Edge Server Placement on Traffic Lights for Smart City Services

THEODOR-FABIAN NICULAE, University of Twente, The Netherlands
t.f.niculae@student.utwente.nl

Cities of the future will be smarter, with services being driven by citizens and adapted dynamically with a goal of ensuring higher efficiency and safety. Towards this vision, Internet-of-Things (IoT) collect data from the city to provide more information about the context and operate more smartly. However, processing the data generated by these sensors may not take place on the sensors due to their resource limitations. Hence, there is a need for edge computing infrastructure. As a densely deployed city infrastructure, street poles can be a promising candidate. In this study, we consider such a system and design an algorithm to decide where the edge servers should be placed to minimise costs without affecting latency and processing time of the edge servers. Using the data on traffic light locations for Enschede, we analyse the performance of our algorithm in comparison to a simple baseline, which places the edge servers on randomly-selected street poles. Our numerical analysis shows that total costs and number of edge servers deployed can decrease significantly compared to the baseline.

Additional Key Words and Phrases: Edge computing, internet of things, edge servers, traffic lights.

1 INTRODUCTION

From homes to transportation, we witness that services become smarter, collect data from the sensors deployed in an area, and interpret data to improve the efficiency and smartness of services. For example, a home resident can turn on the lights remotely through an application, control the doorbell camera, or smart locks. Alternatively, devices can be controlled automatically without a human in the loop by processing the collected sensor data and acting accordingly [19]. Moreover, the majority of the devices and services companies provide are connected to the Internet to process the information collected by the device. Some examples include tags that can track the location in which the item is located and warn the owner in case of a suspicion of being lost.

To process the collected data from all interconnected smart objects, a computation infrastructure is needed. Typically, cloud servers that process the collected data are thousands of miles away, resulting in long round trip times compared to a case with processing near the sensors. Smart Lamps and traffic lights constitute the densest electrically operated public infrastructure in urban areas [14] and could be a promising option for deploying edge servers to minimize the costs due to their large number and already-made placements. Since these city infrastructures are close to the end users where the sensors are deployed, the resulting latency is expected to be lower compared to the case where the processing is performed on a remote cloud. This means that users can get almost instant responses having low latency for actions such as decisions that autonomous cars have to make, traffic monitoring including possible incidents alerts, simple connections of mobile phones in the covered area and sensors that may be on the poles. For such smart city services, edge computing can offer benefits over cloud computing, which is meant to process data that is not time-driven. Besides, edge computing is favoured for remote locations without stable connectivity to a centralized location [9].

An earlier study proposed street lamps as a platform for computing [14]. Inspired by that study, this paper aims to understand whether existing traffic lamp infrastructure suffices to offer the needed edge computing capacity and where the edge servers should be deployed on these traffic lamps to meet the latency requirements of the applications. As a case study, we focus on Enschede city and use the original data for the traffic lights. We developed an algorithm to determine where edge servers should be placed on these traffic lights by considering various factors, such as the number of users and sensors, transmission rates, and computing capacity of the edge servers. Different from the prior work, we consider the multitude of data sent at a point in time and latency of the applications. These factors affect the placement of the edge servers and are important when considering low latency applications that need such fast responses.

Given the problem statement, we will address the following research questions in this paper:

- Where could the edge servers be placed for good spatial coverage?
- What would be the average latency for the users and sensors assigned to a pole?
- Where should the edge servers be placed on traffic lamp infrastructure within a city such that the coverage, processing power of the edge servers, and average latency are acceptable for low-latency applications?

To address the above-listed questions, we will first model an edge computing environment and devise a simple algorithm to determine where the edge servers should be deployed by defining a simple scoring scheme for each traffic light. The score is based on the TOPSIS [13] method for multiple criteria decision making. The core idea behind this method is assessing different traffic lights properties that have different scalar values by computing shortest distance to the positive ideal solution and the longest distance from the negative ideal solution. This is an objective approach of creating a score based on the weights and signs, that can be modified by the users to accomplish their ideal approach for the algorithm to make the placements. Further details about the approach can be found in Section 4. We will then analyse the performance of our algorithm, dubbed Edge Score, under various scenarios. We compare the performance of Edge Score to a baseline which randomly deploys servers to a given fraction of traffic lights without considering load, capacity, or latency requirements of the applications.

The rest of the paper is organized as follows. Section 2 provides an overview of the most relevant work while Section 3 presents the

TScIT 37, July 8, 2022, Enschede, The Netherlands

considered setting. Next, Section 4 introduces Edge Score which is the proposed edge server placement algorithm. Section 5 provides a performance assessment of Edge Score in comparison to a simple baseline. Finally, Section 6 and Section 7 discuss the future research directions and conclude the paper.

2 RELATED WORK

Using smart lamps as an edge computing platform is proposed by [6]. The goal of their study was to analyse where edge servers (cloudlets) can be deployed on access points such as cellular base stations, routers or street lamps. After examining coverage metrics such as spatial coverage that refers to the ratio between the union of the communication ranges of available cloudlets and the total size of the area, that were recorded with the help of two smartphone applications, authors designed two algorithms namely Random and Greedy-Cost-Aware for cloudlets placement. Different from [6], we take into consideration the assigned users/sensor in the radius of each potential placement of edge servers and round-trip times of the data sent by the people and sensors, including edge server processing times. This is necessary to ensure some performance guarantees for the served applications.

