Research Article

# PERFORMANCE COMPARISON OF SPATIAL SEARCH ALGORTIHMS FOR SPECIFIC DATASETS IN SMART CITIES

**Mert Can GİRGİN[†], Ali BOYACI[††]**

[†] Istanbul Commerce University, Department of Urban Systems and Transport Management, Istanbul, Turkey

[††] Istanbul Commerce University, Department of Computer Engineering, Istanbul, Turkey

https://orcid.org/0000-0001-9356-7118, https://orcid.org/0000-0002-2553-1911

**girgin.mert@yandex.com, aboyaci@ticaret.edu.tr**

## ABSTRACT

The concept of smart city has emerged in the digital age. One of the main purposes of smart cities is to provide components that will provide time efficiency. Smart transportation and parking services are included in this concept. The basis of these services is based on real-time spatial search algorithms. We need to use performance spatial search algorithms for real-time spatial searches. Popular spatial search algorithms; k nearest neighbor, rectangle queries, r-tree and kd-tree. In the query made from a point in the spatial plane, the selection of the correct algorithm is important in terms of performance. The purpose of this study; to determine the algorithm that determines the nearest neighbor in a given dataset in the fastest way for the selected center point. The 4 spatial search algorithms written in Python language were compared with the tests and the most suitable algorithm was determined for the data set. The algorithm can be used in the city component model similar to the data set, so efficient time management is provided in the city life where time is valuable.

**Keywords:** Smart Cities, Spatial Search, k Nearest Neighbor, Rectangle Queries, r-Tree and kd-Tree

## AKILLI ŞEHİRLERDE BELİRLİ VERİ SETLERİ İÇİN MEKÂNSAL ARAMA ALGORİTMALARININ PERFORMANS KARŞILAŞTIRILMASI

## ÖZET

Dijitalleşen çağda akıllı şehir kavramı ortaya çıkmıştır. Akıllı şehirlerin temel amaçlarından biride zaman verimi sağlayacak bileşenler sunmaktır. Akıllı ulaşım ve otopark hizmetleri bu konsepte dahildir. Bu hizmetlerin temeli gerçek zamanlı uzamsal arama algoritmalarına dayanmaktadır. Gerçek zamanlı uzamsal aramalar için performanslı uzamsal arama algoritmaları kullanmamız gerekmektedir. Populer uzamsal arama algoritmaları; k en yakın komşu, dörtgen sorgular, r-ağacı ve kd-ağacıdır. Uzamsal düzlemin içerisinde yer alan bir noktadan yapılan sorguda doğru algoritmanın seçimi performans açısından önemlidir. Bu çalışmanın amacı; seçilen merkez noktası için küçük boyutlu sınırları belirli bir veri setindeki en yakın komşuyu en hızlı şekilde saptayan algoritmayı belirlemektir. Python dilinde yazılan 4 uzamsal arama algoritması yapılan testler ile karşılaştırılmış ve veri seti için en uygun algoritma belirlenmiştir. Tespit edilen algoritma veri setine benzer şehir bileşeni modelinde kullanılabilir bu sayede zamanın değerli olduğu şehir hayatında verimli zaman yönetimi sağlanmış olur.

**Anahtar Kelimeler:** Akıllı Şehirler, Uzamsal Arama, k En Yakın Komşu, Dörtgen Sorgular, r-Ağacı ve kd-Ağacı

| | | |
|---|---|---|
| **Geliş/Received** | **:** | **02.05.2020** |
| **Gözden Geçirme/Revised** | **:** | **05.05.2020** |
| **Kabul/Accepted** | **:** | **21.05.2020** |

## 1. INTRODUCTION

As mobile devices play a more active role in every aspect of our lives, the usage areas of the devices have increased and data types have developed and large related datasets have been formed. Compared to January 2019 data, 298 million new internet users have joined the internet users. Looking at the January 2020 data, the number of internet users reached 4.54 billion with an increase of 7%. The number of mobile phone users exceeded 5.19 billion, with a 2.4% increase worldwide, with 124 million new users compared to last year (We Are Social, 2020). These data types and inquiries are gaining more and more importance day by day. Spatial search is a kind of algorithms that arranges point or geometric data for efficient search.

