Naqvi, Syed Ali Hassan and Q. Ni, *Drone-Aided Communication as a Key Enabler for 5G and Resilient Public Safety Networks.* IEEE Communications Magazine, January 2018.

[2] H. T. Silvia Sekander and E. Hossain, *Multi-Tier Drone Architecture for 5G/B5G Cellular Networks: Challenges, Trends, and Prospects.* IEEE Communications Magazine, March 2018.

[3] F. Lav Gupta, Raj Jain and G. Vaszkun, *Survey of Important Issues in UAV Communication Networks.* IEEE Communications Surveys and Tutorials, Volume PP, Issue 99, November 2015.

[4] A. M. J. B. David B., Johnson David, *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks.*

[5] S. R. D. Charles E. Perkins, Elizabeth M. Royer and M. K. Marina, *Performance Comparison of Two On-Demand Routing Protocols for Ad Hoc Networks.* IEEE Personal Communications, February 2001.

[6] R. W. Nathalie Mitton, Enrico Natalizio, *Beacon-less mobility assisted energy efficient georouting in energy harvesting actuator and sensor networks.* ResearchGate, 1 October 2015.

[7] K.-C. C. Shao-Shan Chiang, Chih-Hung Huang, *A Minimum Hop Routing Protocol for Wireless Sensor Networks A Minimum Hop Routing Protocol for Wireless Sensor Networks.* ResearchGate, 30 June 2015.

[8] A. Goldsmith, *Wireless Communications.* Cambridge University Press, 2005.

[9] Y.-B. Ko and N. H. Vaidya, *Location-Aided Routing (LAR) in mobile ad hoc networks.* J.C. Baltzer AG, Science Publishers, 2000.

[10] J.-S. C. S.-M. W. Neng-Chung Wang, Yung-Fa Huang and C.-L. Chen, *An Improved Location-Aided Routing Protocol for Mobile Ad Hoc Networks with Greedy Approach.* WSEAS TRANSACTIONS on COMMUNICATIONS, Issue 8, Volume 8, August 2009.

[11] H. Akeb and Y. Li, *A Basic Heuristic for Packing Equal Circles into a Circular Container.*

[12] M. B. Mohammad Mozaffari, Walid Saad1 and M. Debbah, *Optimal Transport Theory for Power-Efficient Deployment of Unmanned Aerial Vehicles.* arXiv, February 2016.

[13] *Matlab function fmincon.* `https://www.mathworks.com/help/optim/ug/fmincon.html`.

[14] *Optimization probelm with Matlab.* `https://www.mathworks.com/help/optim/ug/constrained-nonlinear-optimization-algorithms.html`.

[15] *Kernel Density Estimation (KDE).* `https://en.wikipedia.org/wiki/Multivariate_kernel_density_estimation`.

[16] *A Comprehensive Introduction to Different Types of Convolutions in Deep Learning.* `https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learnin`

[17] R. G. A. F. Joseph Redmon, Santosh Divvala, *You Only Look Once: Unified, Real-Time Object Detection.* arXiv, 9 May 2016.