Other studies on edge computing using street lamps are typically qualitative, where features of concepts for smart street lamps are listed or hypothetical opinions on the next features of the edge computing integration into cities are discussed [4, 5, 14, 16, 17, 20, 22]. Studies on edge server placement are many, ranging from algorithms considering energy efficiency of the network [11] to those considering security of the communication [10]. Gedeon et. al [7] propose GSCORE for placing cloudlets (edge servers) in urban spaces that considers costs and quality of service (i.e., communication ranges and available resources). The developed algorithm splits an area into smaller squares and for each of them one or more edge servers are placed. In this way, overlay might appear if the communication ranges of the edge servers are greater than the length of an imaginary square. Also, placing an edge server in each of the squares might not be necessary because of the region/number of data traffic that is generated there. Our algorithm works differently because the placements of the edge servers are done such that a minimum number of servers is required, and a maximum number of users and sensors are satisfied as much as possible. Also using sensors' locations, we can better estimate if the average latency is good enough for smart city services such as traffic monitoring, energy monitoring, noise monitoring, urban Lab monitoring [3]. However, the mentioned study does not consider the distance between a user/sensor to a possible edge server placement such that the round-trip times are minimized.

3 SYSTEM MODEL

Let us consider an IoT network of N sensors deployed in a city to collect environment data (e.g., temperature, video, air pollution) as illustrated in Fig.1. We consider M users that can also use the edge servers for different tasks, such as video streaming, web browsing or different actions that might be processed by edge servers. We assume that the information received by an edge server from a user/sensor is processed and sent back, the processed data being between $1Mb$ and the number in Mb of the data received from the users and sensors. Each user device, sensor, and edge servers that will be placed on a traffic light are associated with a transmission speed depending on their connection bandwidth.

We assume that each server has a certain computation capacity, denoted by C seconds per processed Megabit, which depends on the server hardware, e.g., number of processing cores, the CPU clock speed (in GHz). Moreover, placing an edge server at a traffic light might lead to different costs depending on the location or other factors that can influence its placement, such as modifications of the traffic lights for installing and housing the edge server. We randomly assign variable costs to traffic lights that may be equipped with edge servers. This will be shown in the Simulation Parameters Table 2. As our edge server setting, we consider the specifications of Raspberry Pi4 computers that have low price and very limited computational power. Using other, more performance-oriented models of edge servers, transmission speeds, data handled, and round-trip times can be improved significantly.

While the processes might experience queuing delay at a server for being processed, we will ignore the latency due to queuing when the edge server needs to send all the data back to the senders (transmission queuing). This is done because in most cases, the time it takes for processing the data sent by all the users and sensors assigned to a traffic light is short, and transmission speed cannot cope with such large chunks of data sent in a single second. Note that we assume that each edge server also possesses a wireless router to receive the transmitted data from the sensors for processing and to send the data back. Moreover, traffic lights can have a connection to the cloud in cases where the computation capacity of this node falls short of executing a task and thereby requiring a more compute-reach server. We assume the wireless technology used by the sensors and users to be 4G and the edge servers to send the data through their wireless incorporated router. We will denote the wireless coverage range of an edge server by R meters.

Note that a user or a sensor might be in the wireless coverage range of multiple edge servers, depending on the density of deployments. For a dense deployment or for wireless networks with higher coverage range (operating at sub-6 GHz bands), sensors and users can be covered by multiple edge servers. In our approach we assign users to edge servers based on their haversine distance, which is the great-circle distance between two points on a sphere given their longitudes and latitudes. We chose the shortest distance considering all distances between the user and all available edge servers in its range. This can be visualised in the Fig. 1 where there are two traffic lights that can be equipped with edge servers. The user will be handled by the edge server that is in its range and the nearest to the user. This approach can also leave users and sensors unselected if traffic lights do not have the coverage range big enough or simply if there are no traffic lights near them to be equipped with edge servers. For these users and sensors that do not have access to an edge server directly, we claim that their data is sent to the cloud through other channels.