As the cities become smart, the population served increases and the smart city components also increase in parallel. The efficiency of smart parking and smart transportation services is gaining importance. It is one of the smart city components in mobile applications that show the density of the car park or the closest empty car park. These services require distance calculation; the nearest bus stop, the nearest ferry port, the nearest ticket filling station, the distance of the nearest bus to the stop, etc. can be varied. Spatial search algorithms are used in mobile applications included in these services. The most important point is; for example performing queries like “find the nearest vehicle in this area”, “search 400 nearest park areas to this point”, and returning results within milliseconds even when searching millions of objects. Spatial searches performing in real time by a spatial situation, will require more processing power and more multifaceted spatial indexing methods and query computing tactics.

First of all, Benchmark comparisons will run on the python programing language to decide an optimum spatial search algorithm for data size, because python programing language is preferred solution for scientific computing. This makes Python the best choice for scientists and engineers (Oliphant, 2007)

Data set to be used for comparison; It includes the values of the x-axis and y-axis in the spatial space. The data set file containing 10877 locations is 750 kb in size. The "csv" file format was preferred to store the data. The feature of this file format is to create comma separated data lists with its simple structure. It is designed to be the way to export and import between databases. The sample dataset to be used shows that although it contains few locations and is in csv format, it will be processed in this spatial search with a size of 750 kb. For example; The Uber company, which makes more than ten million trips a day, will have to process more than 7000 gb of data only if it travels on this data set. With the comparison to be done for the sample data set, it is aimed to determine the correct algorithm for similar city component models. Benchmark comparisons will perform on 4 different spatial search algorithms. These algorithms are; k Nearest Neighbor (KNN), Range (rectangle) query, R Tree with KNN, KD Tree with KNN.

## 2. RELATED WORK

There are mobile applications born from smart cities. As an example, iTaksi mobile application, which belongs to Istanbul Metropolitan Municipality, Uber, Careem etc. taxi calling applications used worldwide. The main purpose of these and similar applications is to provide efficiency to the crowded smart city population in terms of time management. Spatial search algorithms determine the performance of these distance-oriented mobile applications. Which spatial search method is a preferred for which service is very important for the efficiency of the service.

Although spatial search algorithms need processing power, selecting an optimum spatial search algorithm more important. KNN algorithms has a simple calculation method but not for every data model or size.

The most popular classification method for data mining and statistical calculations is the KNN method because of easy application and outstanding classification calculation performance. However, it is not useful to perform basic kNN methods with a constant k value of all test applications, even if determined by experts (Zhang et al., 2017). Using the kNN algorithm for each data set can have a disadvantage. kNN can be very easy to use, but as the data set grows, the efficiency of the algorithm decreases quickly. KNN works well with a small number of queries, but as the number of queries increases, the kNN algorithm should run again to estimate the output of the new point. One of the major problems with kNN is to identify the most appropriate k-neighbor number when classifying the new query.

In the tendency of the studies in the literature if KNN algorithm combines with rectangle queries, decreasing processing costs are expected. The strategy determined by basic query algorithms; to increase the query performance by applying the measurement distance with pruning techniques on the tree directory. It should perform a large amount of distance calculations to ignore the area that is unquestionable at the stage of determining the next nearest neighbor object. The time spent calculating frequent distances in large amounts of data objects is many, which is a factor that greatly reduces the efficiency of the query. If the grid index is selected to apply the KNN query, the scope of the query will not be flexible, and many blank grid cells will be created with slow search speed and low storage usage (Li and Tang, 2010). By strengthening the rectangle queries with the KNN algorithm, the idea of creating "rectangle queries with kNN" has been created by increasing the search range according to the statistical ranges of the dataset indexes (Liu et al.,2002). Rectangle queries basically aim to find the closest point within an expanding area, but the point being searched brings some disadvantages when the data density is low, or if the nearest neighbor is in a far location. If we explain this; The rectangle query expands due to a coefficient of expansion, wherever it cannot find a neighbor in the area, it switches to the next expansion. This cycle will not continue forever, so it must be limited. This is called a query timeout or a query loop count. When the query starts, every quadrant has to search the same area over and over again in every expansion, and the loop ends and the time spent is lost before any result is reached in the cycle.