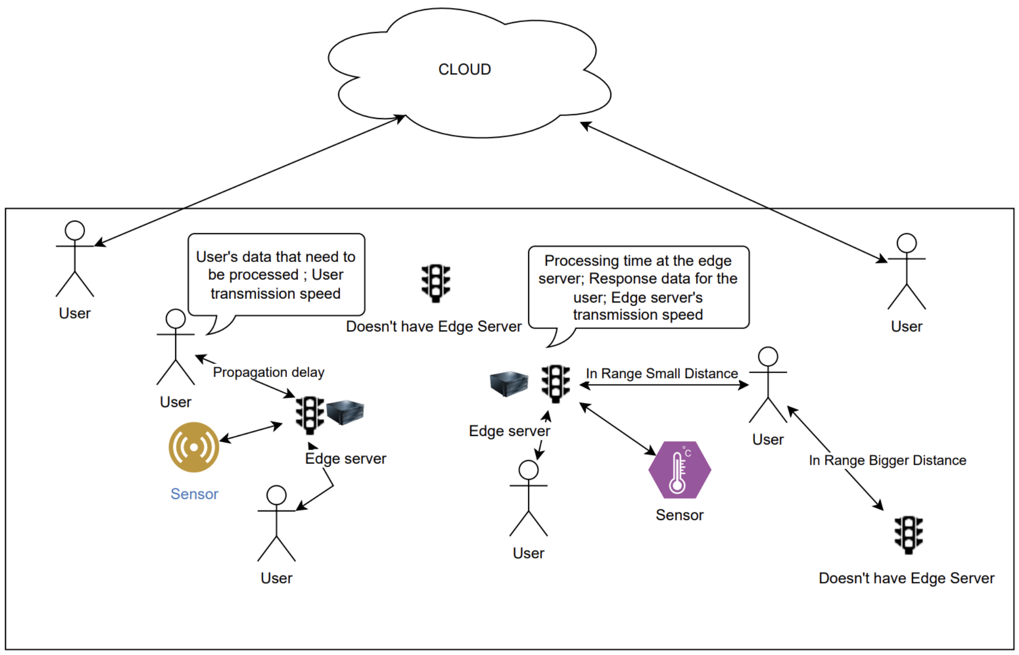

Fig. 1. The System Model. Sensors generate data to be processed, and the users consume the processed data for their services, e.g., checking the traffic conditions or air pollution in different parts of the city.

4 EDGE SERVER PLACEMENT: A HEURISTIC

In this section, we introduce the design of our algorithm, so-called Edge Score. The goal of our algorithm is to place edge servers within a selected area for good area coverage, number of users and sensors served by the edge sensors to be high, but also for as small as possible average latency. Edge Score considers variables such as number of users and sensors, round-trip times of devices if they would be assigned to an edge server and specifications of computing capacities of edge servers models.

To decide on whether a traffic light should have an edge server, we define a score for each traffic light. To determine the score, Edge Score uses the following properties as its input:

- number of devices/sensors in range of a traffic light,
- total cost,
- transmission speed (in Mb/s),
- total data generated by all the users/sensors that are assigned to this traffic light,
- Mean of waiting times of users and sensors assigned to this traffic light,
- wireless coverage radius (in m).

Edge Score uses TOPSIS Multi Criteria Decision Analysis [23] to create a score from columns of data that have different scalar values. TOPSIS uses the principle that the alternatives selected must have the shortest distance from the positive ideal solution and the farthest from the negative ideal solution from a geometrical point by using the Euclidean distance to determine the relative proximity of an alternative to the optimal solution. Each of the properties presented has a weight and a sign. The weight of all the properties needs to sum up to 1. In this way we can define for each property the importance, for the algorithm behind TOPSIS to know the significance of these variables. The sign for each property is a Boolean variable, so that the algorithm knows if higher values (TRUE) or lower values (FALSE) are preferred. An existing TOPSIS algorithm was used [21] for creating the scores in Edge Score . The main steps of Edge Score is as follows: the creation of an evaluation matrix consisting of m alternatives and n criteria, normalising the matrix, calculating the normalised decision matrix, determining the worst and best alternative, calculating the Euclidean between the target alternative and worst condition and calculating the similarity of the normalised matrix. The algorithm returns a list which contains the index of traffic lights sorted from the best to the worst. The first one from the list is chosen for edge server deployment.

Edge Score calculates (for each traffic light) the distances to all these users/sensors that are in its communication range. If the users are in the range of multiple possible edge servers, they will be assigned to the one that is closest to them. The next step is to assign each traffic sign a variable cost and fixed cost to determine the total cost of the traffic light if an edge server is deployed there. The fixed cost is the price that we need to pay for the edge server itself and the variable cost is randomly assigned from a list of prices. This is chosen because modifying a traffic light for hosting an edge server may imply smaller or higher prices to be paid. Then, focusing on the edge server model, Edge Score includes the CPU clock speed in GHz, number of the processors and transmission rate (in Mbps) of the edge servers, sensors and users. Depending on the clock speed and number of processors we can calculate how much data can be processed by a processor at a single clock cycle, considering 100% load. Edge Score needs to first calculate the round trip time for each

Algorithm 1 Edge Score algorithm.

```
Require: Lat/Long coordinates, filled variables
  $avlbES \leftarrow 1$                                    ▷ Available Edge Server
  $EScoordinates \leftarrow []$                 ▷ The coordinates of the Edge Servers
  $latencyT \leftarrow []$          ▷ List for keeping latency times of edge servers
  while $avlbES <= N$ do
    totalData() ▷ Computes the total data arrived at each of the traffic lights and the round trip times of users's devices or sensors at each individual traffic light is they are in its range
    meanwaitingtimes()        ▷ Computes the average waiting time of the devices/sensor assigned to each of the traffic lights
    $allValues \leftarrow []$
    $listOfIndexes \leftarrow []$
    for i in range dataframe indexes do  ▷ prepare the evaluation matrix for TOPSIS to make the scores
      $row \leftarrow []$
      if type of element of the row is 'traffic signal' and traffic light at index i has number of devices >0 then
        $row \leftarrow variablesforTOPSISscore$
        $allValues \leftarrow row$
        $listOfIndexes \leftarrow i$
    if allValues is empty then. ▷ Here we can finish the program if there are no more traffic lights that are necessary to be equipped with edge servers
      Print for the traffic lights selected to be equipped with edge servers information such these means: latency times for them, number of devices in range hadled, total cost and total data received
      break ▷ check if the list of selected street lights that can have a score to be computed if different from 0
    $evaluationMatrix \leftarrow allValues$
    $scores \leftarrow topsis(evaluationMatrix, weights, signs)$ ▷ Compute the score based on the evaluation matrix, weights and signs
    $EScoordinates \leftarrow scores[0]$
    $latencyT \leftarrow avgLatency[scores[0]]$  ▷ The average latency of the highest score traffic light
    $typeTrafficLight \leftarrow edgeServer$  ▷ Change the type of the highest score traffic light to edge server
    $i \leftarrow 0$
    while $i < nrI$ do                                    ▷ number of indexes
      if type of element of the row is 'traffic signal' then
        $b \leftarrow 0$
        $copy \leftarrow copyI$ ▷ copy of number of coordinates of devices in range of i
        while $b < nrCi$ do                        ▷ number of coordinates of i
          $ts_x \leftarrow latI$                  ▷ the latitude of the traffic signal i
          $ts_y \leftarrow lonI$                 ▷ the longitude of the traffic signal i
          $j \leftarrow 0$
          $es \leftarrow nrESc$ ▷ The last selected traffic light that was transformed to an edge server number of coordinates
          while $j < lengthofes$ do
            $es_x \leftarrow latES$                  ▷ the latitude of the edge server
            $es_y \leftarrow lonES$              ▷ the longitude of the traffic signal i
            if $ts_x$ is $es_x$ and $ts_y$ is $es_y$ then
              copy remove the sublist  ▷ Removes from the copy the sublist from the list of lists of the current traffic light
            $j \leftarrow j + 1$
          $b \leftarrow b + 1$
        $trafficlight \leftarrow copyC$         ▷ Gets copy of remaining coordinates
        $nrDevInRange \leftarrow lCopy$ ▷ number of devices in range of the current traffic light received the length of copy
      $i \leftarrow i + 1$
    $avlbES \leftarrow avlbES + 1$
```

sensor/user assigned to each of traffic lights. Latency has four components: i) uploading the data from the sensor to the edge server t_u, ii) processing delay at the edge server t_p, and iii) sending the data back to the users/sensors t_s and iv) the propagation delay t_{pd}.

The first component can be calculated as: $t_u = us_d/u_s$, us_d representing data sent by the user or sensor and u_s the transmission speed of the users/sensors. We assume that users have a different transmission speeds than the sensors. The time that it takes for the edge server to process the received information is based on the processing capacity of the CPU and the amount of bits to be processed. Then, we can calculate t_p as follows: $t_p = us_d/(pc_{cpu} * nr_p * 64)$. ($pc_{cpu}$ represents the processing capacity of the CPU, nr_p the number of processors, and 64 represents the number of bits. The reason behind multiplying with 64 bits is that 1 load instruction represents 8 bytes/64 bits on 64-bit architecture per cycle.

The third component is similar to the first component, in this case representing the transmission speed of the edge server: Since the processed data might be different in size (e.g., usually smaller), this should be considered in calculating the delay for downloading the processed data. Hence, the last delay component t_d can be calculated as: es_d/es_s where es_d represents the data sent back to users and sensors and es_s the transmission speed of an edge server.

The last component is the propagation time. This delay is summed up to the transmission delay for the users, sensors and edge servers. This is calculated as: $t_{pd} = d/l_s$. d represents the distance between the sensor/user and edge server and l_s represents the speed of light. The round-trip time is computed for each individual sensor and for each traffic light that is in its range. An average delay is computed for each traffic signal based on the assigned users/sensors. This is done by summing up all the round-trip times and dividing by the number of assigned users/sensors.

After Edge Score computes the scores using TOPSIS these scores are sorted in decreasing order. Depending on the number of servers to be deployed, the first E traffic light with the highest score will be chosen for deployment, each time, TOPSIS being used for creating a score. After Edge Score selects a traffic light to be equipped with an edge server, it takes out all the coordinates of the sensors/users from other traffic signals that had them also in its range, such that there is no overlap in assigned users and sensors to an edge server and also updates all the properties accordingly.

The output of a program is a list with the indexes of traffic lights that were chosen to be equipped with an edge server, the average latency times of the users of the selected locations for placing the servers, the number of users and sensors handled, the total cost of the placements and also the total data of the users handled. In the end, an HTML will open in the users' default browser in order to show the map with the different types of locations. They have the ability to zoom in and out and see the locations in terms of latitude and longitude of any marked point on the map. Every location is placed keeping in mind the number of users handled and for achieving great spatial coverage.

The computational complexity of Edge Score is $O(n^2)$ where n is the total number of traffic lights received as an input.

This section of the paper covered the first research question, including the idea behind the placements of edge servers.

5 PERFORMANCE ANALYSIS

In this section, we will first introduce the datasets used in the simulations and the goal of simulations. We will compare our Edge Score algorithm with a random approach (referred to as Random) of placing edge servers at traffic lights. We will address the following research questions via simulations:

- What is the difference between using different transmission speeds of edge servers in terms of average latency?
- How does the percentage of selected traffic lights equipped with edge servers affect the total number of users handled both for Edge Score and Random?
- How does the percentage of selected traffic lights affect the total cost for both Edge Score vs Random?