In another study, for spatial search R-tree method gave more effective results in the term of time costs for big size areas. When the results of the application are examined, it was observed that this method supports higher spatial use and decreases the time costs, as well as the number of indexes that can be increased. For this reason, this method is recommended correctly and effectively (Zhang et al., 2007). R-tree algorithms that provide stable performance in the wide area uses, It is one of the most promising spatial search algorithms because it does not search on empty nodes in the search area (Zhu et al., 2007). One of the weaknesses of this algorithm is that it shows a large area's success in small areas, but it has succumbed to k-NN and rectangle queries algorithms. It is not preferred in terms of time costs to create nodes by calculating trees in models where the data set and the area are not large. If we come to another weakness; When searching in areas where the spatial area is small but the data density is high, the area formed by the point crosses the algorithm by searching in many tree nodes where too many nodes intersect.

Another more preferred spatial search method in the literature is K Dimensional-tree (KD Tree) algorithm. According to the experiments KD Tree combine with KNN algorithm. With a unique storage method to represent application data efficiently, the KD-tree is a multi-dimensional binary tree. For this reason, a new classification algorithm is combined by using the strengths of KNN and KD tree. This algorithm is applied as KD-tree-KNN (Hou et al., 2019). Although the kd-tree is an easily applicable tree algorithm. as the number of sizes increases, the number of nodes and the number of nodes intersected by the area formed by the point to be searched will increase. In each query, it is necessary to re-create the nodes according to the data in the data set. As the number of data increases, the node creation process becomes longer. The increase in the number of data and the number of sizes in the data set creates a disadvantage for the kd-tree.

## 3. METHODOLOGY

Each algorithm described in this section is applied individually. Each algorithm will be analyzed individually in the continuation of the study.

### 3.1. k Nearest Neighbor Algorithm

One of the non-parametric traditional classification methods is the KNN algorithm (Cover and Hart 1967). As shown in the Figure 1. with KNN, basically the points closest to a point are calculated. K represents the amount of the closest neighbors of the center point. "Equation (1)" Usually, the Euclidean distance is used as the distance metric. The Euclidean method applies to calculate the distance between two points If A and B represent these points, the normalized the Euclidean metric is generally used by : (Hu et al., 2016).

$$dist(A,B) = \sqrt{\frac{\sum_{i=1}^{m}(x_i-y_i)^2}{m}} \tag{1}$$

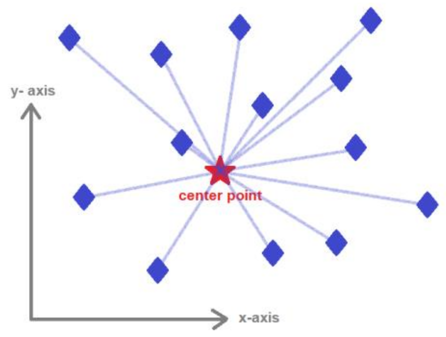

**Figure 1.** k-NN Model.

In order to calculate the closest point to the center point, the distance between all points in the data set and the center point should be calculated. The performance of the closest neighbor algorithm is critical. Therefore, different distance calculation methods can be preferred from the Euclidean method.