5.1 Traffic Light Dataset

We have used the following resource, Overpass Turbo [18] to extract the locations of traffic lights located in Enschede. Using Overpass turbo, we export the data to a CSV file. Our data includes 435 traffic lights which are distributed in the city as shown in Fig.2. As the heatmap shows, an intersection might have multiple traffic lights.

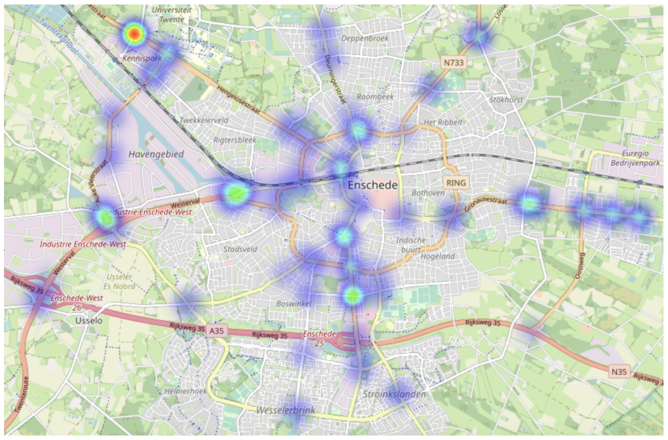

Fig. 2. Heatmap of the traffic lights in Enschede.

Fig.3 shows the distribution of the maximum distance between the nearest neighbour of each individual traffic light around the inner-city area of Enschede. We compute the distances between two neighboring traffic lights using Ball Tree algorithm for Nearest Neighbour searches [12]. As we can observe from the figure, the distance between two traffic lights can be very small (in the order of a few metres) for intersections or city center. However, we also observe that some traffic lights can be as separated as with a distance of 700 metres. From this CDF plot we can also observe that the distance between two nearest neighbour traffic lights is less than 100 metres for almost 70% of the traffic lights.

For the rest of the research, we will focus on the inner-city, which has a dense deployment of traffic lights and where the users and sensors are expected to be more ubiquitous. The considered area is 7.62 km^2 rather than the complete Enschede city with all its suburbs having a total area of 46.34 km^2. In the considered area, there are 146 traffic lights. Table 1 summarises the number of traffic lights in the inner-city and the complete Enschede city with all its suburbs.

5.2 Simulations

We assume that users and sensors are randomly distributed in the area of interest. Please refer to Table 2 for the default values of parameters used in the simulations.

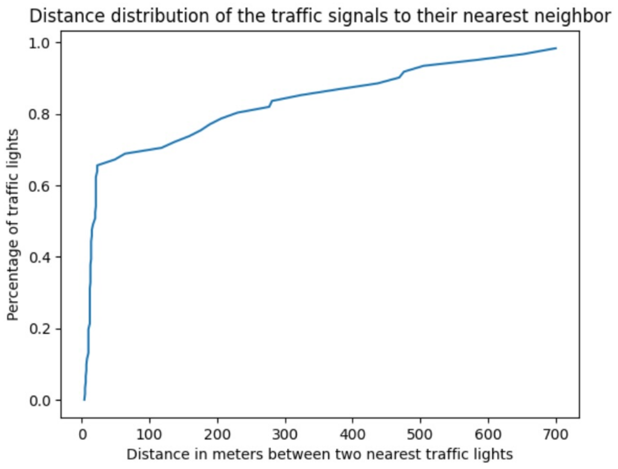

Fig. 3. CDF Nearest Neighbour distance in meters.

Table 1. Number of traffic lights collected

	Traffic lights
Total points of complete area	322
Density per km^2	6.94 $poles/km^2$
Inner city	146
Density per km^2	14.55 $poles/km^2$

Table 2. Simulation parameters.

# of users inner area	$\{1000\}$
Coverage radius (meters)	$[100, 200, \cdots, 600]$
Percentage of traffic lights	$[10, 20, \cdots, 100]$
User link rate (Mb/s)	$[40, 41 \cdots, 65]$
Sensor link rate (Mb/s)	$[50, 51 \cdots, 108]$
Edge server link rate (Mb/s)	$100, 8142$
CPU clock speed (GHz)	1.5
Number of cores	4
Data by sensors (Mb)	$[1, 2, \cdots, 10]$
Data by users (Mb)	$[1, 41, \cdots, 25]$
Data by edge servers (Mb)	$[1, 2, \cdots, sensor/usersentdata]$
Fixed cost (Euro)	60
Variable cost (Euro)	$[100, 200, 300, 350]$
TOPSIS Weights	$[0.15, 0.11, 0.15, 0.15, 0.22, 0.22]$
TOPSIS Signs	$[True, False, True, True, False, True]$

Let us first investigate how an increase in the fraction of traffic lights with an edge server would affect the ratio of inner area coverage.