### 3.2. Rectangle Queries with Euclidean

A virtual rectangle is drawn around the center point to search for the nearest neighbor. By looking at the neighbors in the rectangle, the closest neighbor is calculated, and the out of rectangle points are ignored. If there is no neighbor in the virtual rectangle, the rectangle boundaries are expanded until the first neighbor is found or until the desired number of loop is finished as shown in the Figure 2. The multiplier parameter used to expand the rectangle boundaries in the loop varies. The Euclidean method is used for calculations in the rectangle.

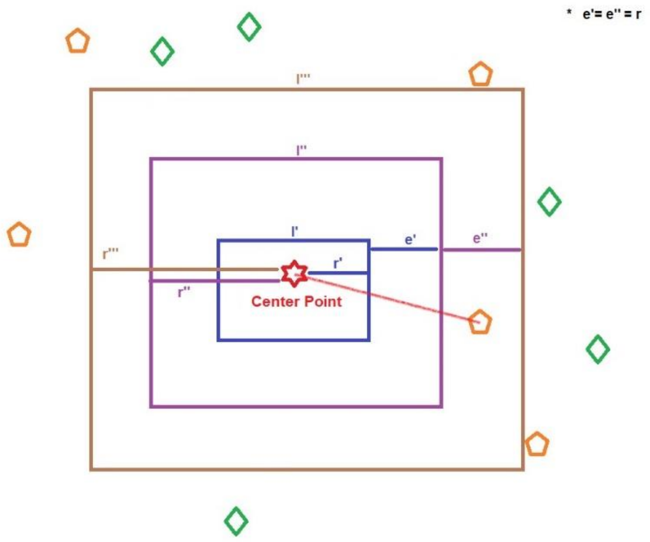

**Figure 2.** Rectangle Query Model.

The variables of the formula “Equation (2)” required for the virtual rectangle are defined as follows; number of loops "l", extension coefficient "e", rectangle boundary "r", coordinates of all neighbor on the axis “x,y”, for the rectangle's boundaries; maximum x-axis value "maxX", maximum y-axis value "maxY", minimum x-axis value "minX", minimum y-axis value "minY ", the center point’s x-axis value "centerX" and the y-axis value "centerY".

$$r = \sum_{i=1}^{l} e \tag{2}$$

After the rectangle boundary is obtained, the boundary coordinates of the rectangle are calculated “Equation (3)(4)(5)(6)” as follows;

$$maxX = basisX + r \tag{3}$$

$$maxY = basisY + r \tag{4}$$

$$minX = basisX - r \tag{5}$$

$$minY = basisY - r \tag{6}$$

After determining the coordinates of the boundaries, whether there are any points in the created area is checked "Equation (7)" by comparing boundary coordinates with the x and y axes of all points;

$$(x < maxX \ \wedge \ x > minX) \ \wedge \ (y < maxY \ \wedge \ y > minY) \tag{7}$$

After finding the coordinates of the points within the boundaries, the Euclidean method is used for the nearest neighbor calculation.

### 3.3. r-Tree with Euclidean

The main working logic of the r-tree; limiting and dividing the points in the spatial space with minimum-sized rectangles. The main partitioning is done according to the density of the points. Each section is partitioned in a balanced way. This process is repeated as needed depending on the size and density of the data as shown in the Figure 3. In the last case, balanced cluster tree is obtained.

If working on a directory located on the disk, nodes and disk pages match. A spatial search structure is created that should runon a minimum number of nodes (Guttman, 1984).

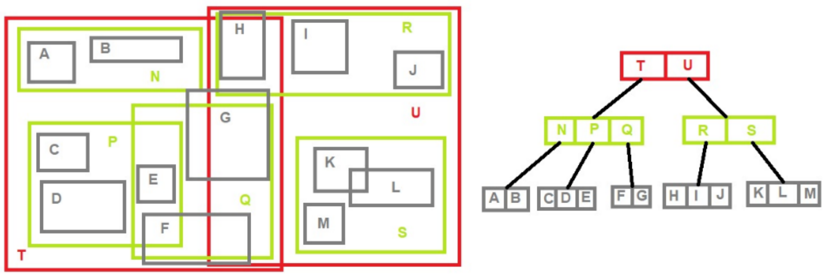

**Figure 3.** r-Tree Structure Model.

A virtual rectangle is created around the center point for the search. The distance of the center point to the boundaries of the created rectangle is a variable. This variable is determined by the magnitude of the tree structure, data and area.

In the next step, find out which rectangles coincide in the r-tree with the rectangle obtained from the center point. The rectangle of the r-tree is the area to be searched. Except for these rectangles, the result of the intersection is ignored. The Euclidean method is applied to find the nearest neighbor for these rectangles.

### 3.4. kd-Tree with Euclidean Method

This is a binary tree. Each node has two sub-nodes, right and left. Each level divides the area over a certain size. Usually, the midpoint is selected as root at each level.

For example, in the 2-dimensional (x, y) space, all children are divided in the root node (first level) according to the x-axis, points with the value of x greater than x go to the right node of the tree, points with the smaller value of X go to the left node. Nodes newly created in the second level are made on the y-axis using the same criteria. At the third level, you return to the x-axis. According to the size of the data and the field, the division repeats itself as much as "n" with the same criteria as shown in the Figure 4.

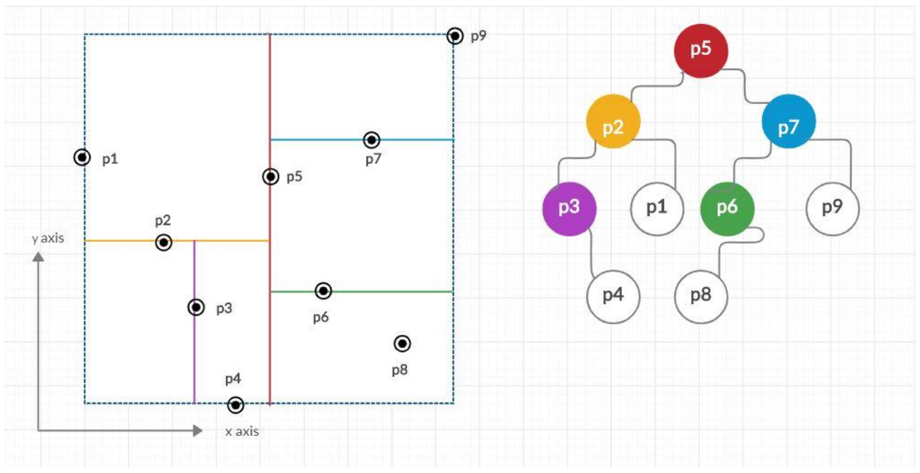

**Figure 4.** kd-Tree Structure Model.

After the nodes are formed; It is also calculated in which node the center point is located. After this stage, the Euclidean method is applied for the points within the boundaries of the node we have determined. The nearest neighbor as "k" can be found in the spatial area divided in dimensional (d).

This finalizes the search of KD-tree-KNN. If you want to apply K neighbors search, you need to search more than one time (multiple times) to determine these K neighbors (Hou et al., 2019).

## 4. APPLICATIONS

### 4.1. Simulation Studies

Simulation studies were carried out on a cloud machine with 2 vCPU, 24 Gb ram capacity with windows server 2019 operating system. The 4 spatial search algorithms processed in the methodology section are written in the python programming language. The data set needed to test these algorithms was used by including the algorithms in csv format. The data set contains 10877 different locations with a size of 750 kb. Locations are the coordinates of the locations taken randomly from the country of Greece. These positions are created from latitude and longitude values.

The world coordinate values are considered as x and y axes. A 2-dimensional plane is obtained with the maximum x value 180, the minimum x value -180, the maximum y value 90 and the minimum y value -90 as shown in the Table1. In this framework, 4 algorithms have been tested.