Fig.4 shows the *coverage ratio* of the selected area which quantifies how much of the total area is covered by the potential edge servers depending on their wireless coverage range R. We compute the area using the ArcGIS [1] software and overlapping areas are considered only once. Analysing the coverage ratio is an important step in order to know what percentage of the selected traffic

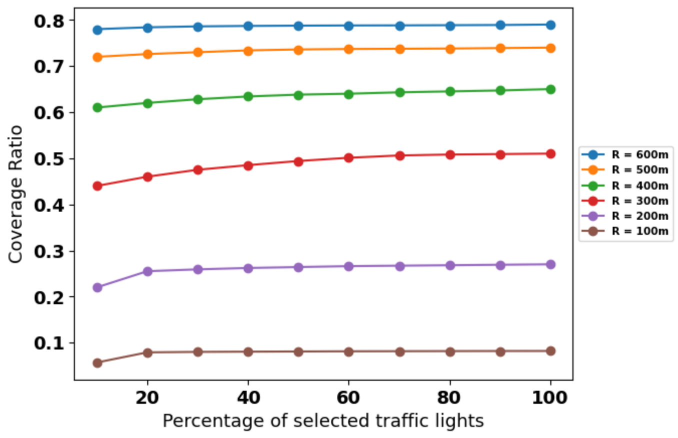

Fig. 4. Coverage ratio with increasing percentage of selected traffic lights, under various radii settings.

lights covers as much area as possible, depending on the communication range. To fully utilise the computation capacities of each edge server, Edge Score avoids placing edge servers that may be in vicinity. From the figure, we can observe that even when all traffic lights have an edge server deployed, it is not possible to cover the whole area. Typically, by selecting between 20%-30% of the available traffic lights out of the 146 poles, the improvement in coverage is marginal. Since traffic lights might be close to each other (e.g., at an intersection), deploying new servers does not offer significant improvement in terms of coverage ratio. Comparing different coverage radii, unsurprisingly, higher radius results in higher coverage ratio. Two key take-aways from this analysis are as follows: first, existing traffic lights may not suffice to meet all computation requirements of a city as they do not cover the whole area. Second, increasing the coverage radii may result in fewer edge server placements, to a smaller total cost.

Furthermore, the map in Fig. 5 shows the output of Edge Score when 10% of edge servers are selected to be placed within the inner-city of Enschede. The communication range of the edge servers was 500 metres and the number of chosen users being 998 and only 2 sensors.

On the map we can see the exact location of the edge servers in order to know on which traffic light we can mount them. The users receive in the program output the average latency times of each edge server, the total cost for placing all of them and the total number of users/sensors within the range of servers. For this example, the total cost was 3640 euro, 838 users and sensors handled and an average time of 5 seconds. This very big average latency time was due to using transmission times of edge servers of only 100 Mb/s. After each edge server processes the data received from their assigned users and sensors, the transmission rate cannot handle all the data at once, so queuing delays before transmission will appear. The selected intervals of data that may be sent at a single point in time are really large, made for showing differences in terms of performance.

In continuation, we will assess how Edge Score performs compared to Random. Together with this random approach of placing

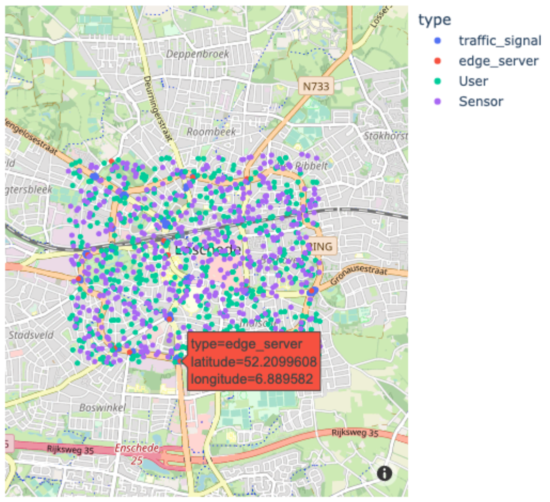

Fig. 5. The output of the Python application showing the edge servers

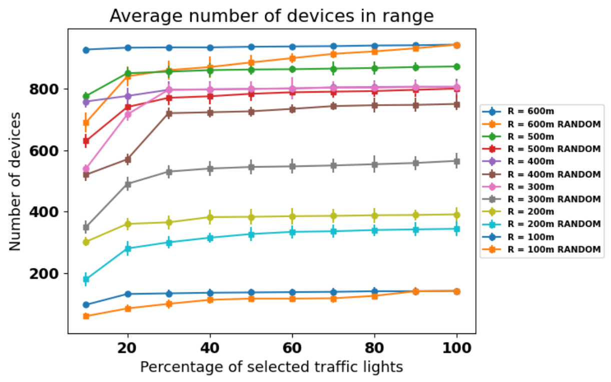

Fig. 6. Average number of devices in range using different coverage transmission ranges and percentages of selected traffic lights to be equipped with an edge server Edge Score vs. RANDOM placement

a proportion of available traffic lights, we assess the difference between the total cost of positioning edge servers and number of devices that are in range of edge sensors. The experiment is done for every proportion and radius from Table 2. We report the average of multiple experiments along with 95% confidence intervals. We use the parameter values in Table 2.