**Table 1.** 2d Area and Dataset Limit Coordinates.

| | Maximum x-axis value | Maximum y-axis value | Minimum x-axis value | Minimum y-axis value |
|---|---|---|---|---|
| **2D Area** | 180.0 | 90.0 | -180.0 | -90.0 |
| **Dataset** | 40.599272 | 22.97482 | 40.56548 | 22.94119 |

The nearest neighbor in the data set has been accepted as the central point (40.59, 22.9598). Limit values in the data set; the maximum x-axis value is 40.599272, the minimum x-axis value is 40.56548, the maximum y-axis value is 22.97482, the minimum y-axis value is 22.94119.

Rectangle Queries with Euclidean algorithm needs an extension coefficient "e" to draw and expand a virtual rectangle around it. As shown in the Table 2; In this fixed data set, this extension coefficient is used as 1/10.000 due to changes in the values of the data less than 1/10.000.

**Table 2.** Center Points Coordinates and Extension Coefficient (e).

| | x-axis coordinate | y-axis coordinate | e |
|---|---|---|---|
| Center point's coordinates and e | 40.59 | 22.9598 | 0.0001 |

The area of the data set for the r-tree is divided into 4 parts. The x and y axis boundaries of these parts are as follows;

1. Node boundaries; maximum x value is 40.60, minimum x value is 40.58, maximum y value is 22.98, minimum y value is 22.96.

2. Node boundaries; the maximum x value is 40.58, the minimum x value is 40.56, the maximum y value is 22.98, the minimum y value is 22.96.

3. Node boundaries; the maximum x value is 40.58, the minimum x value is 40.56, the maximum y value is 22.96, the minimum y value is 22.94.

4. Node boundaries; the maximum x value is 40.60, the minimum x value is 40.58, the maximum y value is 22.96, the minimum y value is 22.94.

Within the scope of the algorithm of the r-tree; the intersection of the virtual rectangle surrounding the center point with these rectangles should be checked. It is calculated as + - 0.005 from the x and y axis values of the virtual quadrangular point surrounding the central point. As a result, the boundaries of the rectangle belonging to the center point are as follows;

The boundaries of the center square; the maximum x value is 40,595, the minimum x value is 40,585, the maximum y value is 22.9648, the minimum y value is 22.9548 as shown in the Table 3 and Figure 5.

**Tablo 3.** Nodes and Center Point's Rectangle Coordinates.

| | Maximum x-axis value | Maximum y-axis value | Minimum x-axis value | Minimum y-axis value |
|---|---|---|---|---|
| r-Tree Node1 | 40.60 | 22.98 | 40.58 | 22.96 |
| r-Tree Node2 | 40.58 | 22.98 | 40.56 | 22.96 |
| r-Tree Node3 | 40.58 | 22.96 | 40.56 | 22.94 |
| r-Tree Node4 | 40.60 | 22.96 | 40.58 | 22.94 |
| Center point's rectangle | 40.595 | 22.9648 | 40.585 | 22.9548 |

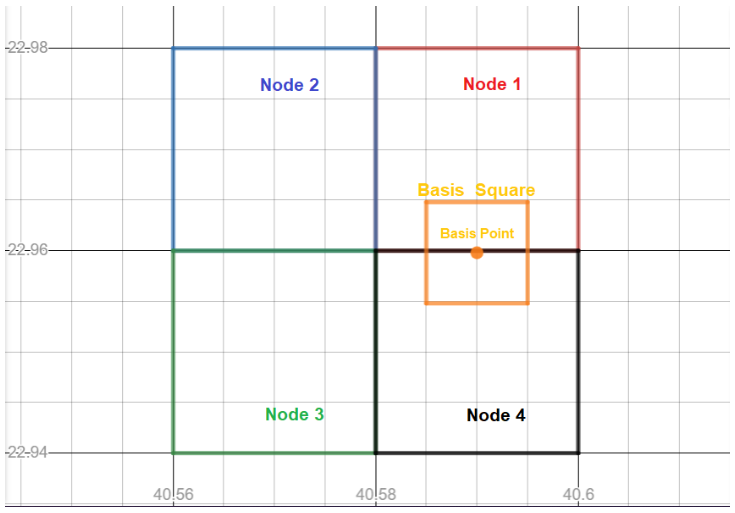

**Figure 5.** r-Tree Simulation Model.

## 4.2. Simulation Studies' Results

Firstly, 50 tests were performed with the nearest neighbor. As a result of the tests, 50/50 correct results have been reached and the average processing time to find the nearest city component has been calculated as 23.844 ms.

50 tests were performed with the rectangle queries with the Euclidean algorithm. As a result of the tests, 50/50 correct results have been reached and the average processing time to find the nearest city component has been calculated as 14.84175 ms.

50 tests were performed with the r-tree algorithm. As a result of the tests, 50/50 correct results have been reached and the average processing time to find the nearest city component has been calculated as 44.59282 ms.

Finally, 50 tests were carried out with the kd-tree algorithm. As a result of the tests, 50/50 correct results have been reached and the average processing time to find the nearest city component has been calculated as 305.2153 ms.

Algorithms' run time test results compared as shown in the Figure 6. The city component finds processes' run time average values calculated and compared as shown in the Figure 7.