In Fig.6, we can see the main differences in terms of the number of devices handled by the computing infrastructure, determined by Edge Score and Random. Because the inner area of the city is quite small, and the traffic lights are grouped together in intersections, around 5% of the users can not have their data processed when using a maximum radius of 600 metres. The difference between Random and Edge Score in terms of numbers of devices/sensors is significant, with a difference of 40% between them. Also, when selecting a small percentage of edge servers to be placed, 20% - 30% out of the total of 146, Edge Score achieves markedly high number

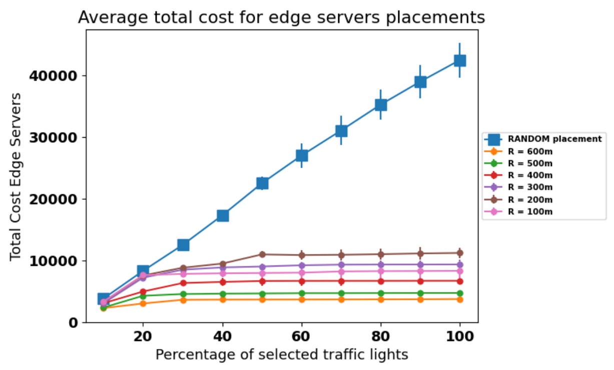

Fig. 7. Average total cost for edge servers placements Edge Score vs. RANDOM placement

of users in range compared to Random and in general because of the Edge Score placing the edge servers efficiently.

Fig. 7 shows the total cost of edge servers based on the percentage of edge servers that are selected to be placed. Random places all the edge servers, increasing the total cost linearly.

Our approach makes the placements as they are described in the past sections based on the number of devices in range, data that needs to be processed and average waiting times. Based on the inner area of Enschede, larger communication range of the edge servers decreases the total cost. For example, out of the 10% of edge servers to be placed, only a few will be really selected by Edge Score . Consequently, this results in ending the process faster and leads to more efficiency than placing all the allowed percentage of servers. Having higher coverage ranges accomplish this goal faster and keeps the total costs low. The total cost for placing percentages of edge servers using random placements increases linearly and can reach high levels depending on the number of traffic lights equipped. Edge Score and related code/simulation plots can be found here [15].

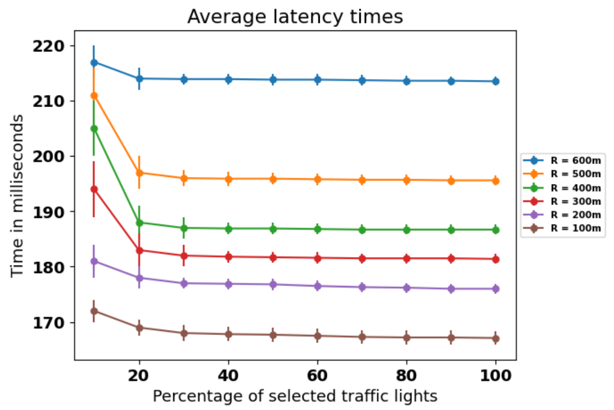

Fig. 8. Average latency based on selected percentage of traffic lights

In this simulation, we will answer the second research question of this paper, namely the average latency times of users and sensors. Real data of the sensors transmitted to be processed in a single second can be very small. This can be seen in [3] that analysed all the current data generated by all the sensors of the city Barcelona. For our simulation we will take into account the same Simulation Parameters Table 2 but with users streaming simultaneously 4K videos, requiring 25Mb/s [8] of data. We took this value of data sent by the users in order to simulate a real life scenario. Now, considering traffic lights are equipped with ThinkEdge SE350/SE450 [2] edge servers with transmission speeds of 1GB/s (these are able to send data from 1GB/s to 1000GB/s), we have the following plot Fig. 8. The average waiting time can vary between 2% and 9% depending on the chosen radius and percentage of traffic lights selected. When choosing only 10% of traffic lights to be equipped with an edge server, the average latency times are slightly higher because Edge Score tries to make the placements such that the maximum number of users are handled. When the coverage transmission range is bigger, more users are handled, so the average latency times are increasing too. The biggest change between 10% and 20% of selected traffic lights is when the radius of edge servers is between 300 and 500 meters. This is due to big amounts of users handled by the first placements made by Edge Score, leaving smaller number of users assigned to other edge servers. The average times remain the same after 30% of selected traffic lights because all the necessary placements of edge servers are full filed by Edge Score. We also made the simulation for randomly assigned users' data between 1 and 25 Mb for different actions such as Web surfing, email, social networking, 4K video streaming, Online multiplayer gaming and the average latency times, giving the same other values as the first simulation for determining the latency, were between $8 - 12 milliseconds$. Choosing to place multiple edge servers, reduces slightly the average latency times, but increases the total cost. When using higher radii coverage ranges more users can be handled, resulting in less total costs for the placements, but bigger average latency times.

Using the Simulation Parameters Table 2 and based on the data sent by the users from the average latency times simulation we would choose a 600 meter coverage radius of edge servers, having a small total cost for the few placements that are made, if the average latency times do no affect the actions made by users and sensors. If applications need low latency times we would make a slightly higher number of placements with edge servers having less radii as coverage range and higher processing capacities including transmission speeds. Using this approach we can answer the third research question, depending on the users' needs.

5.3 Discussion

Using the Simulation parameters Table 2 and the selected inner-city of area Enschede, when we have a only a few edge servers equipped with high coverage radii and increased transmission times, the overall number of handled users increases. On the other hand, average latency times increases too because of the multitude of data received from the assigned users.