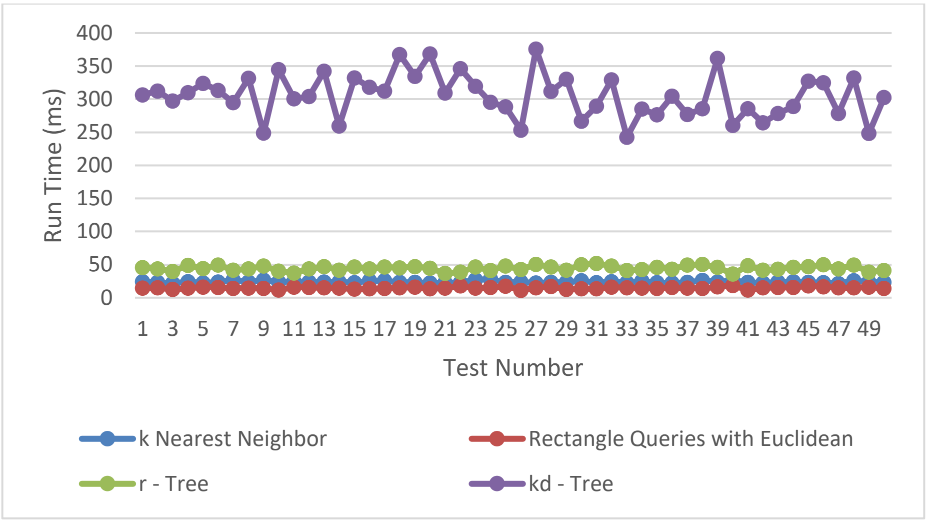

**Figure 6.** Test Algorithms' Run Times Comparing.

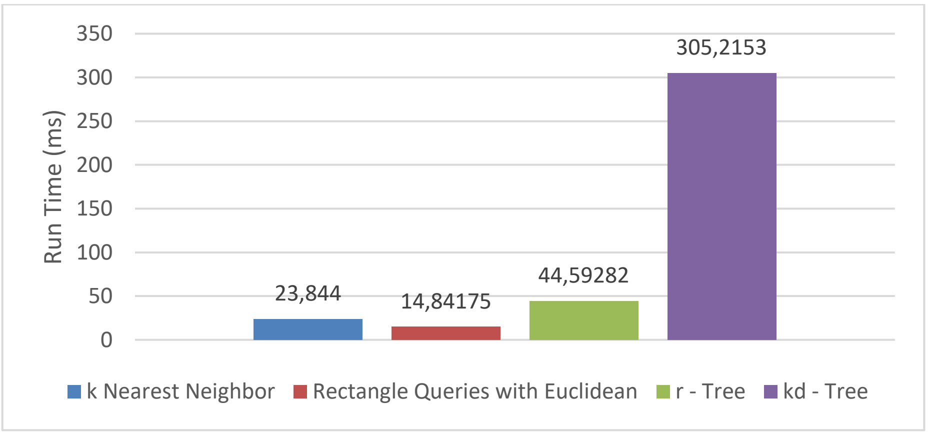

**Figure 7.** Average Run Times Comparing of Test Algorithms.

## 5. CONCULUSION

In this paper, According to 200 tests; It has been determined that 4 algorithms give 100% correct results. Rectangle queries with the Euclidean algorithm creates a rectangle that expands around the center point and searches inside it. When no neighbor can be found in the rectangle, the rectangle boundaries are expanded and the search continues. When the center point is selected from an area where the points in the data set are dense for the nearest neighbor; the algorithm that finds the fastest result; rectangle queries with the Euclidean algorithm. If the distribution area and size of the data set is a model for a smart city component, rectangle queries with the Euclidean algorithm can be preferred as the optimum solution.

The fastest algorithm following rectangle queries with Euclidean algorithm is the KNN algorithm because it uses simpler calculation methods than tree algorithms. The disadvantage of the KNN algorithm is that; it should calculate for all points in the data set.

It was observed that tree algorithms are not needed when the data set and distribution area is small. Because tree algorithms use more complex calculations methods.

In the future, In order to observe the performance of tree algorithms more clearly, tests can be performed by combining different size datasets and areas with different center point locations. In these tests, we plan to use different center points, multiple center points and big data set to model smart cities of different sizes and their components.

## REFERENCES

Cover, T., Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 13, 21-27.

Digital in 2020. We are Social, https://wearesocial.com/digital-2020

Guttman, A. (1984). R Trees: A Dynamic Index Structure for Spatial Searching. Sigmod Record, 14, 47-57.

Hou, W., Li, D., Xu, C., Zhang, H., Li, T. (2019). An Advanced k Nearest Neighbor Classification Algorithm Based on KD-tree. 2018 IEEE International Conference of Safety Produce Informatization (IICSPI), 10-12 Dec. 2018, Chongqing, China.

Hu, L., Huang, M., Ke, S., Tsai, C. (2016). The distance function effect on k-nearest neighbor classification for medical datasets. SpringerPlus, 5, 1304.

Li, G., Tang, J. (2010). A New K-NN Query Algorithm Based on the Traversal and Search of the Dynamic Rectangle. 2010 International Conference on E-Product E-Service and E-Entertainment, 7-9 Nov. 2010, Henan, China.

Liu D, Lim E, Ng W. (2002). Efficient k nearest neighbor queries on remote spatial databases usingrange estimation. In: Proceedings of international conf. on scientific and statistical databasesmanagement (SSDMB), 24–26 July 2002, pp 121–130, Edinburgh.

Oliphant, T. E. (2007). Python for Scientific Computing. Computing in Science & Engineering, 9, 10-20.

Zhang, S., Li, X., Zong, M., Zhu, X., Wang, R. (2017). Efficient kNN Classification With Different Numbers of Nearest Neighbors. IEEE Transactions on Neural Networks and Learning Systems, 29, 1774-1785.

Zhang, Z., Zhang, J., Yang, J., Yang, Y. (2007). A New Approach to Creating Spatial Index with R-Tree. 2007 International Conference on Machine Learning and Cybernetics, 19-22 Aug. 2007, Hong Kong, China.

Zhu, Q., Gong, J., Zhang, Y. (2007). An efficient 3D R-tree spatial index method for virtual geographic environments. ISPRS Journal of Photogrammetry and Remote Sensing, 62, 217-224.