Moreover, the total cost of placement is lower compared to equipping a bigger number or percentages of traffic lights with lower performance edge servers. The selected area is a small example of what can be achieved using Edge Score and the values of chosen simulation parameters depend on the considered setting. Edge Score showed higher performance than a simple baseline which can be also further improved. The idea of creating such an algorithm was within the scope of potential stakeholders to simulate different environments with their own values based on their choice. This creates a powerful tool that can be used together with other simulation applications to gather insights of actual real world edge server placements.

6 FUTURE WORK

In this section, we will address possible features that can be implemented based on our current approach for edge server placement.

First, we can further improve Edge Score by letting the users choose how much overlay between two edge servers placements can exist. Because of the overlay in the area, we can split the number of people evenly to multiple edge servers, instead of a single one to handle all the users from that particular area. In this way, we can lower the average latency times, because of the less number of people assigned to an edge server.

Secondly, efficiency of the algorithm can be improved. Searching all the information in the Python data set and updating it, increases quadratically and would not be efficient for large number of edge servers and large population samples. Lastly, different edge server properties can be considered for a more realistic modelling of the computing nodes. This can be done by introducing their specifications and outputting a score for them, based on the final average waiting time, radius, real transmission speeds tested in real life environment and processing specifications.

7 CONCLUSION

In this paper, we have discussed the impact of selecting proportions of edge servers to be placed, different communication ranges of the servers, transmission speeds and average latency times. The inner-city of Enschede was an example for building the algorithm and assessing its performance. Possibilities of using different parameters were discussed in order to better understand the potential edge server placement algorithms and needed values of different variables. This can be replicated for larger areas, together with more users, sensors and other estimations or values chosen by the users of this tool.

8 ACKNOWLEDGEMENTS

I would like to thank my track chair Suzan Bayhan who was also my supervisor, for all support and efforts made to guide me through all the phases of the project, giving me feedback, helping me develop ideas, but also for being strict and monitoring my work so that I attain my potential and provide the best work.

REFERENCES

[1] Arcgis. *Get Started with Distance and Direction in Web AppBuilder for ArcGIS (Developer Edition).*

[2] Lenovo edge servers. *Purpose-built edge servers for secure IoT, edge computing, and storage.*

[3] Amir Sinaee, Jordi Garcia Almiñana, X. M. E. M.-T. J. C. G. G. F. C. Estimating smart city sensors data generation. *The 15th IFIP Annual Mediterranean Ad Hoc Networking Workshop* (2016).

[4] Carvalho, G., C. B. P. V. . B. J. Edge computing: Current trends, research challenges and future directions - computing. 993–1023.

[5] Corcoran, P., . D. S. K. Mobile-edge computing and the internet of things for consumers: Extending cloud computing and services to the edge of the network. *IEEE Consumer Electronics Magazine 5*, 4 (2016), 73–74.

[6] Gedeon, J. A. Urban edge computing. *TU Darmstadt Publication Service* (2020).

[7] Gedeon, J., S. M. K. J. F. P. K.-K. M. C. W. L. . M. M. From cell towers to smart street lamps: Placing cloudlets on existing urban infrastructures. *2018 Third ACM/IEEE Symposium one Edge Computing* (2018), 187–199.

[8] GVEC. How much data is created every day in 2022? *GVEC Internet Products Services, Tech Insight Tips* (2021).

[9] Hiter, S., . E. C. The pros and cons of edge computing.

[10] Kasi, M. K., Abu Ghazalah, S., Akram, R. N., and Sauveron, D. Secure mobile edge server placement using multi-agent reinforcement learning. *Electronics 10*, 17 (2021), 2098.

[11] Li, Y., and Wang, S. An energy-aware edge server placement algorithm in mobile edge computing. In *2018 IEEE International Conference on Edge Computing (EDGE)* (2018), IEEE, pp. 66–73.

[12] Mohamad Dolatshah, Ali Hadian, B. M. Ball*-tree: Efficient spatial indexing for constrained nearest-neighbor search in metric spaces.

[13] Mohamed Hanine, Omar Boutkhoum, A. T. T. A. Application of an integrated multi-criteria decision making ahp-topsis methodology for etl software selection.

[14] Mühlhäuser, M., Meurisch, C., Stein, M., Daubert, J., Von Willich, J., Riemann, J., and Wang, L. Street lamps as a platform. *Communications of the ACM 63*, 6 (2020), 75–83.

[15] Niculae, T.-F. Edge score.

[16] Oyinlola, T. Energy prediction in edge environment for smart cities. *2021 IEEE 7th World Forum on Internet of Things (WF-IoT)* (2021).

[17] Pan, J., . M. J. Future edge cloud and edge computing for internet of things applications. *IEEE Internet of Things Journal 5*, 1 (2018), 439–449.

[18] Raifer, M. Overpass-turbo. *Query wizard.*

[19] Sardjono, W., . S. E. The relationship between internet growth and implementation of the internet of things. *Journal of Physics Conference Series* (2021).

[20] Shi, W., C. J. Z. Q. L. Y. . X. L. Edge computing: Vision and challenges. *IEEE Internet of Things Journal 3*, 5 (2016), 637 – 646.

[21] shivabehl. Topsispy.

[22] Solmaz, G., C. F. . d. J. J. M. Getting started with smart cities.

[23] Zulqarnain, R. M., a. S. M. Application of topsis method for decision making. *International Journal of Scientific Research in Mathematical and Statistical Sciences 7*, 2 (2020), 76–81.